COMPUTATIONAL INVERSE MECHANICS OF A HIGHLY COMMINUTED TIBIA FRACTURE

by

Waseem Ghassan Tahseen Shadid

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

2014

Approved by:

_____
Dr. Andrew R. Willis

_____
Dr. Ehab Al-Shaer

_____
Dr. Ivan Howitt

_____
Dr. Thomas Weldon

ABSTRACT

WASEEM GHASSAN TAHSEEN SHADID. Computational inverse mechanics of a highly comminuted tibia fracture. (Under the direction of DR. ANDREW R. WILLIS)

This dissertation presents a new computational system that seeks to estimate the inverse mechanics of a bone fracture from 3D CT images. One image provides a model of an unbroken bone and a second records a fractured example of the same bone after an injury. For a bone fracture, these mechanics specify how the bone broke in terms of what object impacted the limb and how the bone fracture fragments moved over time due to that impact. To accomplish this task, novel image processing techniques are used in combination with existing computational dynamics simulation tools. Estimates of a fracture event are generated in three steps: (1) estimate a 3D model of the limb immediately before the fracture event occurred, (2) iteratively simulate the fracture dynamics to search for values of the unknown fracture variables that produce fracture patterns similar to that depicted in the observed image of the fractured limb, and (3) visualize and analyze likely fracture scenarios in a virtual 3D environment. This dissertation investigates the following two challenges: (1) obtaining a physically-meaningful estimate of the unbroken limb and (2) solving the difficult search problem for the unknown fracture event variables. Challenge one requires geometric surface models having unprecedented accuracy to be estimated for the fracture fragments which is a difficult unsolved problem. Challenge two requires conceptualizing and implementing a completely new system to estimate the fracture event. Results for the proposed methods are provided for clinical tibial plafond fracture data.

ACKNOWLEDGMENTS

First and above all, I praise God, the almighty for providing me this opportunity, for granting me the capability to proceed successfully, and to whom I owe my very existence. This dissertation appears in its current form due to the assistance and guidance of several people. I would therefore like to offer my sincere thanks to all of them.

There are no proper words to convey my deep gratitude and respect for my dissertation and research advisor, Professor Andrew Willis. I would like to thank him for the trust, the insightful discussion, offering valuable advices, his support during the whole period of the study, and especially for his patience and guidance during the writing process. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. He has inspired me to become an independent researcher and helped me realize the power of critical reasoning. He also demonstrated what a brilliant and hard-working scientist can accomplish. Without his helpful discussions and directions, this dissertation would not have been possible.

I would like to thank the members of my Ph.D. committee: Prof. Ehab Al-Shaer, Prof. Ivan Howitt, and Prof. Thomas Weldon, for their valuable comments toward improving my work and for their teachings during my Ph.D. education in classes and projects. In particular, Prof. Ehab Al-Shaer accepted to serve as the graduate school representative and offered me daily life advices during my study. Also, I would like to express my thanks to folks in the University of Iowa who provided me with real

clinical data to help me complete my dissertation.

I owe my deepest gratitude towards my parents, Ghassan Shadid and Samira Hamidi, for loving me, for raising me, for bearing with me during the challenges, for joining me in the triumphs, for never damping my sense of curiosity, for never imposing, for allowing me to be as ambitious as I wanted, and for never having anything but confidence in me. I thank my mom for the endless prayers and for showing me the importance of valuing the people in my life, my dad for infecting me with a critical eye and the need to know how things and humans work. It was under their watchful eye that I gained so much drive and an ability to tackle challenges head on. Mom and Dad, you are wonderful parents and wonderful friends.

These acknowledgments would not be complete without expressing my deepest gratitude towards my better half, my wife Duaa Makhoul, for her eternal support and understanding of my goals and aspirations. Her infallible love and support have always been my strength and the bedrock upon which my life has been built. Her patience and sacrifice will remain my inspiration throughout my life. Without her help, I would not have been able to complete much of what I have done and become who I am. It would be ungrateful on my part if I thank her in these few words. I am thankful to my son Laith for giving me happiness during the last three years of my studies. He is the best gift I have ever received.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

In order to understand how a bone fracture is generated, one needs to understand how the bone fragments moved from their original anatomical positions to their fractured positions as presented by a fracture case. The absence of computational systems like this prevent physicians from visualizing plausible reconstructions of the fracture event. For traumatic fracture cases, this information can provide important forensic, diagnostic and treatment insights for these difficult cases [46]. Current technologies do not provide this information which is important, particularly for complex fractures where multiple bone fragments are generated. Important information derived from reconstruction of the fracture event may lead to a different diagnosis or treatment. Existing approaches for orthopedic image analysis 2D and limited 3D analysis methods for bone fractures despite their helpfulness to orthopedic physicians. Such systems are helpful to orthopedic physicians because they allow them to analyze 3D Computed Tomography (CT) images and to visualize fractures in three dimensions. A subset of these systems go further and provide a virtual reconstruction of bone fragments that physicians can use to assist in planning for surgical reconstructions [26, 83]. The system described in this dissertation provides new visualization and computational methods for analysis that may have important clinical benefits.

This dissertation describes a system that seeks to estimate the geometry and dynamics of a fracture event, i.e., the process of breaking a bone into fragments and

moving these fragments away from their anatomic positions. The system requires the following data as input: (1) a 3D CT image of a fractured limb, referred to as a fracture image, (2) a 3D CT image of an unfractured limb, referred to as an intact image, and (3) a collection of hypothesized objects that could break the bone, referred to as strike objects. The outputs of the system are: (1) an estimate of the hypothesized object that broke the bone (the strike object) and (2) a virtual simulation of how bone fragments moved from their original positions to their fractured positions in the fracture as presented by the 3D CT image of the fractured limb, referred to as a fracture simulation. The user interacts with the system to provide input data and visualize the output fracture simulations. The user helps the system construct estimates for the fracture event on the input side and visualizes and analyzes the fracture simulation scenarios results at the output side.

Fracture simulations must be estimated from the provided 3D CT data. The system obtains estimates for likely fracture simulation given the user input and data through the following three steps:

1. Estimate the initial fracture context. The initial fracture context is a virtual model of the limb and strike object at some time t,

2. Search for values of the unknown fracture event variables that produce a fracture simulation that maximizes the likelihood of the observed image of the fractured limb,

3. Users analyze fracture simulations having high likelihood scores.

The listed sequence describes a process for computational virtual reconstruction of a

fracture event.

Step 1 takes three inputs: (1) a fracture image, (2) an intact image, and (3) a collection of hypothesized strike objects that could have broken the bone. The output of this step is a model having geometry and physical attributes that serves as a virtual 3D model of the limb at the instant the bone was fractured, referred to as the initial fracture context. The 3D models specify both the object shape and physical attributes including the object's density and coefficient of friction. With the exception of the soft tissue model, these models are rigid and do not deform over time.

The initial fracture context includes representations for three different types of objects: (1) fracture fragments, (2) soft tissue of the fractured limb, and (3) the strike object. The geometry of the fracture fragments and soft tissue is unknown and must be estimated from the image data. This makes estimation of the initial fracture context a challenging problem and it requires solutions to several sub-problems. The sub-problems are: (1) estimating the geometry of the fracture fragments and their original anatomic positions, (2) estimating the geometry of the soft tissue, (3) estimating the geometry of the strike object, and (4) assigning appropriate physical attributes to all these objects.

For the first sub-problem, the system extracts the geometry of fracture fragments from the fracture image. Obtaining accurate estimates of the surfaces was not possible with existing surface extraction algorithms. This dissertation describes a novel algorithm for computing these surfaces which is one of the contributions of this dissertation [74]. An existing system for virtual reconstruction of bone fragments is

applied to virtually reconstruct the unbroken bone from the extracted bone fragment surfaces.

For the second sub-problem, the system estimates a volumetric model for the soft tissue from the intact image. The model consists of a connected grid of 3D elements that represent positions of soft tissue structures in the fractured limb immediately before the fracture event occurred.

For the third sub-problem, the system takes a user-selected strike object from a list of potential strike objects. The geometry of the strike object is assumed to approximate the shape of the real object that fractured the limb. For the fourth sub-problem, the system takes user-specified values for the physical attributes of the objects. Physical attributes for fracture fragments include density and friction. The physical attributes for soft tissue models include density and friction for each 3D element, and criteria for how those elements are connected. The physical attributes for the strike object model include: mass, friction, initial position and orientation, and velocity. The physical attributes control the physical behavior of the attributed models in a fracture simulation. By solving these sub-problem, an estimate of the initial fracture context is obtained.

Step 2 searches for the unknown values of the fracture event that produce fracture simulations that best match the observed image of the fractured limb. The search applies an optimization procedure to find a subset of fracture event values that are likely to have generated the observed fracture event data. In this dissertation, the fracture event variables are assumed to be the mass, scale, velocity, position, and direction of movement for the strike object. These variables are changed during

the search process and optimized by minimizing a likelihood energy function. The likelihood energy function expresses the likelihood of the data given values of the fracture event variables. Likelihood decreases as the difference between the fracture pattern generated by a fracture simulation and the fracture pattern observed in the fracture image becomes large. The optimization process is an iterative and each iteration requires a fracture simulation which provides a value for the likelihood energy function when complete. After each iteration, the fracture event variables are updated to improve this likelihood. This process continues until a local maximum likelihood value for the fracture event variables is found.

Step 3 provide a virtual 3D fracture environment to the user where they can navigate the computed solutions and visualize their associated fracture simulations. Typical search solution provide multiple plausible fracture event variable values which are stored in a list of fracture event variable values. Users browse this list to explore the space of plausible solutions. For each solution, a fracture simulation result may be viewed which animates the bone fragment surfaces in a virtual 3D environment. The simulations result shows how the bone fragments moved over time during the fracture event. Additional analysis tools provide views of the simulated 3D models and tracks their positions in the intact and fracture CT images. This allows users to visually approximate the trajectory of the bone fragments through the soft tissue which provides clues about the location and extent of the voids created in soft tissue due to the fracture, i.e., areas in the fractured limb where soft tissue is pushed away from its original position. The analysis tool highlights the voids in the soft tissue created by computing soft tissue regions that have been displaced by the motion of

bone fragments during the simulated fracture event. These areas are assumed to correspond to damaged tissue regions in the fractured limb.

The dissertation uses data from clinical cases of a tibial plafond fracture. This type of fracture happens when a high-energy axial impact is inflicted on the foot. In this dissertation, tibial fracture cases are of importance for several reasons: (1) the complexity of this type of fracture creates difficulties for physicians in making accurate and reliable treatment decisions, (2) the ankle joint is usually involved in tibia fractures and is typically difficult to treat when multiple bones that make up the ankle joint are broken, and (3) the quality of tibia fracture reconstruction is an important factor for the long term health of the patient. Due to these reasons, the tibial plafond fracture cases are the focus example of this dissertation.

1.1 Structure of the Dissertation and Terminology

This section describes the structure of the dissertation, the formatting of the dissertation, and the terminology of the dissertation. This helps readers in understanding the discussion throughout the dissertation.

1.1.1 Structure of the Dissertation

This dissertation is organized as follows: Chapter 1 introduces the dissertation topic, the problem of estimating the inverse mechanics of a fracture, and the contributions of this work to the state-of-the-art of the computational methods for medical imaging. Chapter 2 introduces important background information on bone anatomy, digital imaging, the X-ray CT imaging modality, image segmentation, and dynamics simulation. Chapter 3 reviews state-of-the-art in computer based bone fracture anal-

ysis systems and medical image segmentation algorithms. Chapters 4 and 5 describe the system interface and the novel underlying algorithms that the user interface applies to estimate the inverse mechanics of a fracture event with the exception of the image segmentation algorithm. Chapter 6 presents experimental results for several fracture simulations for a high comminuted tibia fracture. Chapter 7 describes the bone fragment segmentation algorithm used to segment fragments within a 3D CT image of a fractured limb which was excluded from chapters 4 and 5. Chapter 8 provides segmentation results for six clinical fracture cases and quantitatively compares these results with several competing methods and with human-generated segmentations. Chapter 9 summarizes the implemented work of this dissertation and its impact on the current state of the research in this field.

1.1.2 Formatting of the Dissertation

Several formatting decisions have been made to improve the clarity of this document. These decisions are:

1. Bold words are new terms introduced in this dissertation,

2. Bold small case variables indicate vector quantities,

These formatting decisions cue the reader to take note of these important components to this dissertation.

1.1.3 Terminology of the Dissertation

Terminology used throughout this document is defined here to clarify and simplify the discussion. These terms are:

1. Fracture image: A 3D CT image of a fractured limb,

2. Intact image: A 3D CT image of an unbroken limb,

3. Highly comminuted fracture: A bone fracture that involves three or more fragments,

4. Fracture event: The process of breaking a bone into fragments and moving these fragments away from their anatomic positions,

5. Inverse mechanics of a bone fracture: A method to compute a fracture event in terms of what object fractured a limb and how bone fracture fragments moved over time from their original anatomical positions to their fractured positions in a fractured limb based on the dynamics of fracture objects,

6. Attributed model: The combination of the geometry and physical attributes of a model. The geometry of a model specify its shape while the physical attributes specify the material properties of the model, for example, density and friction,

7. Strike object: An attributed model for an object that broke a bone,

8. Fracture context: A virtual attributed model for a fractured limb and a strike object at some time t,

9. Dynamics simulator: A system that seeks to approximate the behavior of objects in space according to Newton's laws of dynamics,

10. Fracture simulation: A virtual simulation of how bone fragments moved from their original positions to their fractured positions in a fracture case,

11. Dynamics optimization: A search for the conditions that produce a fracture simulation that best match the image of a fractured limb,

12. Void in soft tissue: Areas in the fractured limb where soft tissue is pushed away from its original position.

13. Image segmentation: A process of assigning labels to the pixels of an image so that each group of pixels that have the same label forms a unique semantic class,

14. Watershed algorithm: An image segmentation method that starts the labeling process from an initial group of labeled pixels and expands the labels to adjacent pixels according to their intensity values in away analogical to rain falls on a topographic relief,

15. Leak: A segmentation problem that assigns pixels wrong labels for neighboring segments these pixels are supposed to belong to,

16. Object position: The location and orientation of an object in space,

17. Reconstructed bone: A collection of aligned geometric fracture fragment models that approximate the original unbroken bone,

18. "Isthmus": a bone area where small parts of different bone fragments with small depth appear touching each other in a CT image.

These terms are used throughout the dissertation and the definitions here help the reader understands the formal definitions for these terms.

1.2 Motivation

According to the American Academy of Orthopedic Surgeons (AAOS), about 6.8 million people need medical attention for bone fractures in the United States each year. Typical treatment involves Orthopedic surgeons moving the bone fragments to reconstruct surgically the normal anatomy of the broken bone [6]. Surgeons often carried out these surgeries either by cutting through skin and soft tissues to move bone fragments to their aligned positions or by attaching manipulators to bone fragments through the skin and then moving the bone fragments to their aligned positions using these manipulators, referred to as minimally invasive surgery [84]. It is hoped that the technologies developed in this dissertation may improve reconstructive surgical treatment and reduce the soft tissue trauma in invasive surgery, and reduce soft tissue trauma in minimally invasive surgery.

Orthopedic surgeons often seek to reconstruct the normal anatomy of the broken bone through surgery. One of the challenges that faces orthopedic surgeons is not knowing a valid reconstruction sequence for bone fragments in terms of which fragment goes where and when. For example, during surgery, surgeons put the first to fragments together and fix them with a screw. But after being fixated to their reconstructed position, these fragments obstruct other pieces from being correctly aligned. Therefore, the surgeons may have to extend surgery time. Consequently, extended surgery time often leads to higher rates of infection and increase the likelihood of a poor prognosis of the patient [60]. Systems that estimate the inverse mechanics of a fracture may help surgeons determine a valid reconstruction sequence for the bone

fragments during preoperative planing.

In invasive surgeries, surgeons have to cut through skin and tissue to put the fragments back together. If surgeons know what tissue has been already damaged then cutting there creates less damage. So that, soft tissue trauma due to the surgery for the reconstruction is reduced by knowing the path of the fragment. The path that the fragment moved through is tissue that has been damaged because the fragment moved through it. Furthermore, in minimally invasive surgery, surgeons stick a pin into a bone fragment then they push the fragment around by moving the pin instead of cutting the skin open. So related to soft tissue trauma, if they know the path way that already has low resistance because the fragment already moved through that path to get there, then that is a path that they can follow in a minimally invasive surgery with manipulators to move the fragment back to its normal position. Reduced soft tissue trauma may help in reducing the infection rate and increase the likelihood of a poor prognosis of the patient[28].

1.3 Contribution

The contributions of this dissertation fall in two areas:

1. A new system to estimate the inverse mechanics of a bone fracture from a CT image,

2. A new bone fragment segmentation algorithm from a CT image.

These two contributions establish new techniques in the field of medical image processing.

The first contribution is a system that estimates the computational inverse mechanics of a bone fracture. Currently, there are no existing approaches to solve this problem. While existing approaches are present for analysis of bone fractures and for virtual reconstruction of bone fractures, these systems fall short in that they do not represent the actual breaking process. This work represents a novel addition to this body of research that is unexplored to date.

The second contribution is a new bone fragment segmentation algorithm that generates a representation for the geometry of bone fragments within a CT image. The segmentation algorithm uses a modified version of the watershed algorithm to utilize a set of probability distributions. Where, each probability distribution is a combination of two novel probability functions. One probability function to segment individual bone fragments and a second to prevent merging bone fragments that are in close proximity. There are three innovations associated with this algorithm: (1) the formulation of a Probabilistic Watershed Transform (PWT), a modified version of the classical watershed transform, (2) unique probability functions for segmenting bone fragments in 3D CT images using the PWT, and (3) a unique probability function to avoid merging bone fragments that are in close proximity which is an acknowledged shortcoming of watershed transform methods.

The first contribution of the segmentation algorithm is the formulation of the PWT algorithm. The PWT algorithm is similar to the work of Grau [34] in which also solves a set of probability distributions to control the segment growth in the watershed transform. However, this work introduces new probability distributions having use for bone fragment segmentation. In the work of Grau, probability distributions represent

the prior probability that a pixel lies on a surface boundary, i.e., it separates two segments. In contrast, the probability distributions in the proposed PWT algorithm defines a stochastic relationship between each unknown pixel label and a collection of candidate segment labels. This allows the use of probability functions having different probabilities, one for each segment.

The second contribution of the segmentation algorithm describes unique probability distributions to segment bone fragments within a CT image using the PWT algorithm. The presented bone fragment segmentation algorithm uses the PWT algorithm to divide a CT image into semantic parts so that each part represents a unique bone fragment. The probability distribution represents the probability that a given pixel is a measurement obtained from each of the provided semantic classes. The probability distribution is computed using the intensity and position data of the image pixels in order to increase the algorithm robustness to image inhomogeneities. The framework of the PWT algorithm is not limited to the probability distributions that are specified in this dissertation. This framework is able to operate on any probability distributions. The probability distributions that are specified in this dissertation address several known shortcomings of segmenting bone fragments using the classical watershed transform method and several shortcomings of other competing segmentation method. Using these probability distributions, one is able to apply the PWT algorithm to segment bone fragments within a CT image of a bone fracture.

The third contribution of the segmentation algorithm describes a probability function to avoid incorrect merging of areas that correspond to different bone fragments within a CT image. This incorrect merging problem is referred to as a leak problem.

Leak problems cause two or more bone fragments to be represented by a single semantic class in the segmentation algorithm. A leak problem occurs when pixels are given incorrect labels and these incorrect labels are expanded to unclassified adjacent pixels. This problem is highly likely to occur in an area that corresponds to a small width part of a bone fragment within a CT image. This area is referred to as a narrow bone area. The intensity data of the pixels in narrow bone areas in a CT image may not correspond to the actual bones due to blurring and low image resolution. The introduced solution seeks to assign low probability values to the pixels of narrow bone areas in a CT image. So that, in a segmentation algorithm, those pixels are assigned labels after assigning labels to all pixels outside narrow bone areas. Labeling the pixels of narrow bone areas at the end will reduce the number of unclassified adjacent pixels that may be given incorrect labels.

CHAPTER 2: BACKGROUND

This chapter provides background information that is needed to understand the terminology of this dissertation and to interpret the dissertation results. The chapter consists of an overview of six topics: (1) bone anatomy, (2) the comminuted tibial plafond fracture, (3) medical images, (4) 3D Computed Tomography (CT) imaging, (5) image segmentation, and (6) dynamics simulation. This dissertation proposes new approaches that integrate work from these topics to provide novel computational bone fracture analysis software that uses CT images as input.

2.1 Bone Anatomy

A bone is a dense, semi-rigid organ that composes most of the human skeleton. Skeletal bones perform several functions for the body that include: providing shape and support for the body, protecting organs of the body such as the brain, producing blood cells, and storing minerals. A bone consists of different tissues including: osseous, endosteum, periosteum, nerves, blood vessels, marrow, and cartilage [89].

Osseous tissue is of particular importance to the analysis methods of this dissertation because bones primarily consist of osseous tissue. Specifically, this dissertation proposes methods to extract bone fragments from an image that depend almost exclusively on the appearance of osseous tissue in an image, see chapter (7). Osseous tissue is a hard and lightweight tissue that is primarily formed from calcium phos-

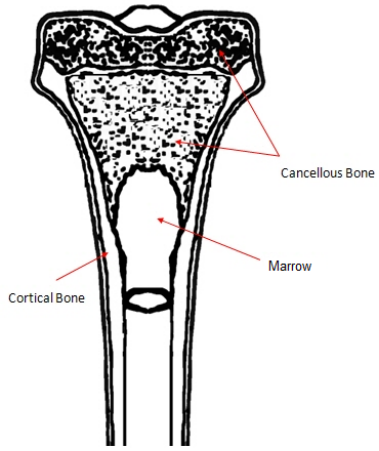


Figure 1: shows the location of different bone tissues within the proximal (top) end of the human tibia. Cortical bone and cancellous bone form the outer and inner layers of the bone, respectively.

phate. Osseous tissue consists of two types of tissue which are: cortical (compact) bone and cancellous (trabecular or spongy) bone, see figure (1). Cortical bone tissue typically forms the outer shell of bones. It is hard, dense, strong, and responsible for the bone's smooth white appearance. Cortical bone makes up 80% of the total bone mass of the skeleton. Cancellous bone tissue is a less dense inner layer of a bone. Cancellous bone consists of a network of rod-and-plate like elements that allow space for blood vessels and bone marrow making cancellous bone lighter and softer than the cortical bone. Cancellous bone makes up 20% of the total bone mass of a skeleton and has a surface area that is about ten times larger than the surface area of cortical bone.

2.2 Comminuted Tibial Plafond Fracture

This dissertation uses as source data a collection of clinical tibial plafond fractures. For this purpose, the medical nature of tibial plafond fracture and some details about its terminology are provided in this section. A comminuted tibial plafond fracture

is both a comminuted fracture, i.e., a fracture that involves many bone fragments, the fracture involves the tibia so have the term "tibial", and it is a specific type of fracture called a plafond fracture. A plafond fracture is a fracture that includes a specific sub-structure of the tibia bone referred to as the articular surface. An articular surface is any surface against which two different bones articulate, i.e., bone surfaces at bone joints. In particular, the articular surface referred to in a tibial plafond fracture is the articular surface at the distal end, i.e., bottom part of the tibia, these surfaces associated with the ankle joint. Comminuted plafond fractures usually occur in the hips, knees, ankles, shoulder, and wrists. The incidence of tibial plafond fractures accounts for 7% of all fractures of the tibia and less than 1% of all fractures of the lower-extremity [70]. Although tibial plafond fracture is not the most common joint injury, it has fracture characteristics similar to the ones of more frequent joint fractures such as the tibial plateau fracture, i.e., a fracture of the proximal tibia. Therefore, new computational technologies developed for the ankle joint fractures may be successfully applied to fractures elsewhere. Tibial plafond fractures are caused by axial compression, bending, and shearing forces. A comminuted tibial plafond fracture is typically the result of a severe compressive talar impact that is axially applied to the distal tibia. The other forces become more influential as the apposition of the tibia and talus deviates from the original anatomic positions [71]. A severe axial compressive force usually generates an explosion of bone fragments, whereas the other forces generate oblique and split fractures in varying degrees. In severe fracture cases, the fracture pattern extends from the articular surface to the diaphysis, cutting across both cortical and cancellous bone regions.

2.3 Medical Images

This dissertation proposes algorithms that process medical image data to perform advanced virtual fracture analysis. This section provides background information to understand the nature of this data and how image data is denoted with mathematical notation. A medical image is defined as a finite two-dimensional (2D) or three-dimensional (3D) array of digital numbers that represents a part of a human body in order to be used for clinical purposes. Each array location denotes a spatial location and includes a real valued sample, denoted the intensity, that is sampled at regular intervals. The intensity values of the sampled signal are quantized to a finite number of levels. Each element of the array is referred to as a pixel.

2.4 3D Computed Tomography Image

Computed Tomography (CT) images encode the amount that a tissue absorbs X-ray radiation as an intensity within a 2D or 3D image. The intensity is proportional to the amount of X-ray radiation absorbed by the tissue, referred to as radiographic attenuation. The radiographic attenuation a tissue depends on its density and composition. The spatial position of each image pixel encode the spatial positions of different tissues. Medical image intensities are often expressed in Hounsfield Units (HU). The Hounsfield unit is a linear transformation of a radiographic attenuation coefficient such that, in the transformed space, the intensity of distilled water at Standard Pressure and Temperature (SPT) is 0 HU and the intensity of air at SPT is -1000 HU. In CT images, the intensity of soft tissues in the vicinity of bones are represented by values ranging from -100 to +100 HU, whereas cortical bones have a

range from 500 to 2000 HU [67].

2.5 Image Segmentation

Segmentation describes the process of partitioning a collection of data into distinct groups or segments. Segmentation algorithms for image data separate images into groups of pixels. Typically these groups are intended to be semantically meaningful or homogeneous with respect to one or more features or characteristics. In practice, a segmentation algorithm maps each data point to a segment label where each segment has a unique label. Let $D = \{d_1, d_2, ..., d_N\}$ represent the image data and d_i represent the i^{th} point of image data. Also, let $L = \{l_1, l_2, ..., l_K\}$ be the collection of K labels associated with the segments within the image. Using this notation, the segmentation operation may be formulated as:

$$Label(D) = Y$$

where *Label* is the segmentation function and Y is the collection of labels, one for each point of the image data, i.e., $Y = \{y_1, y_2, ..., y_N\}$ such that $y_i \in L$ is a label fro d_i. A segmentation function forms the segments by assigning a label to each data point, i.e., $Label(d_i) = y_i$. For each label l_j, the collection of points that share the same label forms the segment s_j as expressed in equation (1).

$$s_j = \{\bigcup_i d_i |\, y_i = l_j\} \tag{1}$$

Each data point can have only one label, hence, segments are disjoint sets, i.e., $s_i \cap_{i \neq j} s_j = \{\emptyset\}$, and each segment, s_i, is formed by a set of data points that are con-

nected to each other by sequences of links that connect points. These sequences are referred to as paths. Two points that are connected by a path of length 1, i.e., a single link, are called adjacent. Adjacent points that are connected horizontally and vertically are referred to as 4-connected, while adjacent points that are connected horizontally, vertically, and diagonally are referred to as 8-connected. 4-connectivity and 8-connectivity are commonly used in segmentation algorithms to generate segments of connected points.

2.6 Dynamics Simulation

Dynamics simulation refers to the computational framework that seeks to model the interaction of physical objects in a virtual space. These physical interactions are modeled computationally using Newton's laws of dynamics in addition to other laws of physics such as Hooke's law. Dynamics simulation programs require a collection of models that virtually represent objects as input. These models must be given physical attributes such as mass and rigidity that allow them to be treated as physical entity. These models can be animated given paths overtime. A dynamics simulation program takes all these attributed models and paths as constraints then simulate the physical interaction of objects for a specified time interval. A dynamics simulation program provides as output all the geometry and attribute information of models as it varies overtime. There are three main challenges to perform a dynamics simulation: (1) getting good models of physical objects, (2) getting physically meaningful attributes for models, and (3) configuring correctly virtual internals of a dynamics simulation program. There are many programs are used to perform dynamics simulations. When

researchers perform dynamics simulations, they use programs such as Bullet engine, PhysX, and AGX Multiphysics to accomplish these simulations. In this dissertation, Bullet engine is used to perform dynamics simulations.

Dynamics simulation is important in scientific modeling of natural processes to gain insight into their functionality. Computational simulation has particular import when the real processes is not accessible or dangerous to reproduce in real life. Simulations are used to approximate the real effects of alternative conditions and courses of action in a processes. For example, in biomechanics field, dynamics simulation is used to study the human anatomical structures in order to assist physicians in planning the medical treatment [24]. For this simulation, a special mechanical model is created from combinations of rigid and deformable bodies, joints, constraints, and various force actuators. By changing variables in the simulation, physicians are able to make predictions about the behavior of the system under study.

2.7 Summary

In this chapter, a brief overview is provided for the main concepts and theories that are used in this dissertation. The overview presents the structure of bones, the concept of medical images, the formation of CT images, image segmentation, and dynamics simulation. Understanding these topics helps analyzing the contents of CT images and understanding the nature of the problem of computing the inverse mechanics for highly comminuted bone fractures.

CHAPTER 3: LITERATURE REVIEW

This literature review offer insights into computational bone fracture analysis in section (3) and the different image segmentation algorithms in section (3.1.2). The first section discusses the existing literature on the topic of dynamics simulation for a fracture analysis. This review is very small because there are not many of fracture analysis systems. The second section discusses the existing literature on the topic of image segmentation algorithms. Segmentation is a classical problem and has many various algorithms. The section divides these algorithms into categories and presents a literature review for each category.

3.1 Computer Based Bone Fracture Analysis

There are four academic and three commercial systems have been developed in the past few years to analyze bone fractures from 3D CT images of broken limbs. Such systems seek to give orthopedic physicians tools that help them make better treatment decisions. All of these systems include 2D and 3D visualizations of objects.

3.1.1 Academic Systems

The four most relevant academic systems are found in recent papers [23, 22, 90, 46]. These systems seek to provide a virtual reconstruction of a bone from its fragments. Physicians can use these systems to assist in planning for surgical reconstructions.

In [23], Cimerman and Kristan develop a virtual preoperative planning system

for displaced pelvic acetabular fractures where the acetabulum bone is broken. The system is used to assist orthopedic surgeons in planning reduction, i.e., a medical procedure to restore a fracture, for pelvic and acetabular structures. The system consists of tools that allow users to virtually perform a surgical simulation. In this system, CT images are manually segmented by an expert to extract bone fragments. Then, in order to reconstruct the bone, i.e., restore its original shape, surgeons need to move and rotate the segmented fragments manually to piece them back together. The process of piecing bone fragments back together is very time consuming and tedious. Surgeons need to make small refinements to the positions of bone fragments in order to obtain an acceptable result. Furthermore the results of virtual reduction are not repeatable since each fragment must be manually positioned.

In [22], Chowdhury et al. develop a system that automates the reconstruction process. This system is specifically designed to help physicians in virtually reconstructing craniofacial multifractures. In this type of fractures, the bone fragments of the mandible, one of the craniofacial bone, are usually displaced with clearly defined bone borders and non-deformed fracture surfaces. Bone fragments are extracted from a 3D CT image of the broken limb automatically using an intensity thresholding method, described in section (3.2). The reconstruction process is implemented as an automatic. The first stage approximately apposes the fracture surfaces using a graph matching algorithm, described in section (3.2.3.2). While, the second stage reconstructs the fracture using an algorithm that minimizes the difference between fracture surface points and surface points for a reference bone model. This system is accurate in reconstructing the mandible from several fragments. However, the system

assumes that a fragment is only aligned to a single other fragment. This assumption is not valid for highly comminuted fractures because the surfaces of a fragment are very likely align to surfaces on more than one other fragments. Furthermore, the algorithm of the second stage because it assumes that fracture surfaces are not plastically deformed has limited application. This assumption does not generalize to bone fractures in anatomic sites other than the mandible.

In [90, 3], Willis et. al. develop a semi-automatic system to virtually reconstruct highly comminuted bone fractures. The system can help orthopedic physicians to plan medical treatments for these bone fractures. The system provides a virtual interactive environment where users control the reconstruction process. The reconstruction process is accomplished by emphasizing geometric surface variations (ridges and valleys) during the alignment to more heavily influence the final reconstruction solution, generating more visually encouraging results. However, the algorithm is only tested on surfaces for simulated bone pieces fabricated from a specialty high-density polymeric foam. These surfaces have less variation and are easier to match than fracture surfaces for fragments extracted from real-world clinical CT images.

In [46], Liu develops a computational system to assist physicians in making a more accurate fracture severity classification. The system allows users to view and edit their fracture cases in both 2D via CT images and 3D via a virtual 3D environment. In order to do so, the system interface is integrated with a combination of 2D/3D image processing and surface processing algorithms. This integration allows users to quantitatively evaluate the fracture severity of bone fracture cases from their raw CT image data. Furthermore, the system provides tools that link the 2D CT imagery with

3D visualizations of the bone surfaces. The surfaces of bone fragments are extracted automatically from CT image data using the watershed segmentation algorithm to segment 3D CT images, described in section (3.2.3.1).

3.1.2 Commercial Systems

There are three commercial orthopedic computational systems that are capable of analyzing bone fractures: (1) OrthoView (Orthopedic Digital Planning, USA), (2) TraumaCad (VoyantHealth,USA), and (3) Mimics software system (Materialise, Belgium).

OrthoView and TraumaCad are two software systems for digital medical image analysis. These systems are designed mainly for orthopedic applications. The systems allow users to perform virtual preoperative planning sessions on-screen with 2D digital images. The user interface of these systems has unique features that allow users to edit, mark, and measure X-ray images. These features include:

1. Making measurements of various anatomic features from digital images,
2. Visualizing fracture patterns in 2D X-ray images,
3. Designing custom implants in 2D to assist physicians during surgeries,
4. Generating pre-surgical reports that specify medical treatment plans for fracture cases,
5. Reconstructing virtually simple fracture cases from 2D images,

These systems provide a set of pre-defined planning modules to help orthopedic surgeons in their preoperative planning sessions. The set of pre-defined planning modules

in OrthoView includes five modules for: (1) joint replacement, (2) limb deformity correction, (3) pediatrics, (4) fracture management, and (5) spine. While TraumaCad's set include nine modules for: (1) adult hip joint replacement, (2) pediatric hip joint, (3) deformity correction, (4) spine, (5) foot and ankle, (6) adult knee joint replacement, (7) upper limbs, (8) trauma, and (9) 3D implant visualization. These systems support only 2D images and do not support viewing 3D objects. Therefore, reconstruction solutions are not available in these systems.

The Mimics software system allows users to process and edit 2D image data, as well as, to construct 3D models for fracture cases. Mimics system provides image segmentation tools that allow users to segment 2D CT images manually. This system also allows users to make measurements directly on 3D models. Mimics system supports a wide range of output formats to export generated 3D data so that downstream applications, e.g., implant design software and surgical simulation software, are able to use the generated data as input. Mimics system attempts to bridge the gap between 2D image data and downstream 3D applications. However, while the system is able to generate 3D models, no further efforts are made to reconstruct these 3D models automatically or semi-automatically.

To conclude, there are no existing computational systems that estimate what broke the limb and how the bone fragments moved over time. While existing systems are present for analysis of bone fractures and for virtual reconstruction of bone fractures, these systems fall short in that they do not represent the actual breaking process. This dissertation presents a novel addition to this body of research that is unexplored to date.

3.2 Image Segmentation Methods

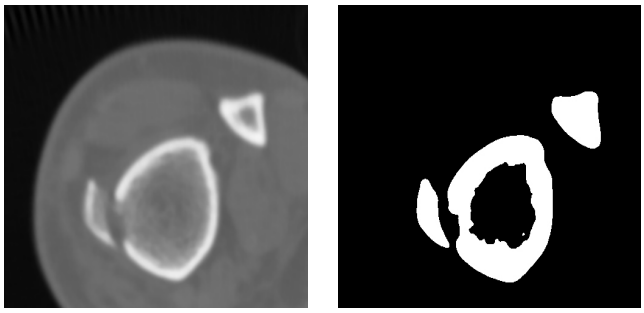
There are various methods proposed for segmenting CT images. Each algorithm has its own merits, limitations, and its field of applications. Segmentation algorithms are classified into the following four categories: thresholding [73, 40, 37, 58], edge based [31, 16, 13, 93, 9, 86, 18], region based [61, 8, 82, 92, 15, 76, 45], and graph models [92, 15, 75, 14, 44, 78, 96, 36, 35, 77, 76]. A description of each category is provided in the following sections with emphasis on the algorithms that are used in segmenting bone fragments within CT images.

3.2.1 Thresholding

Thresholding is a segmentation approach that creates binary images. Thresholding is important to separate image foreground from image background. Thresholding groups image points into segments according to their intensity values such that all points whose intensity values are in a given range fall into the same segment. Thresholding creates a binary image by labeling all points whose intensity values above a certain value, called threshold, with one and all pixels below that threshold with zero, i.e.,:

$$Label(\mathbf{x}) = \begin{cases} 1, & if\ I(\mathbf{x}) > T \\ 0, & otherwise \end{cases} \tag{2}$$

where I is the image, $\mathbf{x}$ is a point in the image, T is the threshold value, and $Label(\mathbf{x})$ is the label at point $\mathbf{x}$. Label 1 indicates a foreground point and label 0 indicates a background point.



(a) (b)

Figure 2: Thresholding of image with manual threshold value, (a) input image, (b) result of thresholding at $T = 127$. White area indicates label 1 the foreground and black area indicates label 0 the background.

Threshold value can be determined either manually or automatically. Threshold is determined manually by an expert who tries a range of values of T to find which one works best at identifying the objects of interests, see figure (2) . Manual determination of threshold value is a subjective process since different experts may produce different values because of the divergence of their opinions and the involvement of their personal work which reduces the reproducibility and comparability of the processed images. Automatic determination of threshold value is preferable. A simple method to compute automatically the threshold value is to use the gray level distribution characterized by the image histogram. Local maxima of the histogram correspond to the objects of interest, therefore, the threshold values are placed at the local minima of the histogram to separate the objects. The method is effective in cases where background and objects points intensities have some average values and the actual point intensities have some variation around these average values. The method may have difficulty in finding the threshold in cases where the image histogram does not have clearly defined local minima points which may lead to unacceptable segmentation

results.

In [79], Sonka provides a method to compute the optimal threshold value iteratively. The method computes the optimal threshold as follows. Let T^i be the threshold value at step i. In order to compute new threshold value, the image is segmented to background and foreground according to equation (2). Let H_0^i and H_1^i be the average intensities for image background points and foreground points, respectively. The new threshold value for step $i+1$ is computed as follows:

$$T^{i+1} = \frac{H_0^i + H_1^i}{2} \tag{3}$$

where T^{i+1} is the threshold value for step $i+1$. The iterative update for threshold value is repeated until there is no change in the threshold value, i.e., $T^{i+1} = T^i$. The iterative method to compute optimal threshold works well if the variances of background and foreground intensity distributions are approximately equal. Otherwise, the method generates usually non desired segmentation results because it does not handle largely different variances of background and foreground intensity distributions.

In [58], Otsu introduces a new criterion to compute threshold value in order to minimize the error of missclassification of points as background and foreground. Otsu's method intends to find a threshold that classifies the image points into two clusters which are: foreground and background such that the area under histogram for objects of one cluster that lies on the other cluster side is minimum. Otsu's method minimizes the within class variance which is equivalent to maximizing the between-

class variance. For threshold value T, the between-class variance, σ_b^2, is computed as follows:

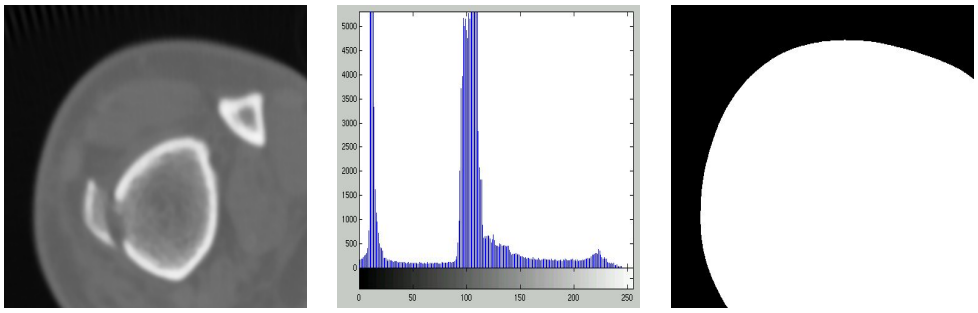
$$\sigma_b^2(T) = n_0(T)n_1(T)\left[H_0(T) - H_1(T)\right]^2 \tag{4}$$

where $\sigma_b^2(T)$ is between-class variance for threshold T, $n_0(T)$ and $n_1(T)$ are the number of background and foreground points for threshold T, respectively. $H_0(T)$ and $H_1(T)$ are the average intensities for image background and foreground points for threshold T, respectively. Otsu's method chooses the optimal threshold, $T_{optimal}$, such that the between-class variance σ_b^2 is minimum, i.e.,:

$$T_{optimal} = \underset{T}{\arg\min}\left\{\sigma_b^2(T)\right\} \tag{5}$$

where $\sigma_b^2(T)$ is between-class variance for threshold T. Otsu's method tries all possible threshold values to find $T_{optimal}$. Otsu's method requires the image to have bimodal intensity distribution. Otsu's method cannot segment the image correctly when the image does not have bimodal intensity distribution, e.g., medical images of anatomical structures, see figure (3) .

In [32], adaptive thresholding technique is provided in order to binary segment images with uneven illumination. Adaptive thresholding technique subdivides the image into multiple sub-images and apply different thresholds on the sub-images, i.e.,:



(a) (b) (c)

Figure 3: Thresholding of non bimodal intensity distribution image using Otsu's method, (a) original image, (b) intensity histogram of the original image, (c) results of Otsu's thresholding. White area indicates label 1 the foreground and black area indicates label 0 the background. The image details are lost after the threshold.

$$Label(\mathbf{x}) = \begin{cases} 1, & if\, I(\mathbf{x}) > T(\mathbf{x}) \\ \\ 0, & otherwise \end{cases} \tag{6}$$

where $T(\mathbf{x})$ is the threshold value at point $\mathbf{x}$. Adaptive thresholding has two problems which are: how to subdivide the image and how to specify the threshold for each sub-image.

Thresholding techniques consider the intensities of the points only without considering any contextual relationship between them. Thus, there is no guarantee that the produced segments are contiguous and makes the segmentation process sensitive to noise and image inhomogeneities. Noise corrupts the image histogram making the separation between different classes more difficult. When the image is noisy, the image is usually smoothed before applying a thresholding segmentation process. A survey of thresholding techniques is provided in [69] and a quantitative evaluation of different thresholding performances is provided in [73].

3.2.2 Boundary Based

Boundary based segmentation are algorithms that find the object boundaries and segment regions enclosed by the boundaries. Boundary based algorithms are divided into two sub categories: Edge based and deformable models which are discussed in the following two sub sections.

3.2.2.1 Edge Based

Edge based segmentation are techniques that locate boundary points of the objects using image gradient which has high values at objects edges in order to segment objects. An object boundary or edge in an image is defined by the local gradient of point intensity. Gradient is the first order derivative of the image function. Let $I(x, y)$ be a 2D image function, then the magnitude of the gradient is calculated as:

$$||\nabla I|| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{7}$$

where $||\nabla I||$ is the magnitude of gradient, $\frac{\partial I}{\partial x}$ is the gradient along the x direction, and $\frac{\partial I}{\partial y}$ is the gradient along the y direction. The direction of gradient is computed as follows:

$$\theta = \tan^{-1}\left(\frac{\frac{\partial I}{\partial y}}{\frac{\partial I}{\partial x}}\right) \tag{8}$$

where θ is the gradient direction. The discrete nature of digital images does not allow the application of continuous differentiation, so, the gradient calculation is performed by differencing [32].

The magnitude of the gradient can be displayed as an image. The magnitude image

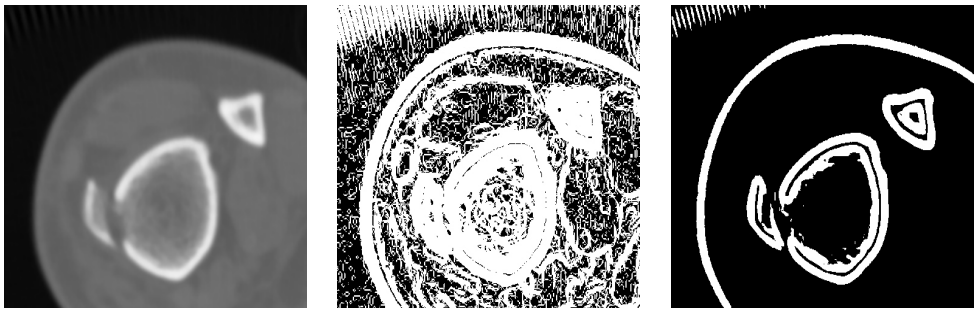
has gray levels which are proportional to the magnitude of local intensity changes. Gradient operators for digital images involve calculation of convolution by performing a weighted summation of the point intensities in local neighborhood. The weights are listed as a numerical array, known as kernel, with a form corresponds to the local neighborhood in the image. For example, Sobel edge detector uses two 3×3 kernels, which are:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

where G_x and G_y are approximations of the first derivative along the x and y directions respectively, i.e., $G_x \approx \frac{\partial I}{\partial x}$, $G_y \approx \frac{\partial I}{\partial y}$. The magnitude image of the gradient is generated by combining G_x and G_y using equation (7). The result of the edge detector depends on the used gradient kernel. Other edge detector kernels are Roberts, Robinson, and Fri-Chen [32, 38, 68]. Many edge detection methods perform a threshold operation on the gradient image after applying the gradient operator in order to decide whether an edge is found [32, 68], see figure (4) . The result of such methods is a binary image which indicates the edges points. The main challenging problem for these methods is to find the appropriate threshold value to enclose the objects of interest.

Edge based methods are fast and do not require prior information about the ob-



(a) (b) (c)

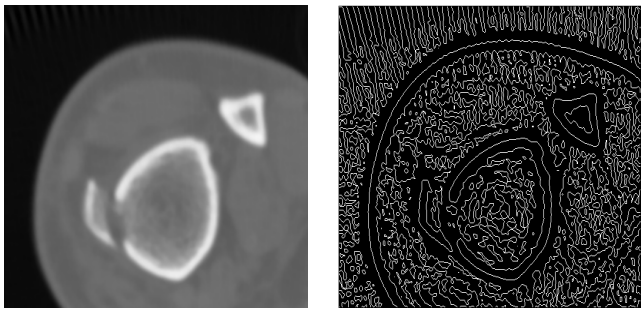
Figure 4: Edge segmentation algorithms, (a) input image, (b) result of Sobel edge detector with threshold 0, (c) result of Sobel edge detector with threshold 5,. White area indicates label 1 the edge and black area indicates label 0 the non edge.

jects in the image. The main problem of edge based segmentation methods is that the objects are not enclosed completely by edges. In order to form closed edges surrounding the objects of interest, a postprocessing step is needed to link or group edges that correspond to a single boundary. A simple method to link edges is examining the neighborhood points of an edge point to link points with similar edge magnitude and direction. Edge linking methods are expensive, computationally, and not reliable which need an expert to draw the edge when the edge tracing becomes ambiguous. In [85], Wang et al. introduces a hybrid algorithm that allows an expert to interact with the edge tracing process to correct errors using anatomical knowledge.

Laplacian operator is an approximation of the second order derivative and it is used to locate edges. First order derivative peaks are zeros in the second order derivatives. Laplace operator, ∇^2, of an image function $I(x, y)$ is defined as:

$$\nabla^2 I(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2} \tag{9}$$

where $\frac{\partial^2 I(x,y)}{\partial x^2}$ and $\frac{\partial^2 I(x,y)}{\partial y^2}$ are the second order derivatives of function $I(x, y)$ along



(a) (b)

Figure 5: An example for LoG edge segmentation algorithm, (a) input image, (b) result of LoG edge detector. The Log edge detector image is generated by setting Gaussian sigma to 2 and using a 13×13 convolution kernel. The white points indicates the edge locations.

the x and y directions, respectively. Laplacian operator for digital images is approximated by an $N \times N$ convolution kernel [79, 32], for example, a 3×3 kernel that approximates the Laplacian operator is:

$$\nabla^2 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Laplacian edge detector locate edges by the points where the Laplacian crosses zero. Laplacian edge detection methods are very sensitive to noise. The effect of noise can be reduced by smoothing the image before applying the edge detector. In [49], Marr and Hildreth applies a Gaussian smoothing filter on the image before applying the Laplacian, this operation is called Laplacian of Gaussian (LoG). The edges generated by LoG compared to Laplacian are smoother and the objects are better outlined, see figure (5) . In [12], Bomans et al. used LoG to segment MR images of the head. The main problem of LoG and Laplacian operations is the detection of non

significant edges in almost constant intensity regions. In [33], Goshtasby and Turner use a combination of zero crossing of LoG operator and local maximum of the gradient magnitude image followed by curve fitting algorithm to segment ventricular chambers in flow enhanced MR cardiac images.

In general, edge based segmentation algorithms are sensitive to noise and have a tendency to detect edges that are irrelevant to the real boundary of the object. Edges extracted by edge based segmentation algorithms are disjoint and do not completely represent the boundary of the object. Edges extracted by edge based algorithms need postprocessing to connect them and form closed object regions.

3.2.2.2 Deformable models

Deformable model based methods are segmentation approaches which evolve a predetermined curve or surface to its minimum energy according to external forces derived from the data set that pushes it toward the features of interest in the image and internal constrains that maintain splines structure shape and smoothness. There are two types of deformable models representations which are: parametric deformable models and geometric deformable models. Parametric deformable models represent curves and surfaces which are in their parametric forms explicitly during deformation. Geometric deformable models represent curves and surfaces implicitly as a level set of a higher dimensional scalar function.

In [41], Kass et al. introduce a parametric deformable model called snake. Snake is a controlled contiguous contour which can deform to match the desired shape under the influence of forces. The snake contour is represent as $\mathbf{v}(s) = \mathbf{x}(s)$ where $\mathbf{x}$ is the

coordinate function and $s \in [0, 1]$ is the parametric domain. The shape of the contour is dictated by the functional:

$$\mathcal{E}(\mathbf{v}) = \mathcal{S}(\mathbf{v}) + \mathcal{P}(\mathbf{v}) \tag{10}$$

The functional is viewed as a representation of the contour energy where $\mathcal{E}(\mathbf{v})$ is the total contour energy, $\mathcal{S}(\mathbf{v})$ is the internal deformation energy, and $\mathcal{P}(\mathbf{v})$ is the external potentials. The final shape of the contour minimizes the energy function $\mathcal{E}(\mathbf{v})$. $\mathcal{S}(\mathbf{v})$ is defined as:

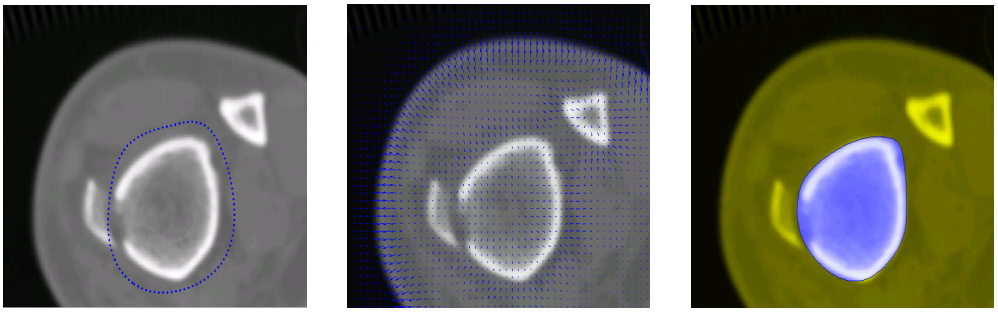
$$\mathcal{S}(\mathbf{v}) = \int_0^1 w_1(s) \left| \frac{\partial \mathbf{v}}{\partial s} \right|^2 + w_2(s) \left| \frac{\partial^2 \mathbf{v}}{\partial s^2} \right|^2 ds \tag{11}$$

where $w_1(s)$ is a function that controls the tension of the contour and $w_2(s)$ is a function controls the rigidity of the contour. $\mathcal{S}(\mathbf{v})$ characterizes the flexibility of the contour and its stretchiness. $\mathcal{P}(\mathbf{v})$ is defined as:

$$\mathcal{P}(\mathbf{v}) = \int_0^1 P(\mathbf{v}(s)) ds \tag{12}$$

where $P(\mathbf{v}(s))$ is a scalar potential function. $\mathcal{P}(\mathbf{v})$ couples the snake contour to the image. External potentials are designed such that their local minima coincide with the image features of interest, for example, edges and intensity maxima. The snake algorithm iteratively deforms the contour until it finds the one with the minimum total energy which hopeful corresponds to the best fit of the object contour in the image, see figure (6) .

In [2], Atkins and Mackiewich applied snake algorithm for brain segmentation. The

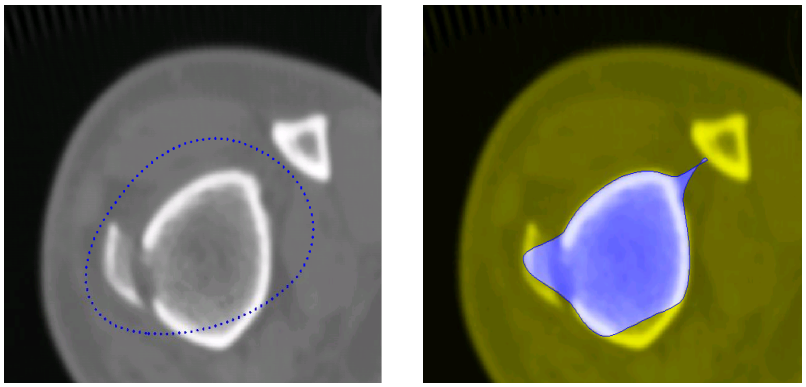


(a) (b) (c)

Figure 6: Snake algorithm segmentation of a bone fragment, (a) original image with the initial contour (blue curve), (b) The external force field, (c) segmentation result (blue curve). The snake algorithm iteratively deforms the contour until it finds the one with the minimum total energy which hopeful corresponds to the best fit of the object contour in the image.

input image is smoothed and a mask to determine the initial brain boundary contour is obtained by thresholding. Then, the snake model is performed to segment the brain.

Snake algorithm is a good approach to model shapes, detect edges, and trace motions because it forms a smooth contour that corresponds to the region of interest boundary. Snake algorithm is sensitive to the initial configuration of the snake contour that different initializations contour and parameters may produce different results, see figure (7) . Snake algorithm has a converging problem to concave and diffused boundaries parts of the region. Image force which is usually composed of image intensity gradient exists in a narrow region near the convex part of the object boundary. Diffused boundaries produces very weak image forces in their regions. Contour points that fall in a region without image forces or very weak ones cannot be pulled toward the object boundary causing the object region to merge with other regions which is known as the leak problem. In [94], Xue and Prince propose a snake algorithm with Gradient Vector Flow (GVF) in order to partially solve the converging problem of



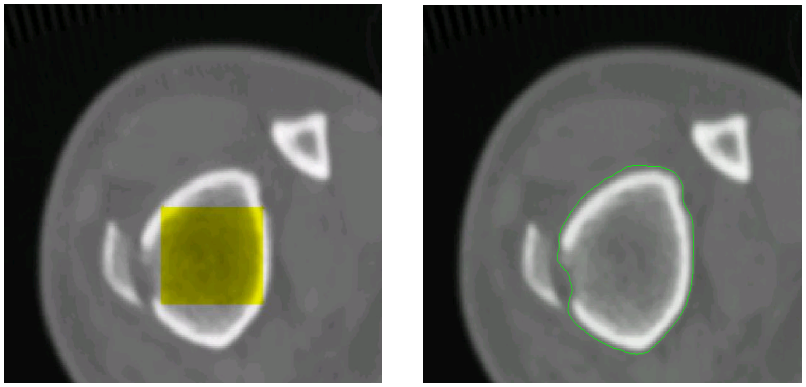
(a) (b)

Figure 7: Snake algorithm sensitivity to the initial configuration, (a) original image with the initial contour (blue curve), (b) segmentation result (blue curve). Different initializations contour and parameters for the snake algorithm produce different segmentation results compared to the ones used to produce figure (6).

snake contour in concave parts of the region. The problem is solved partially by pre-calculating the diffusion of the gradient vectors on the edge map. So, image forces can exists near concave region to pull snake contour points toward the target object boundary. GVF algorithm requires a good initialization for snake contour. GVF is sensitive to noise because it depends on edges in its calculations.

Snake based algorithms have a problem with handling evolving contours which require topological changes, e.g., two evolving contours merge into one. In this case, the algorithm needs to remove the connected contour points which are inside the merged region. This process is computationally expensive especially for higher dimensions, e.g., 3D.

In [72], Sethian introduce a level set approach in order to solve the problem of evolving contours which require topological changes. Level set represents the object boundary by a closed surface front $\delta(t)$ and propagates that surface along its normal with speed v, where v is a function of geometric and image characteristics. To



(a) (b)

Figure 8: Level set segmentation of a bone fragment image, (a) original image with the initial surface (yellow area), (b) segmentation result (green curve). Level set algorithm propagates the surface front along its normal until it converges to the structure boundary.

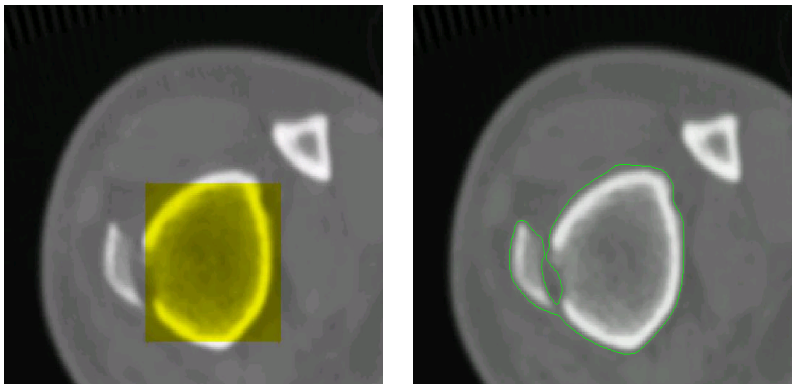
propagate the surface, level set embeds the surface function as the zero level set of a higher dimensional function Ψ which is defined as:

$$\Psi(\mathbf{x}, t) = d(\mathbf{x}) \tag{13}$$

where $d(\mathbf{x})$ is the signed distance function from point $\mathbf{x}$ to $\delta(t)$. The evolution of Ψ is given by:

$$\Psi_t + \upsilon \left|\nabla\Psi\right| = 0 \tag{14}$$

where ∇ is the gradient operation. υ is the speed along the surface normal and it can be a function of position $\mathbf{x}$, geometrical quantities, or physical quantities that are related to the problem under study. $\Psi(\mathbf{x}, t)$ remains as a function during its evolution and $\delta(t)$ may change topology, merge, and break as Ψ changes which give it the flexibility to converge to complex structures, see figure (8) . To compute the evolution of the interface $\delta(t)$, level set algorithm needs to update the level set values



(a) (b)

Figure 9: Level set segmentation sensitivity to initial configurations of the level set surface, (a) original image with the initial surface (yellow area), (b) segmentation result (green curve). The level set configurations are similar to the ones used to generate figure (8) except the initial surface. The level set algorithm wrongly merges the region of the bone fragment with other regions, i.e., leak problem.

near the zero level only. This method is called narrow band level set method [72].

Level set algorithm is sensitive to initial configurations of the level set surface, different configurations may produce different results, see figure (9) .Level set algorithm is sensitive to noise. In [27], Droske et al. introduce a level set method which incorporates curvature terms in the speed function in order to reduce the noise effect. Also, they used an adaptive grid to speed up the evolution process.

Deformable models are widely used in medical image segmentation applications specially snake model. In [98], Yue et al. applied snake algorithm to locate the rib borders in chest radiographs. At the beginning, the thoracic cage boundary is determined to restrict the search space. Then, Hough transform is used to find the approximate rib borders. Finally, the snake algorithm is applied to refine the rib borders. In [39], Jiang et al. used geodesic active contours in order to segment forearm bones. Geodesic active contours is introduced by Caselles et al. in [17].

Geodesic active contours is a snake model algorithm which evolves over time according to geometric measures, e.g., curvature. Jiang initializes the snake contour manually from the X-ray image of the patient at the first visit to the hospital. In [19], Chen and Jian use snake algorithm to extract teeth contours. The initial configuration of the snake contour is obtained by detecting the gap between lower and upper teeth and the gaps between neighboring teeth.

In [21], Chen et al. provide an incremental approach to segment femur bones. The initial configuration of the snake contour is obtained by detecting the main features of the femur bone in X-rays, e.g., parallel lines in the shaft area and circles in the femoral heads, and fitting a curve to match the detected features. Then, the snake algorithm is applied with curvature constraints to refine the femur contour. Chen's work uses a spring force that describes the difference between the actual curvature of the snake and the reference curvature of the model.

In [4], Ballerini and Bocchi apply a modified the snake algorithm to segment hand bones. Snake algorithm is modified by adding an internal energy term to model the spatial relationships between adjacent bones. The new internal energy term appears as an elastic force which connects appropriate points of adjacent snakes contours. The snake algorithm is performed using polar coordinates in order to introduce ordering to contour points and prevent snake elements to cross each other during the evolution. Snake evolution to minimize the contour energy is performed using genetic algorithm by encoding snake configuration into chromosomes. The algorithm has a problem in representing concave shapes and is sensitive to initial contour placement which is chosen randomly.

In summary, despite of the appealing results of the deformable algorithms in segmentation, deformable algorithms encounter several problems: (1) sensitive to the initial configurations which often imply impractical user interaction in order to obtain good ones, (2) poor to converge at weak and diffused boundaries because image forces are very weak in these regions and cannot pull the evolving curve or surface toward the object boundary (3) poor to converge at concave regions because of the contour smoothing terms which are used to produce smooth contours and the absence of image forces which are usually composed of image intensity gradient and exist near the convex part of the object boundary, (4) level set methods are computationally expensive due to their iterative optimization, (5) require users to set the values for many non easily understandable parameters that are used in the terms of the deformable models equations which make the algorithms unintuitive to users in clinical practice. For bone fragment segmentation problem, deformable methods tend to perform well in segmenting intact, i.e., unbroken, bones but have poor performance in segmenting fragments, i.e., fractured, bones because the bone tissues internal to bones are often very similar to the soft tissue surrounding the bone. When the outer (cortical) surface of the bone is fractured these internal structures are often adjacent to the surrounding soft tissues which makes it difficult to distinguish the bone fragment boundary in these areas.

3.2.3 Region Based

Region based algorithms are approaches which look for a group of points with similar features. Region based algorithms divide the image into disjoint regions where

each region R is a group of points that satisfy the homogeneity criteria, i.e.:

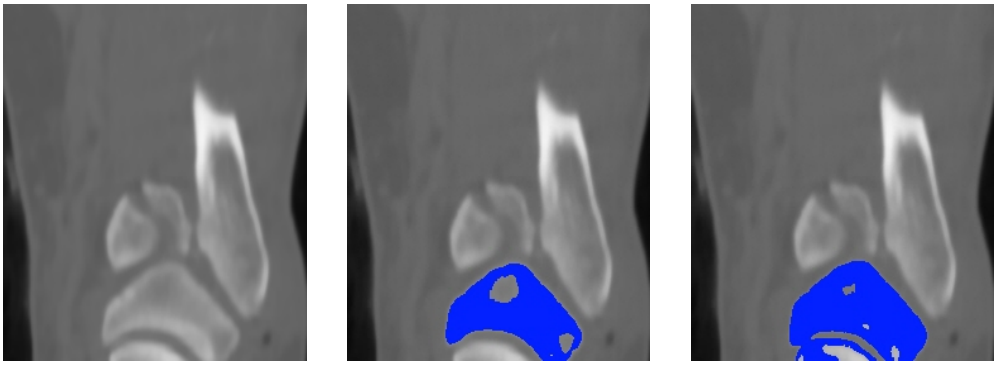
$$I = \bigcup_{i=1}^{K} R_i,\ R_i \bigcap R_j = \emptyset\, \forall i \neq j \tag{15}$$

where I is the image and K is the number of disjoint regions. Region based algorithms are divided into two sub categories: region growing and watershed which are discussed in the following two sub sections.

3.2.3.1 Region Growing

Region growing algorithm begins with a point or a group of points called seeds which belong to the objects of interest. Seeds can be chosen either manually by an expert or automatically by seed finding algorithm, e.g., converging square algorithm [57] which is applied in [1]. Converging square algorithm recursively divides an $n \times n$ image into four overlapping $(n-1) \times (n-1)$ sub images and the sub image with the maximum intensity is chosen. The procedure is repeated until four points of 2×2 square remain, then the point with the maximum intensity is chosen as the seed. For each seed point i a unique region R_i is created. Then, the neighboring points of the regions are examined one at a time and added to the growing region if they satisfy the homogeneity criterion. The process continues until no more points can be added. The objects then are represented by the points of grown regions such that each unique region represents a unique object [79, 32, 1, 80].

Region growing algorithms vary depending on the homogeneity criterion. The choice of the homogeneity criterion is usually based on the nature of the problem in the application. For example, a simple homogeneity criterion is comparing the



(a) (b) (c)

Figure 10: Seeded region growing segmentation, (a) original image, (b) a good bone fragment segmentation result, (c) a bad bone fragment segmentation results. The bone fragment region is incorrectly merged with another region. The homogeneity criterion is based comparing the difference between the point intensity value and the mean intensity value over a region to a predefined threshold value. The threshold value is set to 25.

difference between the point intensity value and the mean intensity value over a region to a predefined threshold value. If the difference is less than the threshold value, for example, one standard deviation of the intensity across the region, the point is added to the region, otherwise, it is defined as an edge point.

Region growing algorithms can correctly segment regions with the same properties and spatially separated. Region growing algorithms generate connected regions. The result of region growing algorithm is highly dependent on the choice of the homogeneity criterion. If the homogeneity criterion is not properly chosen, the regions may merge with regions that do not belong to the target object or may leak out into adjoining areas. Region growing algorithm also have a problem that different seeds may not generate identical regions results, see figure (10) .

In [66], Rosenfeld and Kak provided a region based technique called region splitting. Region splitting begins with an initial segmentation and subdivide the regions that

do not satisfy the homogeneity criterion. A simple region splitting technique is to begin by considering the whole image as a single region. Each region is tested for homogeneity criterion. If the region satisfies the homogeneity criterion, then the region corresponds to an area of an interest in the image. Otherwise, the region is divided into sub regions, for example, four sub regions, and tested for homogeneity criterion. The process is repeated until no more splitting occurs. Region splitting generates regions correspond to areas with similar properties but it may generate too many of them which lead to over segmentation problem.

In [5], Bankman et al. propose a region growing algorithm called hill climbing to segment microcalcifications in mammograms, i.e., tiny clusters of mineral deposits which can be scattered throughout the mammary gland. The algorithm is based on the fact that microcalcifications have closed contour edges around a known point which is the local intensity maximum $\mathbf{x}_0$. The algorithm proceeds as follows. For each point $\mathbf{x}$, a slope value $s(\mathbf{x})$ is defined as:

$$s(\mathbf{x}) = \frac{I(\mathbf{x}_0) - I(\mathbf{x})}{||\mathbf{x}_0 - \mathbf{x}||} \tag{16}$$

where I is the image and $||\mathbf{x}_0 - \mathbf{x}||$ is the Euclidean distance between the local maximum point $\mathbf{x}_0$ and point $\mathbf{x}$. The points of the object edges are detected by radial line search emanating from the local maximum. The line search is applied in 16 equally spaced directions originated from the local maximum point $\mathbf{x}_0$. For each direction, a point is identified as edge point if it provides the maximal slope value. Then, the identified edge points are used as seeds for region growing with a spatial

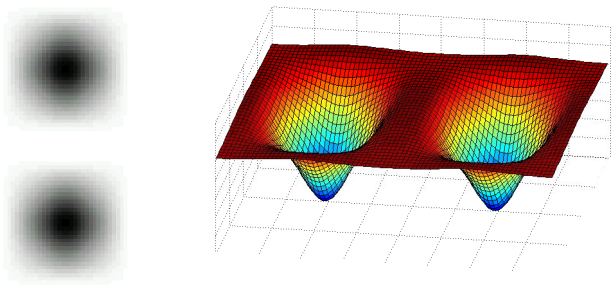
constraint that grows the region inward toward the local maximum and an intensity constraint to include points with intensity values increasing monotonically toward the local maximum. Hill climbing algorithm does not need selection of a threshold value because it grows the region from the edges toward the local maximum point.

In [48], Manos et al. propose a region growing algorithm to segment hand and wrist bones. The algorithm begins with a region growing stage that will produce an oversegmented result. Then, a region merging stage is applied to combine similar regions according to size, connectivity, homogeneity, and edge information. The final segmentation result is obtained by applying a region labeling process based on heuristic rules according to intensity information.

Region growing algorithms are fast but need the objects of interest to have homogenous features. Region growing algorithms are highly dependent on the used homogeneity criterion and initial seeds. Improper choice of the homogeneity criterion or seeds selection may generate undesirable result. Region growing algorithms are sensitive to noise and may generate oversegmentation results, therefore, a post processing step is needed to merge similar regions.

3.2.3.2 Watershed

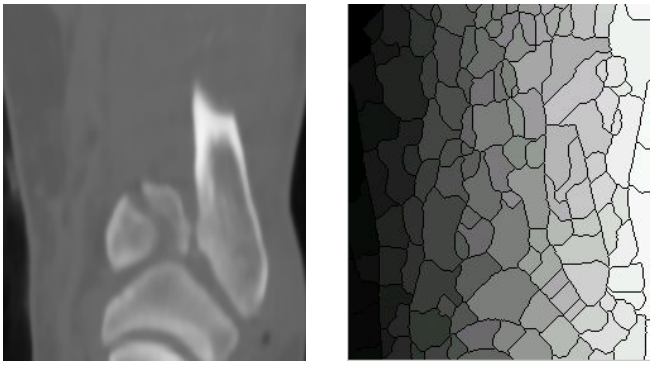
Watershed segmentation is a region based approach comes from the field of mathematical morphology. Watershed algorithm requires a set of connected points called marker inside each object of interest including the background as a separate object. Markers can be chosen either manually by an expert or automatically depending on the application. The markers are grown using a morphological watershed transfor-



(a) (b)

Figure 11: Watershed model, (a) input image with two dark blobs, (b) topographical representation. The bright area represents "high" surface level and dark area represents "low " surface level, the catchment basins in (b) corresponds to the dark regions in the input image.

mation to form the regions [10]. The idea of watershed transform is based on an analogy for how rain water flows when it falls upon a topographic relief. Under these circumstances, water drops follow the landscape downhill forming lakes at the bottom of the valleys. The lowest point in each valley is called a Catchment Basin (CB). As water continues to fall, the catchment basins collect water that accumulates into lakes. Neighboring lakes eventually merge when the water level exceeds the altitude of the ridges that separate the two lakes. When this occurs, dams are built at locations where water coming from different lakes meet to keep them separate. As a result, the landscape is divided into regions and the union of dams form the watershed lines or watershed. This analogy is extended to images by taking the image point intensity values as the altitudes of a topographic relief where high intensities (bright areas) represent high altitudes and low intensities (dark areas) represent low altitudes, see figure (11). A detailed description of watershed transform is provided in section (7.1.1). Watershed algorithm is simple and intuitive. Watershed algorithm has the problem of oversegmentation because each local minimum will form its own



(a) (b)

Figure 12: Sample result of watershed segmentation, (a) original image, (b) watershed segmentation result. The oversegmentation problem is clearly visible due to noise which introduce local minima that are not indicative of semantic contents.

catchment basin, see figure (12) .

In [56], Najman and Schmitt propose to use morphological operations to reduce oversegmentation. In [65], Rogowska et al. provide a watershed implementation which uses Sobel edge operator in place of the morphological gradient to extract edge strength. The algorithm is applied to extract lymph nodes. At the beginning, the expert selects a point inside the node to mark the node and draw a circle around the node to mark the exterior of lymph node as background. Then, an edge image is created by using Sobel edge detector to have high values for points with strong edges. A watershed operation called simulated immersion is performed on the edge image using node and background markers to segment the lymph node from the surrounding tissue. The simulated immersion watershed examines whether a drop at each point in the edge image would flow toward the interior node marker or the background marker. Points that flow to the node marker belong to the lymph node, whereas points that flow to the background marker belong to the background. The watershed algorithm is sensitive to noise. In some applications the noise effect can be reduced by applying

smooth filters before processing the image.

In [34], Grau et al. modify the watershed algorithm to utilize a set of probability functions that encode prior information about the objects to reduce the sensitivity to the noise. The probability function encodes the stochastic dependence between the intensity of two neighboring pixels and the boundary of the unknown object. This work is referred to as the "improved watershed." The main shortcoming of this algorithm is its dependency on a prior information of the objects. Grau states that the success of the algorithm largely relies on the existence of a prior distribution about the objects of interest. The absence of prior information causes the algorithm to be sensitive to noise and not able to segment areas with low contrast boundaries, and this is not desirable. A detailed description of the improved watershed algorithm is provided in section (7.1.3). In [88], Wegner et al. introduce a watershed algorithm that performs the watershed transform twice. The second watershed transform in applied on the mosaic image generated by the first watershed transform to reduce oversegmentation.

Watershed algorithm has many properties that makes it useful for many segmentation problems. It is simple and generates contiguous regions. Watershed algorithm may produce oversegmentation result which requires a post processing step to merge similar regions. Watershed algorithm is sensitive to noise and poor at detecting thin structures [34].

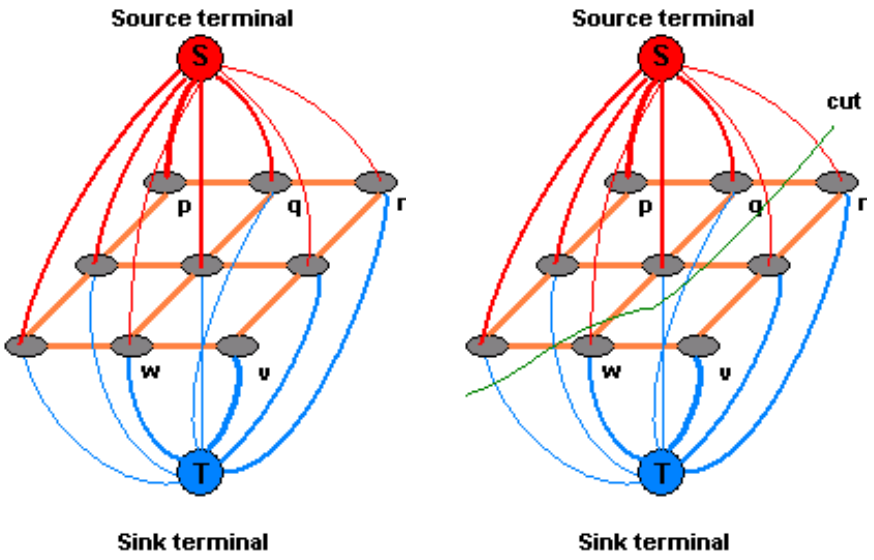
3.2.4 Graph models

Graph based methods usually represent the segmentation problem in terms of a graph $G(V, E)$ where V is the set of vertices corresponds to image elements such that each node $v_i \in V$ corresponds to a point in the image and E is the set of edges connecting pairs of neighboring vertices. Each edge in the graph is assigned a weight that is based on the properties of the points that it connects for example their image intensities. A graph can be divided into multiple partitions by removing the edges between them, the edge removal is called cut. Graph based methods try to divide the graph G into two or more partitions by finding the minimum cuts in a graph where the cut criterion is designed to minimize the similarity between the points that are being split. The cost of the cut between two partitions is defined as:

$$cut(A_i, A_j) = \sum_{u \in A_i, v \in A_j} w(u, v) \tag{17}$$

where A_i and A_j are two unique partitions in the graph and $w(u, v)$ is the edge weight between u and v, see figure ((13)) .

In [92], Wu and Leahy present an algorithm that divides a graph into K partitions such that the maximum cut between them is minimized. Wu's algorithm tends to cut the graph into small partitions because the cut value in equation (17) is proportional to the size of the partition. In order to avoid this problem, Shi and Malik proposed a new algorithm called normalized cut in [75]. Normalized cut algorithm defines a new cost function called $Ncut$. $Ncut$ is defined as:



(a) (b)

Figure 13: Simple weighted graph that shows the vertices and edges, (a) graph G, (b) a cut in graph G. Edges weights are reflected by thickness.

$$Ncut(A_i, A_j) = \frac{cut(A_i, A_j)}{assoc(A_i, V)} + \frac{cut(A_i, A_j)}{assoc(A_j, V)} \tag{18}$$

where $assoc(X, V) = \sum_{u \in X, t \in V} w(u, t)$ is the total connection from nodes in X to all nodes in the graph. Normalized cut algorithm measures the total similarity within the partitions and the total dissimilarity between different ones. Normalized cut algorithm has the problem of dividing high variability regions into multiple ones because of its dependance on cutting large weight edges. Wang and Siskind addressed this issue in their work in [87]. They present a new algorithm called ratio cut which minimizes the ratio of the corresponding sums of two different weights of edges along the cut boundary. The new cost function is called *Rcut* and it is defined as:

$$Rcut(A_i, A_j) = \frac{c_1(A_i, A_j)}{c_2(A_i, A_j)} \tag{19}$$

where $c_1(A_i, A_j) = \sum_{u \in A_i, t \in A_j} w(u,t)$ is the first boundary cost which measures the homogeneity of A_i and A_j. $c_2(A_i, A_j)$ is the second boundary cost which measures the number of edges between A_i and A_j.

In [14], Boykov and Jolly use graph cut to interactively segment bones from abdominal CT images. The proposed technique uses the graph cut to find the global optimal solution of the segmentation problem for the image. The segmentation process is controlled by hard and soft constrains. The hard constraints are obtained from manually marked points on object regions and background by the user at the beginning of segmentation process while the soft constraints are obtained from the boundary and region information. The process starts by "clicks" and "strokes" on the object and background then the graph cut is used to find the segmentation solution that has the best balance of boundary and region properties among all solutions satisfying the constraints.

Graph based algorithms are computationally expensive because they try to find the global optimum solution. Also, they suffer from over-segmentation problem because they depend on low level features such as intensity and edges which are usually affected by noise.

3.2.5 Summary

Image segmentation algorithms are evaluated according to the considered information, computational complexity, performance, sensitivity to noise, and manual initialization. Thresholding algorithms consider the intensities of the points only without considering any contextual relationship between them. Thus, there is no guarantee

that the produced segments are contiguous and makes the segmentation process sensitive to noise and image inhomogeneities. Edge based segmentation algorithms locate boundary points of the objects using the image gradient. Edge based algorithms are sensitive to noise and have a tendency to detect edges that are disjoint and do not completely represent the boundary of the object. Deformable based segmentation algorithms evolve a predetermined curve or surface to its minimum energy according to external forces that pushes it toward the features of interest in the image and internal constrains that maintain splines structure shape and smoothness. Deformable based algorithms are sensitive to the initial configurations and have a problem of converging to concave and diffused boundaries parts of the region. Region based segmentation algorithms divide the image into disjoint regions where each region is a group of points with similar features. Region-based algorithms are sensitive to noise and require the objects of interest to have homogeneous features. Graph based algorithms represent the segmentation problem in terms of a graph $G(V, E)$ which its vertices correspond to the image elements and its edges connect the pairs of neighboring vertices. Each edge in the graph is assigned a weight. Graph based algorithms divide the graph into two or more partitions by finding the minimum cuts where the cut criterion is designed to minimize the similarity between the partitions. Graph based algorithms are computationally expensive and sensitive to noise. Thresholding, region-based and graph-based algorithms tend to have an over-segmentation problem. The computational complexities are roughly linear for thresholding, region-based and edge-based algorithms, while it is higher for deformable based and graph model algorithms. The algorithms of all categories are sensitive to noise but the deformable based algorithms

have the ability to include constraints that make them less sensitive to noise. Region based and deformable based algorithms generally depend on manual initialization.

Among the existing methods for segmenting images, those presented in [48, 98, 39, 21, 4, 14, 34] are the most related to the bone fragment segmentation problem. In [48], Manos et al. used a region growing algorithm to segment hand and wrist bones. The algorithm begins with a region growing stage that will produce an oversegmented result. Then, a region merging stage is applied to combine similar regions according to size, connectivity, homogeneity, and edge information. The final segmentation result is obtained by applying a region labeling process based on heuristic rules according to intensity information. In [98], Yue et al. applied the snake algorithm to locate the rib borders in chest radiographs. At the beginning, the thoracic cage boundary is determined to restrict the search space. Then, Hough transform is used to find the approximate rib borders. Finally, the snake algorithm is applied to refine the rib borders. In [39], Jiang et al. used geodesic active contours in order to segment forearm bones. Jiang initializes the snake contour manually from the X-ray image of the patient at the first visit to the hospital. In [21], Chen et al. provided an incremental approach to segment femur bones using the snake algorithm. The initial configuration of the snake contour is obtained by detecting the main features of the femur bone in X-rays and fitting a curve to match the detected features. Then, the snake algorithm is applied with curvature constraints to refine the femur contour. In [4], Ballerini and Bocchi apply a modified the snake algorithm to segment hand bones. Snake algorithm is modified by adding an internal energy term to model the spatial relationships between adjacent bones. The snake algorithm is performed us-

ing polar coordinates in order to introduce ordering to contour points and prevent snake elements to cross each other during the evolution. Snake evolution is performed using genetic algorithm by encoding snake configuration into chromosomes. The algorithm has a problem in representing concave shapes and is sensitive to initial contour placement which is chosen randomly. In [14], Boykov and Jolly used a graph cut approach to interactively segment bones from abdominal CT images. The algorithm uses the graph cut to find the global optimal solution of the segmentation problem for the image. The segmentation process is controlled by hard and soft constrains. The hard constraints are obtained from manually marked points on object regions and background by the user at the beginning of segmentation process while the soft constraints are obtained from the boundary and region information. The process starts by "clicks" and "strokes" on the object and background then the graph cut is used to find the segmentation solution that has the best balance of boundary and region properties among all solutions satisfying the constraints. In [99], Zhang et. al. propose a 3D adaptive thresholding method to segment bones within CT images. The proposed method classifies image pixels into two classes: bone and non-bone. Then, an iterative process of 3D correlation is performed to update the classification of pixels. A post-processing step of 3D region growing is performed to extract bone regions. Thresholding methods assume homogeneity of the object being segmented and high contrast between the object and background. This assumption is invalid in bone fragment segmentation problem because in fractured bone case the internal structures of the bone are usually adjacent to the surrounding soft tissues which are very similar to the internal bone tissues. In [97], Yousefi et. al. propose a method

to segment the radius bone in wrist within MR images using active contours. The proposed method computes an initial segmentation of an MR image. The method estimates a mask of the radius bone in the MR image using anatomical knowledge about that bone shape and size. The mask is applied to derive a convex hull region around the radius bone in the image. This region is used as an initial mask for active contour. The problem with this method is its dependence on a prior knowledge about the shape of bones within images. This is knowledge is not available for bone fracture fragments which have arbitrary shapes. In [91], Wu et. al. propose a method to segment pelvic bones from CT images. The proposed method computes edges in a CT image. The computed edges are best matched with predefined manually generated templates of pelvic bones. Then, the method applies a supervised learning technique that requires a set of labeled training images to segment bone fragments from the image. The main problem of their approach is the need for predefined templates of bone fragments. Bone fragments of a fracture case have arbitrary shapes so it is not possible to define templates for all these shapes. In [59], Paulano et. al. propose a region growing algorithm to segment bone fragment from CT images. The algorithm is a semi-automatic one and based on 2D seeded region growing. A user has to set a seed in the first slice in which a bone fragment appears. When multiple fragments are wrongly joined the user has to place additional seeds to separate them. The propagation of labels from the seeds is based on heuristic rules according to intensity information. This algorithm is highly dependent on the used homogeneity criterion and initial seeds. Improper choice of the homogeneity criterion or seeds selection may generate undesirable result and this not good for bone fragment segmentation since

bone is an inhomogeneous tissue. In [34], Grau et. al. used the improved watershed algorithm to segment knee cartilage from MR images. The algorithm combines the watershed transform and atlas registration through the use of markers. The algorithm applies anisotropic diffusion filter to reduce noise without compromising the contrast at significant edges. Then a user selects manually pixels as markers for each of three classes: cartilage, bone and other tissues. The last class represents the ligament and muscle around the cartilage. The algorithm then uses a knee atlas to complete the segmentation process by using the improved watershed algorithm. There two shortcomings of this algorithm in segmenting fractured bones: (1) its dependency on prior information and (2) wrong labeling of pixels in close proximity area. The prior information for bone fragments is not available due to the large variation and randomnicity in fragments characteristics such as shape, position, orientation, and texture. The algorithm may assign pixels with wrong labels for other nearby fragments, referred to as leak problem. This problem likely to appear at close proximity areas where small parts of different bone fragments with small depth appear touching each other in a CT image, see figure (14) . These close proximity areas are referred to as “isthmus” of bone area. An ”isthmus” bone area has a small depth so that it appears blurry in a low resolution CT image.

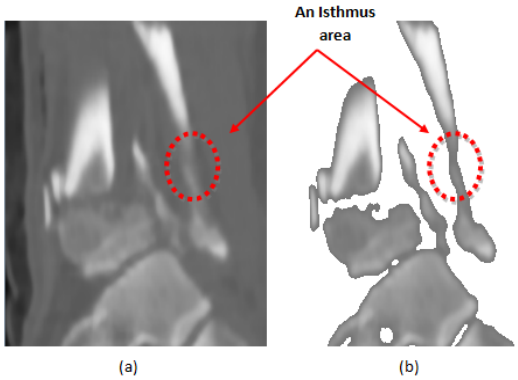


Figure 14: An example of an “isthmus” of bone area. (a) A CT image of a fractured bone. (b) An image of the estimated bone region. Circled area shows a location for an “isthmus” of bone area where two bone fragments are touching. An ”isthmus” area has a small width compared to other bone region areas.

CHAPTER 4: SYSTEM INTERFACE

This chapter describes the graphical user interface for the system that estimates the inverse mechanics for a fracture event. The system interface allows a user to control the system by specifying data and allows a user to analyze the results. In order to find the plausible solutions for a fracture event, the system requires the following datasets as input: (1) the surfaces of the fracture fragments, (2) the position and orientation of the fracture fragments at the start and end of a fracture event, (3) the model for the limb soft tissues at the start of a fracture event, (4) a 3D model for the strike object, (5) the physical attributes for these objects, and (6) the initial settings which control the search process to find the set of plausible solutions for a fracture event. The first data set is provided by applying the 3D segmentation described in chapter 7. The second data set is provided by applying the reconstruction system described in [46] which reconstructs fracture fragments using two 3D CT images of a fractured limb. Here, one image provides a model of the unbroken bone and a second records the fractured bone fragments after an injury. The third data set is estimated from the intact CT image by applying the 3D segmentation algorithm to extract soft tissues from that image as described in section (5.2). Data sets 4, 5, and 6 are unknown to the system and a user has to provide them to the system.

The system interface is divided into three components: (1) generating attributed models, (2) dynamics optimization, and (3) fracture simulation analysis. In the first

component, the system interface collects the required data from a user to generate the geometry for the attributed models and assign them physical attributes. These represent virtually the soft tissue, the bone fragments, and the strike object that are involved in the fracture event. In the second component, the system interface collects the data from a user to control the optimization process which seeks to find the plausible solutions for the unknown variables of the fracture event. In the third component, the system interface provides visualization and analysis tools that allow users to navigate the solutions computed by the system to aid in creating improved understanding and insights regarding how a fracture event occurred.

The system interface is implemented as a Java application that extends the work in [46] which is called FxRedux. FxRedux is a software system for reconstructing highly comminuted tibia fractures from 3D CT images. New interface elements added to the FxRedux system interface enable for this dissertation fracture inverse mechanics simulations visualization and analysis via the following three steps:

1. Generate attributed models,

2. Dynamics optimization,

3. Fracture simulation visualization and analysis.

Using these steps estimates of the fracture event can be generated and analyzed Discussion of the system interface starts with a brief overview of the FxRedux interface from [46] followed by three sections that describe the new interface components associated with the three steps above.

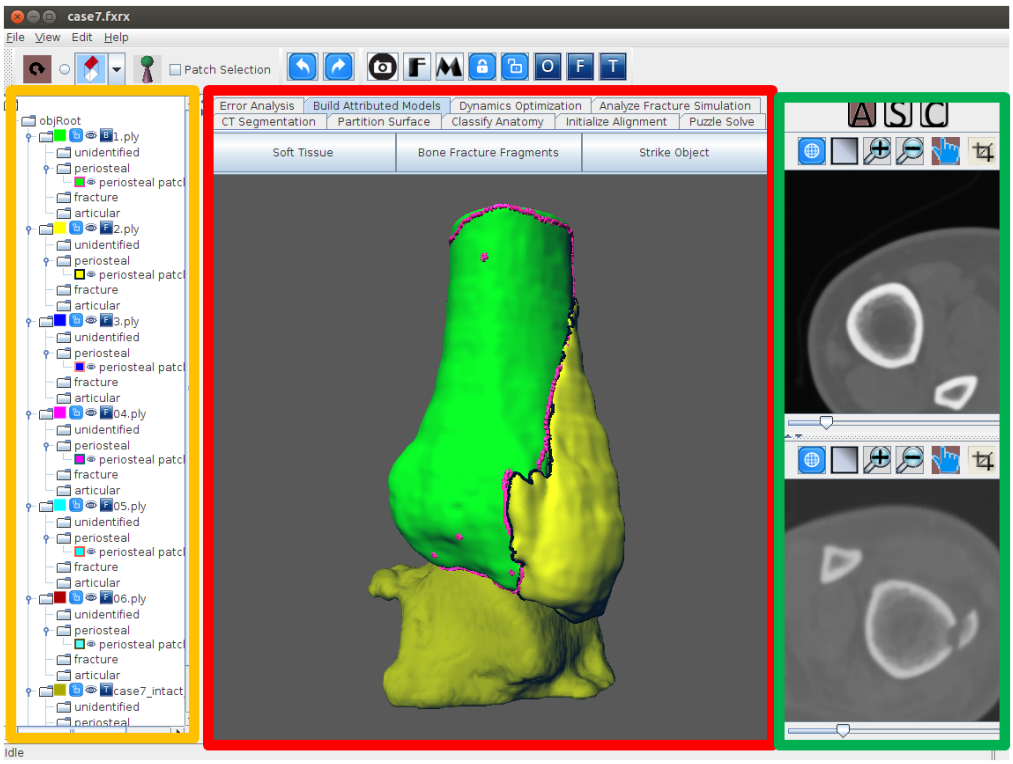


Figure 15: A screen capture of the system interface developed in this dissertation is shown. This image was captured after the reconstruction of fracture fragments and after the alignment of a talus surface. Tabs at the top of the window outlined with a red line access separate interface elements that reconstruct bone fragments from the new interface elements that estimate the inverse mechanics of fractures.

4.1 FxRedux Overview

FxRedux is a system that assists in reconstructing a highly comminuted tibia fracture from 3D CT images of a fracture case. FxRedux generates two datasets needed by the system for estimating fracture inverse mechanics: (1) a collection of 3D surfaces for bone fragments and (2) a collection of transformations that move bone fragment surfaces from their fracture positions to their positions in the reconstructed bone. FxRedux provides visualizations and an interface to control these algorithms. A detailed description of FxRedux is available in [46].

The FxRedux interface window consists of three main regions:

1. the 3D canvas, shown as the region outlined with a red line in figure (15),

2. the image panel, shown as the region outlined with a green line in figure (15),

3. the tree panel, shown as the region outlined with a yellow line in figure (15).

The user integrates with items in these regions to control the bone reconstruction process.

The 3D canvas provides a virtual environment where users view and manipulate 3D objects. Each 3D object is shown as 3D surface that is drawn onto the canvas. Using the tree panel and the 3D canvas, users view, hide, and transform 3D objects individually or as a group.

The image panel (on right of figure (15)) displays two CT image slices in two separate panels. The upper panel shows a CT image of an unbroken limb, while the lower panel shows a CT image of a fractured limb. Below each displayed image, a slider bar is provided to view different slices in the 3D CT image. Three different anatomic views are supported: axial, sagittal, and coronal.

The tree panel displays a tree representation of the objects shown in the 3D canvas. Users select objects in the tree panel to perform actions on specific objects or groups of objects shown in the 3D canvas. The tree panel allows users to perform a number of useful actions on the 3D objects such as select object, delete object, merge object, and split object.

Using FxRedux, virtual fracture reconstruction of bone fragments is accomplished through the following sequence of three interactive steps:

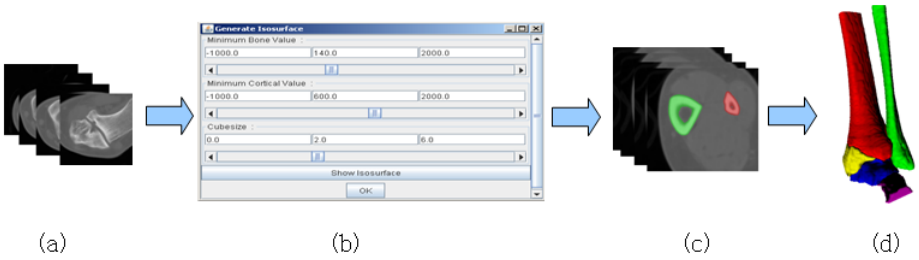


Figure 16: An illustration picture shows the interactions that take place to segment a 3D CT image in step one.(a) a 3D CT image, (b) users specified parameters for bone segmentation, (c) segmented image, and (d) generated 3D surfaces.

1. Segment 3D bone surfaces from intact and fracture CT images to generate a surface for the intact, i.e., unbroken, bone and a surface for each bone fragment,

2. Align intact tibia surface to surfaces of tibia bone fragments,

3. Reconstruct surfaces of tibia bone fragments,

By performing the listed steps, a virtual reconstruction of the unbroken bone can be approximated from the aligned bone fragments.

4.1.1 Segmentation of 3D Bone Surfaces

The segmentation process extracts the outer surfaces of the bone fragments from the 3D CT fracture image and to extract the outer surface for the unbroken bone from the 3D intact image. This step takes three data sets as inputs: (1) a 3D CT image of a broken limb, (2) a 3D CT image of an unbroken limb, and (3) a users input. The output of this step is a collection of 3D surfaces, one for each fragment, and a 3D surface for the unbroken bone. Figure (16(b)) show the three parameters needed from the user to complete the segmentation step: (1) the minimum bone intensity value, (2) the minimum cortical intensity value, and (3) the cube size. The minimum bone

intensity value specifies the minimum CT intensity of a pixel in HU to consider that a bone tissue pixel. The minimum cortical intensity value specifies the minimum CT intensity of a pixel in HU to consider that pixel a cortical bone tissue pixel. The cube size parameter specifies the sampling density (resolution) for the 3D bone surface reconstruction. The PWT algorithm described in chapter (7).

After the segmentation algorithm finishes segmenting a 3D CT image, the interface displays the segmented image by marking pixels associated with each bone fragment with a unique color, see figure (16(c)). Figure (16 (d)) shows the 3D mesh extracted for each bone surface generated by applying the marching cubes algorithm on the segmented image data [47]. One extracted, the 3D surfaces are shown in the 3D canvas and added as objects to the tree panel.

4.1.2 Alignment of Bone Surfaces

The alignment process solves for the orientation and position provides a gross alignment between the intact surfaces and all of the fragments of the fracture. This step takes three data sets as inputs: (1) all bone fragment surfaces, (2) the surface of the unbroken bone, and (3) user input. The output of this step is an aligned intact tibia surface. The surface of an unbroken tibia is aligned in its position to the surfaces for tibia bone fragments. This step consists of three stages:

1. Partitioning the 3D surfaces into sub surfaces,

2. Classifying the sub surfaces,

3. Aligning the intact tibia,

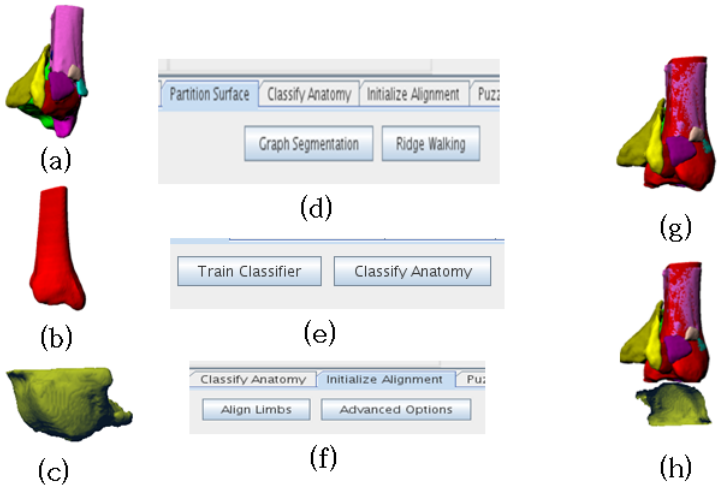


Figure 17: Illustration picture shows the interactions that take place to align 3D surfaces in step two.(a) a collection of 3D surfaces for bone fragment, (b) a 3D surface of an unbroken tibia, (c) a 3D surface of an unbroken talus, (d) an interface window for the partitioning stage, (e) an interface window for the classification stage, (f) an interface window for the alignment stage, (g) aligned intact tibia surface to the surfaces of the bone fragments, and (h) aligned unbroken talus surface to its relative position with respect to surfaces of tibia fragments.

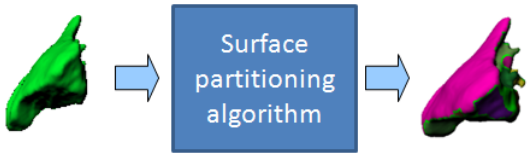


Figure 18: Illustration picture shows the result of partitioning a surface into patches in stage one of the alignment step.

Figure (17) shows illustrations of these three stages which ultimately produce a virtual 3D reconstruction of the bone.. Stage 1 takes as input a collection of 3D surfaces and outputs a collection of sub surfaces called patches. In this stage, each 3D surface is divided into patches, see figure (18). The system interface allows users to split, merge, and delete surface patches. Stage 2 takes as input the surface patches that were generated in stage 1 and assigns a label for each surface patch. There are three different labels: (1) fracture, (2) periosteal, and (3) articular, see figure (19). The system interface provides an automatic method and a manual method to classify

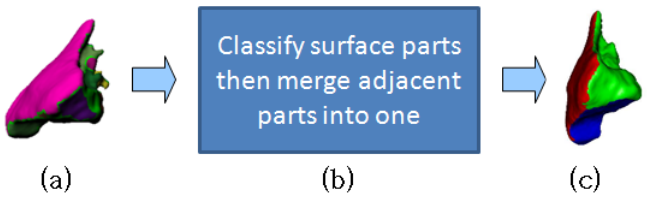


Figure 19: Illustration picture shows the output of the classification stage in the alignment step. (a) a fragment with a collection of patches, (b) the process of stage 2, (c) classified surface patches of the fragment.

patches. Stage 3 takes as input the labeled patches and outputs an aligned intact bone surface to the surfaces of bone fragments. This stage is accomplished by clicking on the button "Align Limbs", see figure (17 (f)).

4.1.3 Reconstruction of Bone Fragments

The reconstruction of tibia fragments estimates the original positions of tibia fragments before the tibia was fractured. This step takes three inputs: (1) an aligned surface for an unbroken tibia, (2) a collection of surfaces for tibia fragments, and (3) a users input. There are two outputs for this step: (1) reconstructed surfaces of tibia fragments, and (2) the reconstruction transformations for surfaces. The system interface provides two solutions for the reconstruction step: (1) an automatic solution and (2) a semi automatic solution. In the automatic solution, user click on the "Automatic Reconstruction" button which invokes the 3D puzzle solving algorithm described in [46], see figure (20 (b)). In the semi automatic solution, the interface allows users to manually improve the reconstruction of misaligned surfaces. The system interface provides a custom alignment algorithm called "jiggling" to assist users in correctly positioning misaligned surfaces [46]. Figure (20) shows an illustration picture for this step.

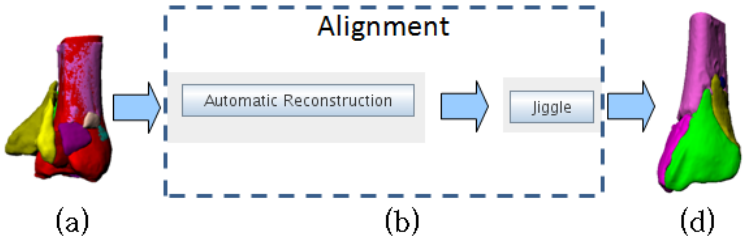


Figure 20: Illustration picture shows the interactions that take place to reconstruct 3D surfaces in step three. (a) A surface of an unbroken tibia and surfaces of tibia fragments in their aligned positions after the alignment step, (b) an interface window for the two reconstruction solutions, i.e., automatic and semi automatic, and (c) a reconstruction result.

4.2 Generating Attributed Models

Generating attributed models is the first step in the system of estimating a fracture inverse mechanics. This step takes two inputs: (1) a collection of 3D surfaces for fracture fragments and (2) a 3D CT image of an unfractured limb. The outputs of this step are: (1) a collection of attributed models for fracture fragments, (2) a collection of attributed models for soft tissues, and (3) an attributed model for a strike object. The outputs are generated in three parallel track. The user interacts in the input side to produce the output.

4.2.1 Track for Generating Attributed Models for Fracture Fragments

This track takes as input a collection of 3D surfaces for fracture fragments. The output of this track is a collection of attributed models for fracture fragments. A user specifies the following settings for physical attributes to generate these models: (1) bone density and (2) friction. Figure (21) shows the interface window for the settings to generate attributed models for fracture fragments. Bone density is the amount of substance per square centimeter of bones. Bone density is used to estimate the mass

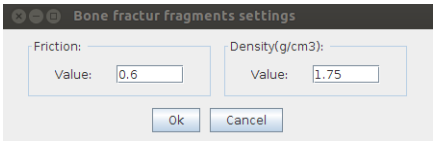


Figure 21: A screen snapshot of the interface window for bone fracture fragments. There are three dynamics characteristics: (1) friction, (2) bounciness, and (3) density. Friction and bounciness characteristics determine the amount of gain and loss in velocity and energy , respectively, during collisions, while density specify the amount of substance in g/cm^3.

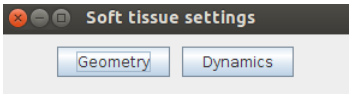


Figure 22: A screen snapshot of the interface window for soft tissue settings. The interface window consists of two button: geometry button and dynamics button. Using these buttons users are able to specify the settings for dynamics models for soft tissues for a fracture simulation.

for fracture fragments. Friction is a percentage value that specifies how much velocity is lost when a 3D surface collides with other objects. A friction value of 0 means no loss in velocity, while a value of 1 means complete loss in velocity. These attributes determine the amount of gain and loss in velocity and energy, respectively, for 3D surfaces during collisions.

4.2.2 Track for Generating Attributes Models for Soft Tissues

This track takes as input an intact image and provides as output a collection a collections of attributes models for soft tissues. A user specifies the following two groups of settings to generate these models: (1) geometry and (2) physical attributes. By specifying these settings, the physical attributes for soft tissues are generated. Figure (22) shows a snapshot of the interface window.

In the geometry settings, a user specifies a set of settings to generate surfaces for two types of soft tissues muscle and fat. These setting are: (1) fat threshold, i.e.,

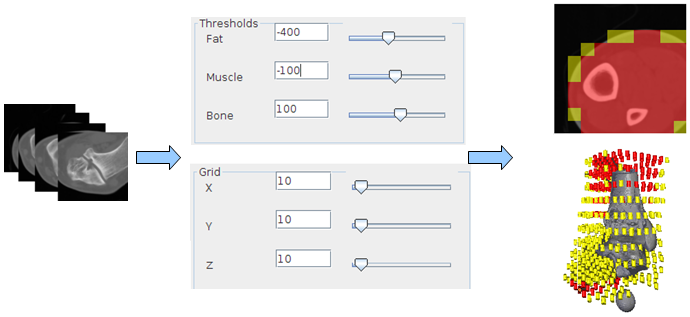


Figure 23: A screen snapshot of the interface window for the settings used to generate the geometry for soft tissues. The interface window consists of controls to specify settings for the process of generating soft tissue elements from a 3D CT image of an unfractured limb. There are two types of soft tissue elements: fat and muscle. The display of the 3D CT image marks pixels belong to fat elements in yellow, while pixels belong to muscle elements are marked in red.

the minimum intensity value in HU considered a fat tissue, (2) muscle threshold, i.e., the minimum intensity value in HU considered a muscle tissue, (3) bone threshold, i.e., the minimum intensity value in HU considered a bone tissue, (4) x grid, i.e., the number of grid divisions along the x-axis, (5) y grid, i.e., the number of grid divisions along the y-axis, and (6) z grid, i.e., the number of grid divisions along the z-axis. These settings helps a segmentation algorithm to extract the soft tissue surfaces from the intact image. Details for the segmentation algorithm are provided in section (5.2.2). After the segmentation algorithm finishes generating soft tissue surfaces, the interface displays a 3D CT image where pixels belong to fat tissue are marked with yellow color, while pixels belong to muscle tissue are marked with red color, see figure (23) .

In the physical attributes settings, a user specifies physical attributes for fat tissue models, muscle tissue models, and non-rigid constraints. The physical attributes for fat tissue models are: (1) fat density and (2) friction. Fat density is the amount of

substance per square centimeter of fat tissue. Fat density is used to estimate the mass for fat tissue models. Friction is a percentage that specifies how much velocity is lost when a fat object collides with other objects. A friction value of 0 means no loss in velocity, while a value of 1 means complete loss in velocity. The physical attributes for muscle tissue models are: (1) muscle density and (2) friction. Muscle density is the amount of substance per square centimeter of muscle tissue. Muscle density is used to estimate the mass for muscle tissue models. Friction is a percentage that specifies how much velocity is lost when a muscle object collides with other objects. A friction value of 0 means no loss in velocity, while a value of 1 means complete loss in velocity. The physical attributes for non-rigid constraints are: (1) damping, (2) linear limit, (3) angular limit, (4) stiffness, (5) neighbor max distance, and (6) breaking threshold. Damping specifies the percentage decrease in the energy stored in the oscillation of a constraint per second. Damping value of 0 means no loss in energy, while 1 means a total loss. Linear limit specifies the amount of translation allowed along each coordinate axis. Angular limit specifies the amount of rotation allowed around each coordinate axis. Stiffness specifies how bendy the constraint is. A low value creates a very weak constraint, while a high value creates a strong constraint. Neighbor max distance specifies the max distance between two adjacent elements connected by a constraint. Breaking threshold specifies the impulse strength that needs to be reached before a constraint breaks. Details for physical attributes for attributed models for soft tissue and their non-rigid body constraints are provided in section (5.2). Figure (24) shows a snapshot of the interface window.

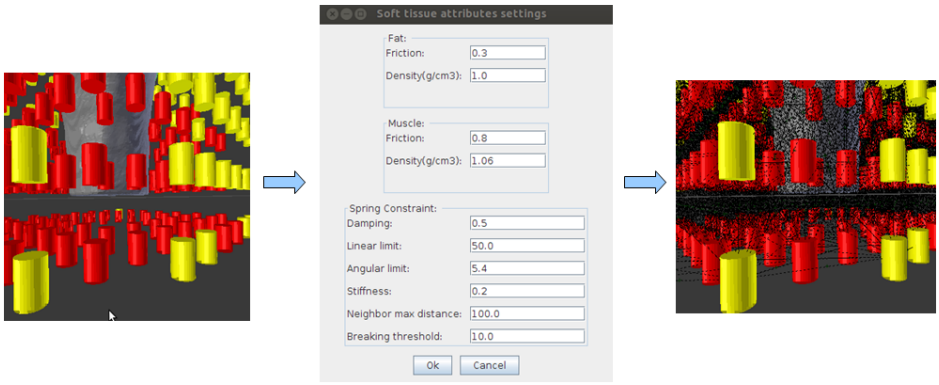


Figure 24: A screen snapshot of the interface window for the settings used to specify physical attributes for soft tissues. There are three groups of attributes: (1) fat, (2) muscle, and (3) spring constraints. These physical attributes control the behavior of soft tissues objects in a fracture simulation.

4.2.3 Track for Generating an Attributed Model for a Strike Object

This track takes a user input and provides as output an attributed model for a strike object. A user specifies the following settings to generate this model: (1) search type, (2) dimensions (x,y,z), (3) mass, (4) velocity, (5) direction and (6) position. Search type specifies the number of plausible solutions that should be generated by the system. There are two types: single and multi. The single type allows the system to generate one solution and allows the user to specify all other settings for the strike object in this interface. While the multi type allows the system to generate multiple solutions and does not allow the user to specify the direction and position settings since their space is sampled automatically by the system. Dimensions specify the length of a strike object along the x direction, y-direction, and z-direction in units. Mass specifies the amount of material in a strike object in kilograms. Velocity specifies how fast the strike object is moving. The direction specifies amount of velocity along the x-direction, y-direction, and z-direction in units per second. Figure (25) shows a

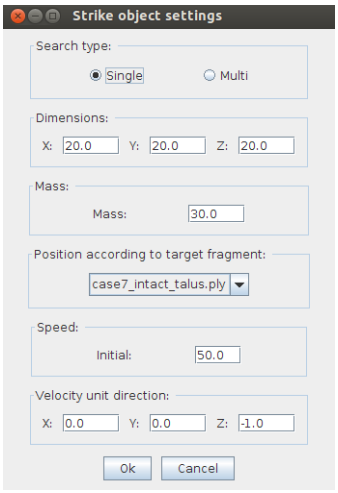


Figure 25: A screen snapshot of the interface window for the strike object dynamics settings. There are seven characteristics: (1) shape, (2) x dimension, (3) y dimension, (4) z dimension, (5) mass, (6) initial speed, (7) position, and (8) velocity unit direction. These physical attributes control the behavior of a strike object in a fracture simulation.

snapshot of the interface window. The position specifies the target impact location of the strike object on the virtual limb. In this interface, the position is specified by the center of the selected fragment. This is designed to simplify the determination of the targeted impact point to the user.

4.3 Dynamics Optimization

Dynamics optimization is the second step in the system of estimating a fracture inverse mechanics. This step takes three inputs: (1) a collection of attributed models for fracture fragments, (2) a collection of attributed models for soft tissues, and (3) an attributed model for a strike object. The output of this step is a set of plausible solutions for the fracture event. The user specify the following parameters to control the system iterative process to find the best fracture simulation: (1) time interval, (2) environment damping, and (3) the maximum number of iterations. Time

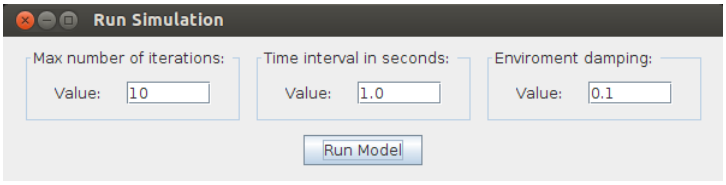


Figure 26: A screen snapshot of the interface window for the simulation optimization settings. There are three unknown settings: (1) maximum number of iterations, (2) time interval, and (3) environment damping. These settings help the system to in finding a simulation that best matches the fracture observed in a fracture case.

interval specifies the time duration length in seconds for a fracture event simulation. Environment damping specifies the percentage decrease of linear/angular velocity per second for each object in the simulation. The damping of linear velocity affects the freedom of an object to move. The damping of angular velocity affects the freedom of an object to rotate. A damping value of 0 means there is no decrease in velocity, while 1 means complete loss of velocity. In space there is almost zero damping, while in water the damping value should be set quite high. The maximum number of iterations determines the maximum number of trials to perform before stopping the search process for a plausible solution. The details for the search process are provided in section (5.3). Figure (26) shows a snapshot of the interface window. These settings help the system in finding the set of plausible solutions given a user input.

4.4 Fracture Simulation Analysis

Fracture simulation analysis is the third step in the system of estimating a fracture inverse mechanics. This step takes as input a set of plausible solutions for a fracture event. The system provides three tools to explore the space of plausible solutions: (1) list of plausible values for simulation variables, (2) play a fracture simulation and, (3) visualize the estimated soft tissue void that was created during the fracture event.

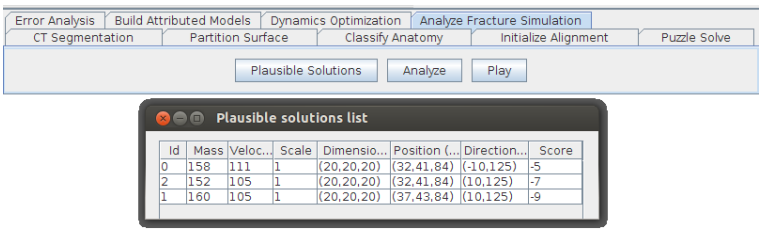


Figure 27: A screen snapshot of the interface window for the list of plausible solutions for the fracture event. The solutions are presented in a descending order according to the likelihood score value. The user is able to select a specific value from the list to visualize later its fracture simulation and the estimated soft tissue void created in that simulation.

These tools allow the user to explore the solution space to understand how a fracture event may have occurred.

The list of plausible values for simulation variables reports the solutions found for a fracture event. The simulation variables specify the initial conditions for the strike object that virtually hits the virtual limb as described in chapter (5). The the plausible values are listed in a descending order according to a score value called likelihood score as shown in figure (27). This value measures how likely a solution is based on the similarity between the fracture pattern generated by a fracture simulation and the fracture pattern observed in a fracture image. The system allows the user to select a specific value from the list to visualize later its fracture simulation and the estimated soft tissue void created in that simulation.

Playing a fracture simulation allows the user to visualize how bone fragments moved during a fracture simulation. The fracture simulation that is played is the one generated using the selected value in the list of plausible solutions. Figure (28) shows the interface window for the fracture simulation player. The player allows the user to perform four operations on a fracture simulation: (1) play, (2) pause, (3) stop,

Figure 28: A screen snapshot of the interface window for the simulation player. The interface allows the user to play, pause, stop, and navigate through the recorded keyframes for a fracture simulation. Playing a fracture simulation allows the user to visualize how bone fragments moved during a fracture simulation.

and (4) navigation. The play operation allows the user to automatically move the bone fragments and the strike by changing their positions and orientation according to the recorded keyframes generated by the fracture simulation at a specific rate, i.e., keyframe/sec. In this interface, the used rate is two keyframes per second. The pause operation allows the user to temporary stop the movement of bone fragments and the strike at the current keyframe. The stop operation allows the user to completely stop the movement of bone fragments and the strike object and to display them at their positions as recorded in the first keyframe. The navigation operation allows the user to go back and forth through the keyframes as well as to seek for a specific keyframe. The user may change the viewperspective for the display during these operations. These operations may help the user to visualize how bone fragments move during a fracture event.

Visualizing the estimated soft tissue void that was created during a fracture event allows the user to estimate the location, shape, and amount of pushed soft tissue. This tool provides a 2D and a 3D representation of the void area. The 2D representation highlights the pixels on the fracture image where bone fragments passed through during a fracture simulation. For each slice, the estimated void pixels are counted and highlighted on that slice, see figure (29) . The 3D representation generates a surface for the void volume. This surface is displayed on the 3D canvas area, see

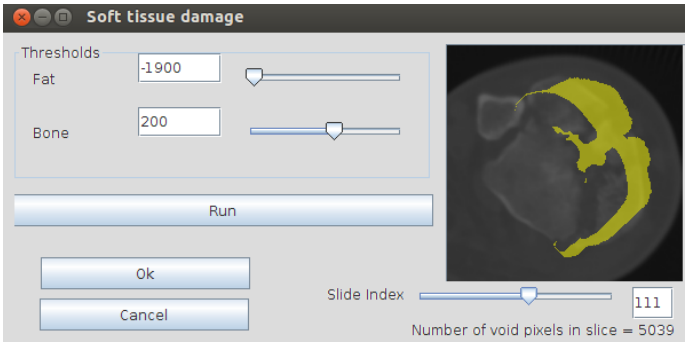


Figure 29: A screen snapshot of the interface window for the 2D representation of the void region in soft tissue. The interface provides: (1) a display window to view the void region pixels highlighted in yellow for each 2D slice, (2) a slide control to allow the user to navigate through the different image slices, (3) two slide controls to allow the user to specify the minimum intensity values to classify a pixel as a soft issue and as a bone, and (4) a button control to start the computation of the void region as described in section (5.3.1). The 2D representation highlights the pixels on the fracture image where bone fragments passed through during a fracture simulation. The number of void pixels in the displayed slice is shown in the interface window.

figure (30) . These two representations allow the user to estimate the location of the created soft tissue void on the fractured limb, as well as the estimated volume of that void.

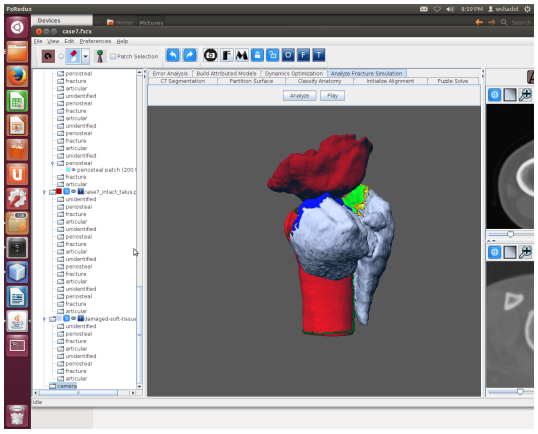


Figure 30: A screen snapshot of the interface window for the 3D representation of the void region in soft tissue (light blue). The 3D representation generates a surface for the void region volume. This surface is displayed on the 3D canvas area along side other bone fragments to show its estimated relative shape and position in the fractured limb.

CHAPTER 5: COMPUTATIONAL INVERSE MECHANICS SYSTEM FOR A HIGHLY COMMINUTED FRACTURE

This chapter describes a system to generate a virtual fracture model and subsequently the inverse mechanics of a fracture event for that model. The process that the system uses to estimate a fracture event consists of two parts: (1) estimate the "initial" state of fractured limb, i.e., the state of the bone at the time the bone fracture fragments were generated, and (2) search for values of the fracture event variables that explain the 3D fracture CT data. Since the system goal is to estimate a solution to the inverse kinematics problem of the fracture, the proposed problem is both complex, i.e., it depends on a large number of dependent variables, and ill-posed, i.e., it may have many plausible solutions. The system copes with these difficulties by using fracture reconstruction software to reduce the number of unknowns associated with the problem and by providing a collection of solutions for the user to examine rather than a single solution. Users are expected to integrate their domain knowledge (medical or forensic) and other external information to interpret these results to facilitate their analysis (for surgical planning or forensic investigation).

Figure (31) shows a block diagram of the system. The system consists of three separate components:

1. Estimate the intact limb, i.e., the state of the limb at the beginning of a fracture event,

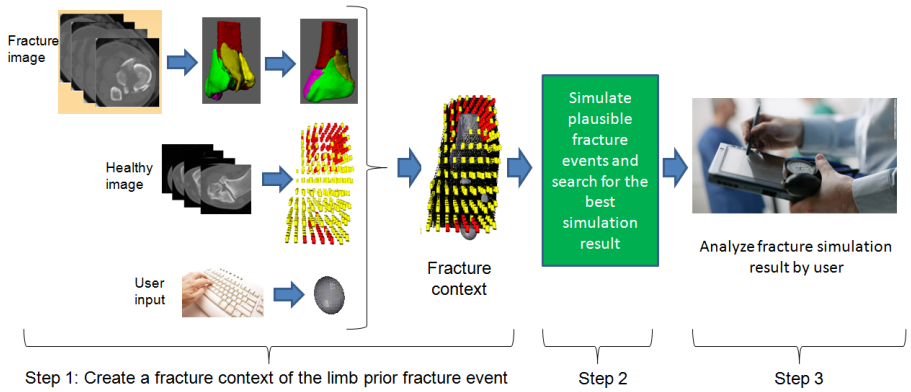


Figure 31: The fracture event analysis system consists of three operations: (1) generate an estimate the intact limb from CT images and user input, (2) perform virtual simulations to find plausible values for fracture event variables, and (3) display to the user plausible simulations ranked by their likelihood score for visualization and analysis.

2. Search for the values of the fracture event variables that agree with the data in the fractured limb 3D CT image,

3. Display plausible fracture event simulations to the user in a 3D virtual environment for analysis.

The listed sequence describes a process to virtually computationally reconstruct and analyze a fracture event. To my knowledge the proposed approach represents the first system to attempt to recreate the fracture event. It includes a novel framework which makes computation on this difficult problem tractable by using innovative technologies. It also provides a rationale for addressing ill-posed inverse problems by allowing users to navigate a collection of plausible solutions rather than displaying a single "best" solution.

The system uses a model of the intact limb to simulate fracture events. While a complete limb model would include models for numerous body substructures, e.g., bone, muscle, skin layers, connective tissue, tendons, etc. This prototypical system

categorizes limb tissue into two basic types: (1) bone tissue and (2) soft tissues (fat and muscle). Since the limb has already been injured, the models for both the intact bone tissue and the intact soft tissue for the intact limb must be estimated.

An estimate of the intact bone tissue for the intact limb is obtained by applying the bone fracture reconstruction system [46]. This system takes as input the fracture image and an intact image. Typically the intact image is generated by imaging the undamaged contra-lateral limb of the patient and subsequently transforming the image by negating or "flipping" the image across the anatomic plane of symmetry. As part of the reconstruction process, the reconstruction system extracts the bone fracture fragments from the fracture image via the PWT segmentation algorithm described in chapter (7). Let $\mathbf{B}_i$ denote the collection of (x, y, z) coordinates identified as members of the i^{th} bone fragment and let $\mathbf{B} = \cup_i \mathbf{B}_i$ denote the collection of all bone fragment 3D coordinates in the fracture image. The segmentation algorithm is also applied to the intact image to extract the the intact bone surface. Reconstruction then proceeds by aligning the surfaces of the bone fragments to the surface of the intact bone. In this way the intact bone model acts as a template into which the bone fragments are fit to reconstruct a model of the fractured bone from its fragments. The reconstruction system provides a Euclidean transformation for each bone fragment that repositions the bone fragment from its as-found position in the fracture image to its estimated anatomic position within the fracture image. Let $\mathbf{T}_i$ denote the Euclidean transformation that transforms the points of bone fragment $\mathbf{B}_i$ to their estimated anatomic location. For simplicity we collect the transformation parameters for a fracture involving N fragments into a single parameter $\mathbf{T}$ which incorporates 6

variables for each fracture fragment, i.e., it includes 6N variables. Using this notation we can denote the state of the bone tissue for the intact limb as in equation (20).

$$\mathbf{B}(t_0) = \{\mathbf{T}\}_i(t_0)\mathbf{B}. \tag{20}$$

Where t_0 denotes the initial, i.e., beginning time of the fracture event and $\{\mathbf{T}\}_i$ denotes the Euclidean transformations for all of the bone fragments. This notation is also used for fracture event simulation. Specifically, the position and orientation of the bone fragment data at a generic time, t , is denoted as simply$\mathbf{B}(t)$.

An estimate of the soft tissue of the intact limb model is generated by segmenting soft tissues from the registered intact image. While many soft tissues exist, the prototype system characterizes regions of soft tissue generically as either fat or muscle. The segmentation algorithm is simplistic as it is needed only to differentiate soft tissue from objects other than bone which, in typical bone fracture CT data, is typically only open air. The tissue segmentation algorithm identifies (x, y, z) locations in the 3D CT intact image associated with soft tissue of the damaged limb. Simulation of the fracture event requires incorporation of the tissue as a geometric model within the virtual physical simulation. The system models soft tissue as a lattice of simple geometric elements (currently cylinders) connected by breakable elastic constraints. Let $\mathbf{S}_j$ denote the j^{th} element within the 3D model for the soft tissue and $\mathbf{S} = \cup_j \mathbf{S}_j$ denote the collection of all 3D soft tissue model elements. Under simulation, the soft tissue elements will rotate and translate over time due to impact and fragment dispersion associated with the fracture event. Let $\mathbf{T}_j(t)$ denote the transformation for $\mathbf{S}_j$ at time t. the generic position of all the soft tissue data at time t is denoted

in equation (21).

$$\mathbf{S}(t) = \{\mathbf{T}\}_j(t)\mathbf{S} = \cup_j \mathbf{T}_j(t)\mathbf{S}_j. \tag{21}$$

Where $\{\mathbf{T}\}_j$ denotes the Euclidean transformations for all of the geometric elements (cylinders) of the intact limb soft tissue model. The position of a soft tissue lattice element at a generic time, t, is denoted as $\mathbf{S}_j(t) = \mathbf{T}_j(t)\mathbf{S}_j$. The position of for all soft tissue lattice elements at the beginning time of the fracture event, t_0, is denoted as simply $\mathbf{S}(t_0)$.

The end result of the image processing functions and reconstruction system are geometric estimates for the intact/unbroken bone in terms of its fragments, $\mathbf{B}(t_0)$, and the soft tissue surrounding the intact bone, $\mathbf{S}(t_0)$, which is further subdivided into elements of fat or muscle. These two elements combine to form the estimate of the geometry intact limb needed to simulate the fracture event.

Fracture event simulations seek to approximate the physical process that occurred when the injury was sustained. The simulation component of the system accomplishes this task in 3 steps: (1) the estimated geometry of the intact limb model is assigned physically-meaningful attributes, (2) a strike object is generated whose purpose is to deliver the fracture impact, and (3) a search algorithm searches for values of the fracture event that are plausible. The output of this process is a collection of fracture event simulations where each simulation is ranked by a likelihood score. Each simulation score indicates the likelihood of the fracture image data given the assumed value for the fracture event variables.

Each fracture simulation is carried out in 3D using the Bullet physics modeling engine and the Python interface to Bullet made available through the open-source Blender modeling package [54]. Simulation data is passed to the Bullet engine and the transformation data, i.e., $\mathbf{T}$ for both the soft tissue elements and the bone fragments are recorded when the simulation is complete.

The dynamics simulator records the transformation data for the continuous process of a fracture event by recording the generic position of all virtual objects in the limb model at specific time intervals called keyframes as defined in equation (22).

$$\{\mathbf{B}(n), \mathbf{S}(n)\} = \{\mathbf{B}(t), \mathbf{S}(t)\} \mid_{nTs, n\in[0,N]}. \tag{22}$$

Where Ts is the time sample interval and $N+1$ is the number of samples such that $NTs = t_f$, where t_f is the final time for the fracture event. The recorded positions for all virtual objects in the limb model at the start of a fracture simulation is denoted by $\{\mathbf{B}(0), \mathbf{S}(0)\}$. While, the recorded positions for all virtual objects in the limb model at the end of a fracture simulation is denoted by$\{\mathbf{B}(N), \mathbf{S}(N)\}$.

Estimation of the intact limb model provides a geometric description of the limb at the time immediately prior to the impact. However, to simulate the fracture event, these geometric models have to be attributed with physically-meaningful attributes. Physical attributes for bone tissue are a density and a coefficient of friction. Constraints are implemented that specify bone tissue as rigid, i.e., non-deformable, objects. Physical attributes for soft tissue also include a density and a coefficient of friction for fat elements and muscle elements. These quantities determine the mass of the 3D objects in the intact limb model and how energy is dissipated as these objects

contact with each other due to motion.

The cylindrical elements of the soft tissue model are also connected by a lattice of "breakable constraints." Breakable constraints within the Bullet engine specify elastic/spring-mass connections between the soft tissue elements that constrain their joint movement. These constraints can "break" when the force exerted on the connection exceeds the connection bond strength. For the soft tissue model, the breaking condition serves to approximate how soft tissue will tear apart when placed under severe forces.

A strike object must be created as the object which delivers the traumatic impact to the intact limb model. The strike object geometry and physical attributes are typically unavailable for measurement at the point of treatment. The prototypical system requires the user to specify a geometry, position, orientation, and physical attributes for the strike object. This information provides an initial point within the fracture event space, i.e., this provides sufficient data to perform a fracture simulation. Yet, this user-specified data will be very noisy and inaccurate and when simulated the fracture outcome is likely to be significantly different from that observed in the measured fracture image.

For this reason, a search procedure is prescribed that varies the user-specified variables of the fracture event to find values for these variables that are supported by the recorded fracture image data. For the prototypical system described here, the fracture event variables are the variables that specify the geometry, position, direction of travel and velocity of the strike object at the initial time of the fracture event; denoted Θ. The resulting vector of unknown fracture event variables is $\mathbf{\Theta} = [v,\, m,\, s,\, \mathbf{p},\, \mathbf{d}]^t$

where v denotes the strike object velocity, m denotes the strike object mass, s denotes the scale factor for the strike object size, $\mathbf{p}$ denotes the 3D position of the strike object to hit and $\mathbf{d}$ denotes the 3D direction of travel of the strike object as a unit vector. Under these circumstances $\mathbf{\Theta}$ is a vector of 8 variables.

The search procedure attempts to efficiently search the space of all plausible values for the fracture event variables to find those which generate fracture patterns similar to that measured in the fracture image. The search is facilitated by providing an external scoring function which rates the quality of a chosen set of fracture event values. The system applies a Maximum Likelihood Estimation (MLE) framework for scoring specific collections of fracture event variable values. Let $\mathcal{D}$ denote the fracture image data and $p(\mathcal{D}|\Theta = \Theta_i)$ is the likelihood of that data given the fracture event variables $\mathbf{\Theta} = \Theta_i$. The fracture image data in this system consists of bone fragment data only, i.e., $\mathcal{D} = \{\mathbf{B} = \cup_i \mathbf{B}_i\}$, since they are easy to identify with respect to soft tissue and to represent by a virtual structure. The conceptual goal of the MLE framework is simple, for each guess of the fracture event variables, $\Theta = \Theta_i$, a likelihood value will be computed which reflects the likelihood that the fracture image data given the chosen, $p(\mathcal{D}|\mathbf{\Theta} = \Theta_i)$. Values of Θ with higher likelihoods are values of the fracture event variables that are better supported by the fracture image data, i.e., they are "more likely." The likelihood distribution chosen is a Gaussian distribution which expresses the probability of the all data as the sum of the differences between the simulated bone tissue intensities at the final time, $\mathbf{B}(t_f)$, and the bone tissue intensities identified by segmenting the fracture image, $\mathbf{B}$ as shown in equation (23).

$$p(\mathcal{D}|\mathbf{\Theta}) = p(\mathbf{B}|\Theta, \mathbf{B}(t_f)) = k * \exp\left(-\frac{1}{2}\left\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\right\|^2\right) \tag{23}$$

Since $\mathbf{B}(t_f) = \{\mathbf{T}\}_i(t_f|\Theta)\mathbf{B}$ we can see that equation (23) is ultimately seeks values of the fracture event variables that cause the simulated bone fragments final position to coincide with their segmented position in the fracture image. In this case $\mathbf{B}(t_f) = \mathbf{B}$ and the likelihood function is a maximum.

The search procedure guesses at values for $\mathbf{\Theta}$ as inputs for simulated fracture events. When the simulation ends final transformation values for the bone fragment data are provided, denoted $\hat{\mathbf{T}}$. Each of these values for $\hat{\mathbf{T}}$ let us evaluate $p(\mathcal{D}\,|\mathbf{\Theta} = \mathbf{\Theta}_i)$ and generate a likelihood score for the guessed $\mathbf{\Theta}$ value. The search procedure assigns each value of $\mathbf{\Theta}_i$ the likelihood score $p(\mathbf{B}|\mathbf{\Theta}, \mathbf{B}(t_f))$ and stores the result in a list sorted in order of decreasing likelihood.

This prototypical system does not use a perfect model for the limb and it uses a bone model that is already broken into fragments. The limb model is not perfect since the system does not incorporate models for tendons or ligaments. The system uses a bone model that is broken where bone fragments are already separated, so that it does not account for any energy that is taken for bending the bone or creating the fracture. That energy is subtracted from the kinetic energy for the strike object. Despite this imperfection of the model, the system provides a first trial of how to estimate a fracture event from 3D CT images of a fractured limb.

The system requires three collections of data sets as inputs: (1) a fracture image, (2) an intact image, and (3) a user input. A fracture image is a 3D CT image of

a fractured limb and it is used to measure the data for fracture fragments **B**. An intact image is a 3D CT image of an unfractured limb and it is used for two purposes: (1) estimating the data for soft tissue **S** and (2) estimating virtually the original unbroken bone for the reconstruction method in [46]. The user input specifies the physical attributes for different attributed models as well as settings to control the search process for the initial conditions of the strike object. The output of the system is a collection of recorded transformation data for all virtual objects in the limb model and the virtual strike object. This collection provides plausible solutions of how bone fragments moved from their anatomic positions to their fractured ones during the fracture event. The user interacts with the system at both the input side and the output side. The user helps the system to construct the virtual limb model in the input side, and he also navigates and analyzes the result visually at the output side.

5.1 Blender Virtual Modeling Software Overview

Blender is a free and open-source 3D computer graphics program that enables dynamics simulations as well as the creation of a diverse range of 2D and 3D content. Blender approximates the physical behavior of virtual objects using the Bullet engine. Blender interfaces with the Bullet engine using the Python scripting computer language. Blender also provides a 3D modeling environment that allows users to instantiate geometric objects, relationships, and attributes for simulations. There are three components from Blender are used in a fracture simulation: (1) rigid body, (2) rigid body constraints, and (4) animation tool. These are are used to create virtual models, assign them attributes, and specify their motion.

5.1.1 Rigid Body Component

Blender allows users to interact with objects as virtual solid non-deformable bodies called rigid bodies. A rigid body does not change shape over time. Each rigid has attributes that control its behavior in simulations. From these attributes, nine attributes are used in fracture simulations: (1) shape, (2) name, (3) position, (4) orientation, (5) type, (6) mass, (7) damping, (8) friction, and (9) animated. These attributes characterize a rigid body and describe its behavior in fracture simulations. The shape attribute is determined by a set of polygonal faces, edges, and vertices, referred to as a mesh. The mesh of a rigid body is created manually, imported from other 3D applications, or automatically using primitive shapes. There are three primitive shapes used in a fracture simulation: (1) cube, (2) sphere, and (3) cylinder. The name attribute is a unique text that identifies an object. The position attribute specifies the location of the center of an object in global coordinates. The orientation attribute specifies the rotation angles of an object around the global axes at the object's center. The type attribute defines the role of a rigid body object in simulations. There are two types of rigid bodies: (1) active and (2) passive. Active body is dynamically simulated so its position and orientation may change during simulations. While, passive body remains static so its position and orientation do not change during simulations. The mass attribute specifies how heavy an object is. The mass attribute affects the force that is required to move an object. The greater the mass the greater the force that is needed to move an object. The damping attribute is a percentage decrease of linear/angular velocity per second. The damping of linear

velocity affects the freedom of an object to move. The damping of angular velocity is only affects the freedom of an object to rotate. In space there should be almost zero damping, while in water the damping value should be set quite high. The friction attribute specifies how much velocity is lost when objects collide with each other. The animated attribute enables or disables a rigid body to be driven by the animation system, i.e., not according to Newton's laws of dynamics, refer to section (5.1.3).

5.1.2 Rigid Body Constraint Component

Blender allows users to connect rigid bodies to each others using virtual joints called rigid body constraints. Constraints are used to limit and control the freedom of motion for an object either in global space or relatively to other objects. Constraints control the position and orientation attributes of rigid bodies.

Each rigid body constraint has attributes that control its behavior in simulations. For fracture simulations, nine of these attributes are used: (1) name, (2) constrained objects, (3) location, (4) axes, (5) limits , (6) springs, (7) type, (8) breakable, and (9) breaking threshold. These attributes characterize a constraint and describe its behavior in fracture simulations. The name attribute is a text that identifies a constraint. The constrained objects attribute determines the two objects that are connected to a rigid body constraint. The location attribute specifies the position of the entity hosting the physics constraint. The location of the constraint remains fixed during a simulation and distinct from the two constrained objects. The axes attribute specifies the x-axis, y-axis, and z-axis of a constraint. These axes are distinct from the global axes for the two constrained objects. The origin of the axes for a constraint

is specified by the location attribute of that constraint. The limits attribute specifies the translation and rotation range for constrained objects with respect to the axes of their connecting constraint. The limits for the translation has three parameters for each axis: (1) enable, (2) lower limit, and (3) upper limit. The enable parameter enables translation limit for the specified axis. The lower limit parameter specifies the lower translation limit along the axis. The upper limit parameter specifies the upper translation limit along the axis. The limits for the rotation has three parameters for each axis: (1) enable, (2) lower limit, and (3) upper limit. The enable parameter enables rotation limit for the specified axis. The lower limit parameter specifies the lower rotation limit from the axis. The upper limit parameter specifies the upper rotation limit from the axis. The springs attribute specifies the bending and bouncing limits for constrained objects along the axes of a constraint. For each axis, the springs attribute has three parameters: (1) enable, (2) stiffness, and (3) damping. The enable parameter enables spring for the specified axis. The stiffness parameter specifies how bendy the spring is from the axis. The damping parameter specifies the amount of damping the spring has along the axis. The type attribute specifies what the constraint is. For a fracture simulation, the used type is a generic spring. A generic spring constraint uses the limits and springs attributes. A generic spring constraint uses the limit attribute to limit the amount of translation and rotation between its objects with respect to its axes. while, it uses the springs attribute to allow its objects to bounce around. The breakable attribute allows a constraint to break during simulation. The breaking threshold attribute specifies the force needed to break a constraint. Each constrained object has its own force value according to

its mass. The force value for a constrained object, k, is calculated as follows:

$$Force_k = a \cdot Mass_k$$

where $Force_k$ is the force value that is needed to break constrained object k, a is the breaking acceleration threshold for the constraint, and $Mass_k$ is the mass of object k. According to this equation, heavy objects need stronger force to break than light objects.

5.1.3 Animation Component

Blender provides an animation tool to make objects move or change shape over time. The animation tool takes objects and specified paths as input, then, it moves these objects along these paths using criteria that specify their velocities and accelerations. For an animated object, the velocity and acceleration are specified using a method called keyframes. At a certain time, the object needs to be at a specific location. That constraints the acceleration and the velocity of the object. This is important for specifying the initial conditions by which a fracture event was created. The object that impacts the bone needs to be animated using the animation tool to create the fracture event. For this object, the only thing that needs to be controlled by the animation tool is the velocity which determines the rate of change in position for the center of the object.

Velocity is the first derivative of a position function for an object with respect to time. Velocity is mathematically defined as:

$$\mathbf{v}(t) = \lim_{\triangle t \to 0} \frac{\mathbf{x}(t + \triangle t) - \mathbf{x}(t)}{\triangle t}, \tag{24}$$

where $\mathbf{v}(t)$ and $\mathbf{x}(t)$ are the velocity and position of an object at time t, respectively. $\triangle t$ is a time interval. When $\triangle t$ has a fixed non-zero value instead of approaching zero, the velocity is approximated using the difference in position of the object at the specified time interval, i.e.,:

$$\mathbf{v}(t) \approx \frac{\mathbf{x}(t + \triangle t) - \mathbf{x}(t)}{\triangle t}. \tag{25}$$

This equation divides the forward difference in position by the time interval$\triangle t$ to approximates the velocity. For an animated object, the velocity is specified using the keyframes method. The keyframes method saves complete positions of an animated object for units of time referred to as frames. The velocity of an animated object between two consecutive frames is the rate of change in position of the object's center between these two frames. For an animated object, the velocity between two consecutive frames fr_i and fr_{i+1} is computed as follows:

$$\mathbf{v}_{anime}(fr_i, fr_{i+1}) = (\mathbf{x}_{anime}(fr_{i+1}) - \mathbf{x}_{anime}(fr_i)) \cdot fps \tag{26}$$

where, $\mathbf{v}_{anime}(fr_i, fr_{i+1})$ is the velocity of an animated object between the two consecutive frames fr_i and fr_{i+1}. $\mathbf{x}_{anime}(fr_i)$ and $\mathbf{x}_{anime}(fr_{i+1})$ are the positions of an animated object at frames fr_i and fr_{i+1}, respectively. fps is the number of frames per second referred to as frame rate. So, in order to move an animated object at a specific velocity, $\mathbf{v}$, between the two consecutive frames fr_i and fr_{i+1}, an animator has

to create two keyframes at these two frames such that the change in position is equal to the specified velocity divided by the frame rate, i.e., $(\mathbf{x}_{anime}(fr_{i+1}) - \mathbf{x}_{anime}(fr_i)) = \frac{\mathbf{v}}{fps}$. In simulations, the animation tool changes the position of the animated object accordingly at these specified frames.

5.2 Generating Attributed Models

Generating attributed models is the first step in the system of estimating inverse mechanics of a bone fracture. This step takes three inputs: (1) a fracture image I_f, (2) an intact image I_n, and (3) a user input. The outputs of this step are four: (1) a set of attributed models for bone fracture fragments $\mathbf{B}(t_0)$, (2) a set of attributed models for soft tissues $\mathbf{S}(t_0)$, (3) an attributed model for a strike object, denoted by O, and (4) the bone fragment data as provided by a segmentation of the fracture image $\mathcal{D}$. This step seeks to estimate a virtual representation of a fractured limb before a fracture event occurred. The user input provides settings that help the system to construct the virtual limb model in the input side.

5.2.1 Generating Attributed Models for Fracture Fragments

The generation process for attributed models for fracture fragments requires three inputs: (1) a fracture image I_f, (2) an intact image I_n, and (3) a user input. The outputs of this process are two: (1) a set of attributed models for bone fracture fragments $\mathbf{B}(t_0)$ and (2) the bone fragment data as provided by a segmentation of the fracture image $\mathcal{D}$. The system obtains the attributed models for fracture fragments by applying the bone fracture reconstruction system in [46] to obtain the surfaces of bone fragments then by assigning these surfaces physical attributes, i.e., density and

friction. The user input helps the system in controlling the reconstruction system and in specifying the physical attributes for the generated surfaces.

The system generates the attributed models for fracture fragments as a sequence of four steps:

1. Estimate the bone fracture fragments from the fracture image I_f,

2. Estimate the intact bone surface from the intact image I_n,

3. Reconstruct the surfaces of the bone fragments by aligning them to the surface of the intact bone,

4. Assign physical attributes for the generated surfaces of the bone fragments.

The listed sequence describes a process to generate virtual attributed objects for bone fragments in a fracture event.

In step (1), an estimate of the bone fracture fragments, $\mathbf{B}$, is generated by segmenting bone tissues from the registered fracture image I_f. While many algorithms have been developed for bone segmentation, fractured bone segmentation is a very challenging problem because the osseous tissue that forms the bones does not always produce distinguishable features from soft tissue regions in CT images. The system applies the novel segmentation algorithm described in chapter (7) to segment bone fragments within CT images. The proposed approach uses the Probabilistic Watershed Transform (PWT) algorithm to accomplish this goal. The PWT is a formulation of the classical watershed transform to perform segmentation using probabilistic models. The algorithm classifies the image pixels into three sets: high confidence bone

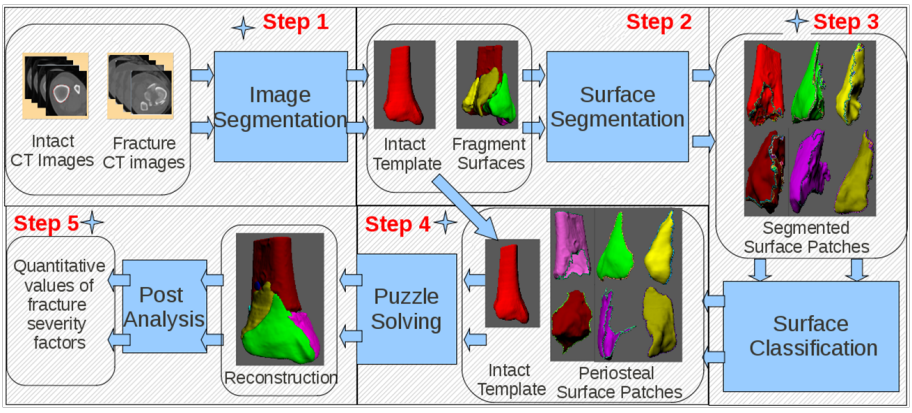


Figure 32: The system diagram for extracting and reconstructing fracture fragments using CT images of an intact and a fracture cases. These fragments are used to specify the geometry of the attributed models for bone fragments for a fracture event [46].

areas, referred to as markers, low confidence bone areas, and non bone areas. Each high confidence area defines a unique bone fragment and is assigned a unique label. Then, the algorithm classifies low confidence bone areas by propagating the labels from high confidence areas to lower confidence ones following the path with maximum confidence. The collection of pixels with the same label represents a unique bone fragment $\mathbf{B}_i$.

In step (2), an estimate of the intact bone surface is generated by segmenting bone tissues from the registered intact image I_n. The system applies the PWT segmentation algorithm described in chapter (7) to segment bone fragments within CT images. The intact bone surface is used to align the fracture bone fragments to estimate their anatomic positions before the fracture event occurred.

In step (3), the system reconstructs the surfaces of the bone fragments by aligning them to the surface of the intact bone. The intact image is aligned (registered) to the fracture image by aligning the intact bone surface with the surface of the remaining/undisturbed portion of bone tissue from the fracture image, i.e., the bone

fragment in the fracture image which remains in correct anatomic position after the fracture has occurred (also referred to as the "base fragment" in [46]) figure (32). Reconstruction then proceeds by aligning the surfaces of the bone fragments to the surface of the intact bone. In this way, the intact bone model acts as a template into which the bone fragments are fit to reconstruct a model of the fractured bone from its fragments. Geometric alignments are performed using the Iterative Closest Point (ICP) algorithm [11]. The details of this work is out of the scope of this dissertation. As a result, the reconstruction system provides a Euclidean transformation, $\mathbf{T}_i$, for each bone fragment that repositions the bone fragment from it's as-found position in the fracture image to it's estimated anatomic position within the fracture image. This estimated anatomic position represents the initial state of the bone tissue for the intact limb at t_0 as $\mathbf{B}(t_0) = \{\mathbf{T}\}_i(t_0)\mathbf{B}$.

In step (4), physical attributes are assigned for the generated surfaces of the bone fragments to give them physical meaning for the simulation. Physical attributes for bone tissue are a density and a coefficient of friction. The quantity of these attributes are specified by the user through the system interface, refer to chapter (4). These quantities determine the mass of the 3D objects in the intact limb model and how energy is dissipated as these objects contact with each other due to motion, respectively.

The end result of the image processing functions and reconstruction system described in this section are geometric estimates for the intact/unbroken bone in terms of its fragments, $\mathbf{B}(t_0)$, and the bone fragment data as provided by the segmentation of the fracture image $\mathcal{D} = \mathbf{B}$. This result form the estimate of the physically

attributed geometry for the intact bone needed to simulate the fracture event.

5.2.2 Generating Attributed Models for Soft Tissue

The generation process for attributed models for soft tissue requires two inputs: (1) an intact image I_n, and (2) a user input. The output of this process is a set of attributed models for soft tissue $\mathbf{S}(t_0)$. The system obtains an estimate of the soft tissue of the intact limb model by segmenting soft tissues elements from the registered intact image then by assigning these elements physical attributes, i.e., density, friction, and breakable constraints. The user input helps the system in controlling the segmentation algorithm for soft tissue and in specifying the physical attributes for the generated soft tissue elements.

The system performs this process as a sequence of two steps:

1. Estimate the soft tissue elements $\mathbf{S}(t_0)$ from the intact image I_n,

2. Assign physical attributes for the generated elements of the soft tissue.

The listed sequence describes a process to generate virtual models for the soft tissue found in the fractured limb before the fracture event occurred.

In step (1), an estimate of the soft tissue of the intact limb model is generated by segmenting soft tissues from the registered intact image. While many soft tissues exist, the prototype system characterizes regions of soft tissue generically as either fat or muscle. The segmentation algorithm is simplistic as the system needs only to differentiate soft tissue from objects other than bone which, in typical bone fracture CT data, is typically only open air. For this reason the segmentation approach classifies the pixels of the image into three categories: (1) fat pixels, (2) muscle pixels,

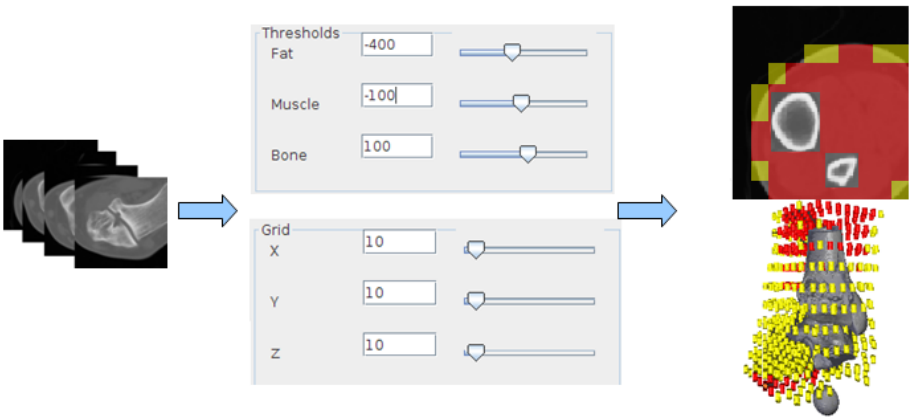


Figure 33: (left) a "stack" of CT images with unknown soft tissues. (middle) the user specifies basic segmentation criteria and a resolution for the categorization of the image into the classes fat, muscle, and other. (right) a labeling of the CT image into fat (yellow) and muscle (red).

and (3) non-soft tissue pixels. These regions are estimated using a global threshold where a user specifies three threshold values: (1) fat T_{fat}, (2) muscle T_{muscle}, and (3) other tissues T_{other} such that $T_{fat}<T_{muscle}<T_{other}$. Pixels are then classified by applying these thresholds as follows: (1) a pixel is categorized as fat if its intensity is greater than T_{fat} and less than T_{muscle} and (2) a pixel is categorized as muscle if its intensity is greater than T_{fat} and less than T_{other}. All other pixels are considered to not be soft tissue. The resulting pixel-wise categorization of the soft tissues are then simplified by applying an octree scheme and averaging categorizations over larger volumes of the image where the larger volumes are labeled according to the majority vote of the categorized pixels that they contain. The final result is a coarse covering of the soft tissue regions within the registered intact image with cubes where each cube includes a categorization of the tissue it contains as either fat (yellow) or muscle (red) figure (33). A 3D model of these tissue elements, $\mathbf{S}$, is generated by placing rigid cylinders centered at each cube center figure (33) . The positions of these estimated

elements represent the initial state of the soft tissue for the intact limb at t_0 as $\mathbf{S}(t_0) = \{\mathbf{T}\}_j(t_0)\mathbf{S}$.

In step (2), physical attributes are assigned for the generated elements of the soft tissue to give them physical meaning for the simulation. Physical attributes for soft tissue are: (1) a density for fat elements, (2) a coefficient of friction for fat elements, (3) a density for muscle elements, (4) a coefficient of friction for muscle elements, and (5) breakable constraints. The density and a coefficient of friction attributes for fat and muscle elements determine the mass of the 3D fat and muscle objects in the intact limb model and how energy is dissipated as these objects contact with each other due to motion, respectively. The breakable constraints attribute serves to approximate how soft tissue will tear apart when placed under severe forces.

The cylindrical elements of the soft tissue model are connected by a lattice of "breakable constraints." Breakable constraints within the bullet engine specify elastic, spring-mass, connections between the soft tissue elements that constrain their joint movement figure (34). These constraints can "break" when the force exerted on the connection exceeds the connection bond strength. Five properties are needed to specify the breakable non-rigid constraints figure (34) : (1) damping, (2) linear limit, (3) angular limit, (4) stiffness, and (5) breaking threshold. Damping property specifies the percentage decrease in the energy stored in the oscillation of a spring per second. Damping value of 0 means no loss in energy, while 1 means a total loss. Linear limit property specifies the amount of translation allowed along each coordinate axis. Angular limit property specifies the amount of rotation allowed around each coordinate axis. Stiffness property specifies how bendy the spring is. A low value creates a very

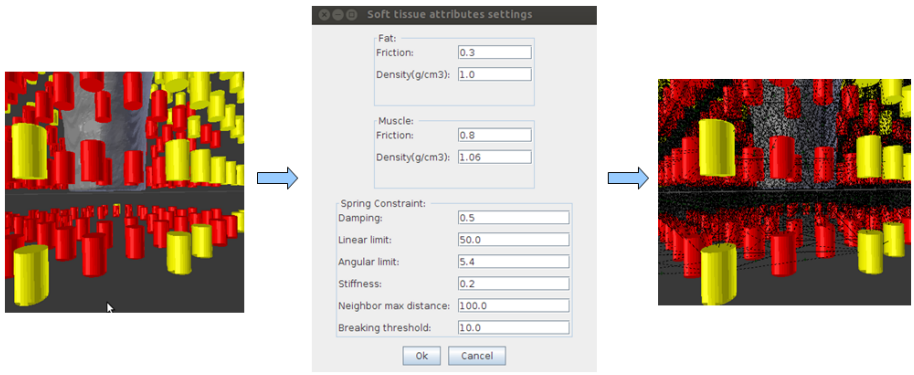


Figure 34: A screen snapshot of the interface window for the settings used to specify physical attributes for soft tissues. There are three groups of attributes: (1) fat, (2) muscle, and (3) spring constraints. These physical attributes control the behavior of soft tissues objects in a fracture simulation.

weak spring, while a high value creates a strong spring. Breaking threshold property specifies the impulse strength that needs to be reached before a constraint breaks. These properties are specified by a user through the system interface to reflect the inner stretching, bending, tension characteristics of soft tissues.

The end result of the image processing functions and segmentation algorithm described in this section is a geometric estimate for the soft tissue surrounding the intact bone, $\mathbf{S}(t_0)$, which is further subdivided into elements of fat or muscle. These elements combine with the bone fragment ones, i.e., $\mathbf{B}(t_0)$, estimated in section (5.2.1) to form the estimate of the geometry intact limb needed to simulate the fracture event.

5.2.3 Generating an Attributed Model for Strike Object

The generation process for an attributed model for the strike object requires a user input. The strike object geometry and physical attributes are typically unavailable for measurement at the point of treatment. The prototypical system requires the user

to specify a geometry, position $\mathbf{p}$, orientation, and physical attributes for the strike object. Let O denote the strike object. The position and orientation of the strike object at time t are denoted as shown in equation (27).

$$O(t) = \mathbf{T}_O(t)O \tag{27}$$

Where $\mathbf{T}_O(t)$ is the transformation matrix for the strike object at time t. The physical attributes include: mass m, velocity v, and direction $\mathbf{d}$. Mass is the amount of material enclosed by the surface of the object. Velocity is the distance traveled per unit of time. Direction specifies the course along which the strike object moves. This information provides an initial point within the fracture event space, i.e., this provides sufficient data to perform a fracture simulation.

5.3 Simulating Fracture Events

Simulating fracture events is the second step in the system of estimating a fracture inverse mechanics. This step takes four inputs: (1) an estimate set of attributed models for fracture fragments $B(t_0)$, (2) an estimate set of attributed models for soft tissues $S(t_0)$, (3) an estimate attributed model for a strike object O, and (4) measured data for bone fracture fragments $\mathcal{D}$. The output of this step is a collection of fracture event simulations where each simulation is ranked by a likelihood score. Each simulation score indicates the likelihood of the fracture image data given the assumed value for the fracture event variables. This step implements a search procedure that varies the user-specified variables of the fracture event to find values for these variables that are supported by the recorded fracture image data. Fracture event simulations

seek to approximate the physical process that occurred when the injury was sustained.

The search procedure seeks to find the values of the fracture event variables, $\mathbf{\Theta}$, that are most likely to have produced the fracture event. These values provide the most accurate description of the measured positions for the fracture fragments observed in the fracture event, in terms of how closely the estimated positions of the fracture fragments in a simulation fit the measured data. These values maximize the likelihood equation (23) and are defined as in equation (28).

$$\hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\arg\max} \left[p(\mathbf{B}|\mathbf{\Theta}, \mathbf{B}(t_f)) \right] = \underset{\mathbf{\Theta}}{\arg\max} \left[k * \exp\left(-\frac{1}{2} \left\| \mathbf{B} - \mathbf{B}(t_f|\Theta) \right\|^2 \right) \right]. \tag{28}$$

Where $\hat{\mathbf{\Theta}}$ is the maximum likelihood estimator. $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$ is the difference between the measured positions for the fracture fragments observed in the fracture event $\mathbf{B}$ and the estimated positions of the fracture fragments in a simulation $\mathbf{B}(t_f|\Theta) = \mathbf{T}_i(t_f|\,\Theta)\mathbf{B}$. $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$ is computed as the average amount of translation that is needed to displace bone fragments from their estimated fracture positions in a fracture simulation to their measured positions observed in the fracture image, refer to equation (29).

$$\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\| = \frac{1}{K} \sum_{i=1}^{K} \|\mathbf{B}_i - \mathbf{B}_i(t_f|\Theta)\|. \tag{29}$$

Where K is the number of bone fragments. The data for bone fragments are positions so that the rotation and intensity differences are ignored in $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$ computation. This difference of data, i.e., $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$, is referred to as a translation error. Let $\mathbf{B}_i = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2 ..., \mathbf{x}_M\}$ be the measured data for the i^{th} bone fragment and $\mathbf{x}_m$ be the (x, y, z) coordinate in the image for the m^{th} point in bone fragment

$\mathbf{B}_i$. Then, $\|\mathbf{B}_i - \mathbf{B}_i(t_f|\Theta)\|$ is computed as the average amount of translation that is needed to displace the transformed points in $\mathbf{B}_i(t_f|\Theta)$ to their measured positions as defined in equation (30).

$$\|\mathbf{B}_i - \mathbf{B}_i(t_f|\Theta)\| = \frac{1}{M}\sum_{m=0}^{M} ||\mathbf{x}_m - \mathbf{T}_i(t_f|\Theta)\mathbf{x}_m||. \tag{30}$$

Where $\mathbf{T}_i(t_f|\Theta)$ is the data transformation computed for i^{th} bone fragment, $\mathbf{B}_i$, by the fracture simulation at the final time for the fracture event. $||\cdot||$ is the norm of a vector. $\mathbf{T}_i(t_f|\Theta)\mathbf{x}_m$ is equal to $\mathbf{x}_m$ only and only if the data transformation $\mathbf{T}_i(t_f|\Theta)$ is the identity matrix. So, the desired fracture simulation is the one that is able to move bone fragment pixels from their estimated original anatomic positions at t_0, i.e., $\mathbf{T}_i(t_0|\Theta)\mathbf{x}_m$, and to put them back to their measured positions. Such a simulation has the highest likelihood $\mathbf{\Theta}$.

By applying the natural log, equation (28) is rewritten as in equation (31) since the maxima of $p(\mathbf{B}|\mathbf{\Theta}, \mathbf{B}(t_f))$ are equivalent to the maxima of $\ln p(\mathbf{B}|\mathbf{\Theta}, \mathbf{B}(t_f))$.

$$\arg\max_{\mathbf{\Theta}} \left[\ln p(\mathbf{B}|\mathbf{\Theta}, \mathbf{B}(t_f))\right] = \arg\max_{\mathbf{\Theta}} \left[\ln k - \left(\frac{1}{2}\left\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\right\|^2\right)\right]. \tag{31}$$

Applying the natural log function in this context is helpful for three reasons. First, logs reduce the potential for underflow in numerical analysis due to very small likelihoods. Second, logs allow converting a product of factors into a summation of factors. Third, the natural log function is a monotone transformation that does not change the locations for maxima or minima. In equation (31), $\ln k$ and $\frac{1}{2}$ are shift and scale factors, respectively, that do not affect the location of maxima, so they are removed

from the computation as in equation (32).

$$\arg\max_{\boldsymbol{\Theta}} \left[\ln p(\mathbf{B}|\boldsymbol{\Theta}, \mathbf{B}(t_f))\right] = \arg\max_{\boldsymbol{\Theta}} \left[-\left\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\right\|^2\right]. \tag{32}$$

Equation (32) is rewritten as in equation (33) since $\arg\max_x(-x) = \arg\min_x(x)$. This form is helpful to get rid of the negative sign in equation (32).

$$\arg\max_{\boldsymbol{\Theta}} \left[-\left\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\right\|^2\right] = \arg\min_{\boldsymbol{\Theta}} \left[\left\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\right\|^2\right]. \tag{33}$$

Equation (32) is further reduced to equation (34) since $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$ is a non-negative value and the square function is monotonic in the interval of non-negative values, i.e., $x \geq 0$. So that $\arg\min_x(|x|^2) = \arg\min_x(|x|)$.

$$\hat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} \left[\left\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\right\|^2\right] = \arg\min_{\boldsymbol{\Theta}} \left[\left\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\right\|\right] \tag{34}$$

This form is interesting since it directly connects the value of $\hat{\boldsymbol{\Theta}}$ to the translation error, i.e., the difference between estimated fracture positions for fragments in a fracture simulation and the measured fracture positions observed in the fracture image.

The search procedure implements a numerical optimization algorithm to find the MLE estimates for Θ. The goal for the optimization algorithm is to quickly find a list of plausible optimal values that maximize the log-likelihood [55]. This goal is achieved by searching smaller sub-sets of the 8-dimensional space instead of exhaustively searching the whole space, which becomes intractable as the number of variables increases. The search proceeds by selecting different starting values for Θ from the 8-dimensional space. For each starting value, the search proceeds by means

of trial and error over the course of a series of iterative steps. On each iteration, a new set of variable values is obtained by adding small changes to the previous variables, computed in the previous iteration, in such a way that the new variables are likely to lead to better estimate for Θ. The algorithm uses the gradient descent method in updating these variables. This iterative process continues until predefined criterion are achieved. The stopping criterion include the maximum number of iterations allowed and the minimum amount of change in variable values between two successive iterations. The most likely values of Θ are then selected as plausible solutions for the fracture event. This optimization algorithm is practical since it is not possible in practice to obtain an analytic form solution for the MLE estimate in a model that involves so many parameters as the one in this system.

The 8-dimensional volume of the solution space is sampled to select different starting values for $\mathbf{\Theta}$ in the optimization algorithm. The multi-scale aspect of the algorithm defines an 8-dimensional volume by bracketing each of the 8 components of the fracture variables to the subset of values which are physically meaningful. For example, the direction vector must be constrained to point in the direction of the fracture site and the velocity has to be above a minimum speed and below some maximum speed. Within this 8-dimensional volume samples are taken at locations along a rectangular grid.

For each starting value for $\mathbf{\Theta}$, an iterative process is performed to find the most likely value for $\mathbf{\Theta}$. This process uses the gradient descent method to update the values of $\mathbf{\Theta}$ in each iteration by taking into account the results from the previous iteration. The iterative process is performed in a sequence of four steps:

1. Run a fracture simulation using Θ,

2. Compute the translation error, i.e.,$\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$,

3. Update the values of Θ using the gradient descent method,

4. Repeat steps (1 - 3), until the stopping criterion are achieved.

The listed sequence describes a process performs an iterative gradient descent minimization to find a set of MLE values for $\mathbf{\Theta}$.

In step (1), the search procedure takes the current value of the fracture event variables $\mathbf{\Theta}$ and runs a fracture simulation. When the simulation ends, the system provides transformation data $\hat{\mathbf{T}}$ for for all virtual objects, i.e., bone fragments, soft tissue, and strike object, at each keyframe of the simulation. The collection of these objects with their transformation data at a specific keyframe form the fracture context at that keyframe. Let $FC[n \,|\, \Theta]$ denote the fracture context at keyframe n given simulation variables Θ, then, the fracture contexts for a fracture simulation are defined as in equation (35).

$$FC[n \,|\, \Theta]|_{n\in[0,N]} = \{\{\hat{\mathbf{T}}\}_i[n|\,\Theta]\mathbf{B}, \{\hat{\mathbf{T}}\}_j[n|\,\Theta]\mathbf{S}, \hat{\mathbf{T}}_O[n|\,\Theta]O\}. \tag{35}$$

Where N is the number of keyframes recorded by the fracture simulation.

In step (2), the value for $\{\hat{\mathbf{T}}\}_i$ is used to compute the translation error $\|\mathbf{B} - \mathbf{B}[N|\Theta]\|$ as in equation (29) for the guessed $\mathbf{\Theta}$ value. The computed translation error represents the difference between the estimated fracture positions for fragments in a fracture simulation at the last keyframe , i.e., $\mathbf{B}[N \,|\, \Theta]$, and the measured fracture positions observed in the fracture image, i.e., $\mathcal{D} = \mathbf{B}$.

In step (3), the search process updates the values of Θ to minimize $\|\mathbf{B} - \mathbf{B}[N|\Theta]\|$ using the gradient descent method [25]. Let $TE(\Theta) = \|\mathbf{B} - \mathbf{B}[N|\Theta]\|$ be a function of Θ that denotes the translation error computed for the fracture simulation generated using Θ. Then, the values of Θ are updated as in equation (36).

$$\Theta^{r+1} = \Theta^r - \gamma \nabla_\Theta TE(\Theta^r). \tag{36}$$

Where Θ^r is the simulation variables at the r^{th} iteration. $\nabla_\Theta TE(\Theta^r)$ is the gradient with respect to Θ for the computed translation error in equation (29) for the r^{th} iteration. γ is the step size. The gradient descent method finds a local minimum of a function by taking steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point.

The gradient of the translation error $\nabla_\Theta TE(\Theta^r)$ is computed numerically. For this system, $\nabla_\Theta TE(\Theta^r)$ is an 8-dimensional vector defined as in equation (37).

$$\nabla_\Theta TE(\Theta^r) = [\frac{\partial TE(\Theta^r)}{\partial v_1}, \frac{\partial TE(\Theta^r)}{\partial v_2}, ..., \frac{\partial TE(\Theta^r)}{\partial v_8}]^\top. \tag{37}$$

Where $\frac{\partial TE(\Theta^r)}{\partial v_m}$ is the derivative of the translation error with respect to the m^{th} variable in Θ, i.e., v_m. Let δ be a small positive number that represents the step

length and e_m be the m^{th} standard basis vector as defined in equation (38).

$$e_m = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{-row number } m \tag{38}$$

Then, the partial derivative of $TE(\Theta^r)$ with respect to the m^{th} variable is approximated as in equation (39).

$$\frac{\partial TE(\Theta^r)}{\partial v_m} \approx \frac{TE(\Theta^r + \delta e_m) - TE(\Theta^r)}{\delta}. \tag{39}$$

Where $(\Theta^r + \delta e_m)$ is the incremental step associated with the m^{th} variable. Note that, in a fracture simulation, the measured fragment data $\mathbf{B}$ is fixed and does not change over time with respect to Θ^r. The transformation data $\hat{\mathbf{T}}_i[N|\,\Theta]$ in $\mathbf{B}[N|\Theta^r] = \hat{\mathbf{T}}_i[N|\,\Theta]\mathbf{B}$ is the only variable that changes with respect to Θ^r in a fracture simulation. $TE(\Theta^r + \delta e_m)$ is evaluated by running a fracture simulation using the incremented simulation variables $(\Theta^r + \delta e_m)$, then, at the end of the fracture simulation, the translation error is computed as in equation (29). As a note, the step length δ should be small, but not too small.

In step (4), the iteration process in steps (1 - 3) continues until predefined criterion are achieved. The stopping criterion include the maximum number of iterations

allowed and the minimum amount of change in variable values between two successive iterations. the likelihood score reaches its minimum. The maximum number of iterations, denoted as R, is specified by the user. The iteration process stops when $r \geq R$, where r is an iteration counter. The minimum amount of change in variable values between two successive iterations is measured by the norm of the gradient vector $\nabla_{\Theta} TE(\Theta^r)$ computed in equation (37). The iteration process stops when the norm of that gradient vector is very small, i.e.,$\|\nabla_{\Theta} TE(\Theta^r)\| < \epsilon$, where ϵ is a small positive quantity. When on of these criterion is achieved, the system finishes the current iteration process and stores the simulation variables Θ that corresponds to the smallest translation error, i.e., $TE(\Theta) = \|\mathbf{B} - \mathbf{B}[N|\Theta]\|$, obtained. For each stored Θ a likelihood score is computed as in equation (40).

$$LKS(\Theta) = -c_1 * \ln\left[c_2 * \|\mathbf{B} - \mathbf{B}[N|\Theta]\| + 1\right]. \tag{40}$$

Where $LKS(\Theta)$ is the likelihood score computed for Θ and c_1 and c_2 are positive scale factors. The natural log is a monotonic function that preserves the order of an ordered set of values. The negative of the natural log is taken so that when the translation error $TE(\Theta)$ is small the likelihood score is high and vice versa.

After processing all the sampled points of the simulation variables space, the search procedure collects the simulation variables with the top likelihood scores and returns them as a list of plausible solutions for the fracture event observed in the provided CT images. The search procedure arranges the stored simulation variables Θ for the different sample points in an descending order according to the value of their corresponding $LKS(\Theta)$. So that, the first one is the most likely simulation variables

that generate a fracture pattern similar to that measured in the fracture image. While, the last one is the least likely simulation variables that generate a fracture pattern similar to that measured in the fracture image.

5.3.1 Visualizing and Analyzing Fracture Simulations

Visualizing and analyzing fracture simulations is the third step in the system of estimating a fracture inverse mechanics. This step takes two inputs: (1) the list of Θs found in section (5.3) and (2) fracture image I_f. In this step, the system provides two tools: (1) a tool to visualize virtually how bone fragments moved in space during a fracture event and (2) a tool to visualize virtually the void that is created in soft tissue due to the movement of bone fragments. These tools allow a user to explore the space of plausible solutions for a fracture event.

The movement of bone fragments is visualized by animating the surfaces for the virtual fragment objects over time. The animation process creates the motion by rapidly displaying a sequence of changes to the position and orientation for each bone fragment. This sequence of changes uses the computed transformation data in all keyframes produced by the fracture simulation that is generated for Θ of interest. Each keyframe corresponds to a point of time in a fracture event. The computed transformation data for that keyframe specify the estimated position and orientation for all bone fragments at that point of time. Let n be the index of the keyframe to be displayed, then the displayed bone fragments are $\mathbf{B}[n] = \hat{\mathbf{T}}_i[n|\,\Theta]\mathbf{B}$.

The void in soft tissue is visualized in two ways: (1) by displaying a 3D virtual surface for the void region and (2) by highlighting the void region on the fracture

image. These visualizing ways allow the user to estimate the location and shape of the void region.

The void region in soft tissue corresponds to the estimated soft tissue areas where bone fragments passed through during their movements in a fracture event. This region is computed by intersecting the soft tissue positions on the fracture image at the start of a fracture simulation with all positions on the fracture image for bone fragments in all keyframes in a fracture simulation. This region is defined as in equation (41).

$$\mathbf{V} = \mathbf{S}(0) \bigcap \left\{ \bigcup_{n \in [0,N]} \mathbf{B}[n] \right\}. \tag{41}$$

Where $\mathbf{V}$ is the computed void region. This region does not include the positions of bone fragments at the start of a fracture event since these positions are not part of the soft tissue $\mathbf{S}(0)$, i.e., they will be excluded by the intersection operation.

The 3D representation for the void region is generated by triangulating the surface of this region. The 3D representation is displayed along side the bone fragments at their reduced positions, i.e., their estimated original anatomic positions, see figure (35). This display allows the user to visualize the void shape and position in the fractured limb.

The highlighting of the void region on the fracture image is generated by displaying the image pixels that are located inside this region with a unique color. By navigating through the image slices, the user sees the void pixels highlighted in each slice, see figure (36). This way allows the user to quantify the amount of void region in each

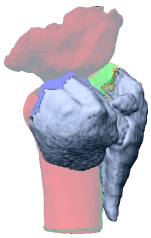


Figure 35: shows a 3D representation of the estimated void region in soft tissue. The void region in soft tissue surface is shown (in solid light blue) with respect to other bone fragment surfaces (in faded colors) in their estimated original anatomic positions in order to visualize the shape and location of void region in soft tissue in the fractured ankle.

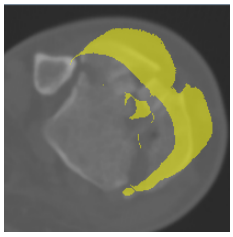


Figure 36: shows void region pixels on one of the 2D slices of the fracture image. The image pixels that are located inside the void region are displayed with a unique color in the slice. The number of pixels inside the void region is reported for each slice. This way allows the user to quantify the amount of void region in each slice and to estimate the start and end of the void region in the fractured limb.

slice and to estimate the start and end of the void region in the fractured limb.

These presented tools allow the user to explore the space of plausible solutions computed for a fracture event. Animating bone fragments shows how these fragments moved over time during the fracture event. While, displaying the void region in soft tissue approximate the trajectory of the bone fragments through the soft tissue. This void region may provide clues about the location and extent of soft tissue trauma in the fractured limb.

CHAPTER 6: RESULTS OF COMPUTING THE MECHANICS OF A HIGHLY COMMINUTED TIBIA FRACTURE

This chapter discusses results of experiments for the system of computing the inverse mechanics of a highly comminuted tibia fracture. The experiments were performed to find the estimated plausible solutions for a fracture event and visualize them. The visualization shows how bone fragments moved over time during the fracture event and shows the void created in soft tissue. Generated fracture simulations are visually inspected and the errors are computed to conclude the results.

The system was used to estimate the fracture event of a clinical tibial plafond case by following the interactive steps discussed in chapter (4). For this case, the fracture image, the intact image, the bone fracture fragment surfaces, the intact bone surface, and the reconstructed fracture fragment positions were provided by the work in [46], which virtually reconstructs highly comminuted tibia fragments from 3D CT images of a fracture case. The fracture case includes six bone fragments and also includes a model for the talus bone. The images are 3D CT scans in DICOM format and 16-bit are used to express the CT numbers.

The experiments were conducted to generate three fracture simulations for three different shapes of a strike object while other system settings are held constant. All experiments were conducted with the following collection of values for different parameters and settings of the system. The settings that are used to generate soft

tissue attributed models are: fat threshold T_{fat} = -400HU, muscle threshold T_{muscle} = -100HU, bone threshold T_{bone}= 100HU, fat density = $0.9g/cm^3$ [29], fat friction = 0.3, muscle density = $1.06g/cm^3$ [81], and muscle friction = 0.8. The breakable constraints settings are: damping factor = 0.5, linear limit = 5mm, angular limit = 5 degrees, stiffness = 0.2, and breaking threshold = 1. The settings for physical properties of fracture fragment attributed models are: bone density = $1.85g/cm^3$ [95] and friction = 0.6. Fracture simulations were set to run to a final time of 1.5 seconds and the maximum number of search iterations allowed to find the best fracture simulation is set to 100. The system solved for likely values of the unknown fracture event variables. These variables control the impact of the strike object and determine its: speed v, mass m, scale s, 3D position $\mathbf{p}$, and direction $\mathbf{d}$. At the end of each experiment, the list of most likely $\hat{\Theta}$ values is provided with their likelihood score. The likelihood score is computed according to equation (40) with $c_1 = 100$ and $c_2 = 0.001$. These scale factor values are used since the translation error $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$ is computed in mm. The listed values of $\hat{\Theta}$ are used to produce the plausible fracture simulations for how the fracture event occurred.

Each experiment is presented in a separate section, sections (6.1 to 6.3). These sections describe the unique aspects of each experiment and adhere to a fixed structure for clarity. This structure consists of seven figures, two tables, and a discussion. The discussion details some information in these figures and tables and also mentions the distinctive aspects of the experiment using the system. The following list defines the elements in each experiment analysis section in detail:

1. Strike Object: This figure shows a snapshot of the strike object that is used to virtually hit the virtual model of the fractured limb,

2. Axial Simulation View: This figure shows four snapshots from the axial view perspective for the fracture simulation generated using the values of the most likely $\hat{\Theta}$. These snapshots are taken at four even samples during the simulation time interval.

3. Coronal Simulation View: This figure shows four snapshots from the coronal view perspective for the fracture simulation generated using the values of the most likely $\hat{\Theta}$. These snapshots are taken at four even samples during the simulation time interval,

4. Saggital Simulation View: This figure shows four snapshots from the saggital view perspective for the fracture simulation generated using the values of the most likely $\hat{\Theta}$. These snapshots are taken at four even samples during the simulation time interval,

5. CT Images: This figure shows two slices of the 3D CT fracture image, slice 80 and slice 108, with the contours of the fractured tibia fragments. These contours represent the intersection of the specified slices with the fragment surfaces. The fragments are at their final fracture position estimated by the fracture simulation generated using the values of the most likely $\hat{\Theta}$.

6. Soft Tissue Void: This figure shows a 3D representation of the estimated void region created in soft tissue due to the movement of the fracture fragments.

7. Soft Tissue Void Chart: This chart shows the amount of void region in pixels per slice on the fracture image.

8. List of $\hat{\Theta}$ Values: This table consists of the top three likely values for $\hat{\Theta}$. These values produce fracture simulations which generate fracture patterns that are most similar to that measured in the fracture image,

9. List of Translation Error Per Fracture Bone Fragment: This table consists of the average amount of translation, i.e., $\|\mathbf{B}_i - \mathbf{B}_i(t_f|\Theta)\|$, that is needed to displace the transformed points in $\mathbf{B}_i(t_f|\Theta)$ to their measured positions as defined in equation (30).

The results of the experiments are discussed in a separate section (6.4). The discussion details some information about the result and the distinctive aspects of the generated fracture simulations using the system.

6.1 Experiment-1: Spherical Strike Object

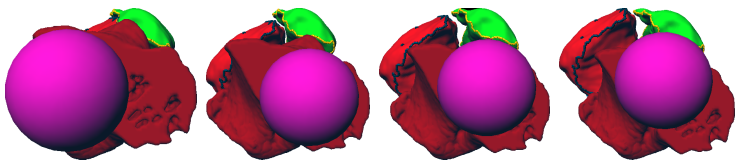
This section presents the experimental result for estimated fracture simulations for a fracture event that are generated using a spherical strike object. Figure (37) shows the shape for the spherical strike object. The diameter of the sphere is 20mm. Table (1) shows the top three possible values for $\hat{\Theta}$ that are highly likely to generate a fracture pattern similar to that measured in the fracture image. The values are sorted in a descending order according to the likelihood score value. Figures (38-40) show snapshots for the fracture simulation that is generated by the top value of $\hat{\Theta}$ from three different view perspectives: axial, coronal, and sagittal. Each figure consists of four snapshots captured at four evenly spaced time samples for the fracture simulation,

(a)

Figure 37: shows strike object shape for experiment-1. The strike object shape is a sphere with a diameter of 20mm. This shape of the strike object is used to hit virtually the virtual limb model of the fracture event.

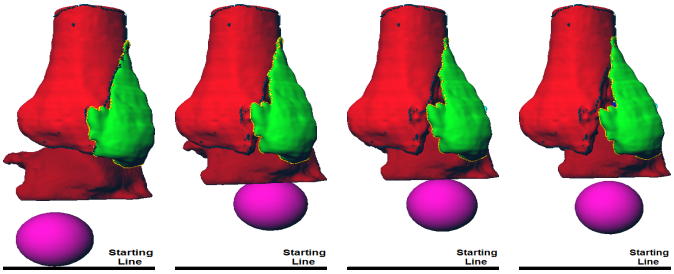
Table 1: List of the top three possible values for $\hat{\Theta}$ for a spherical strike object that are highly likely to generate a fracture pattern similar to that measured in the fracture image. Each row in the table, show the values of the variables in $\hat{\Theta}$ as well as the reported likelihood score. The values for $\hat{\Theta}$ are sorted in an descending order with respect to the likelihood score.

$\hat{\Theta}$	m (g)	s	v (mm/s)	**d** (theta,phi) in rad	**p** (x,y,z)	LKS($\hat{\Theta}$)
1	158	1.5	111	(2.18,-.174)	(32,41,84)	-5.9
2	152	.98	107	(2.06,0.0)	(35,44,88)	-6.0
3	199	1.4	72	(2.0,-.611)	(34,36,87)	-6.3



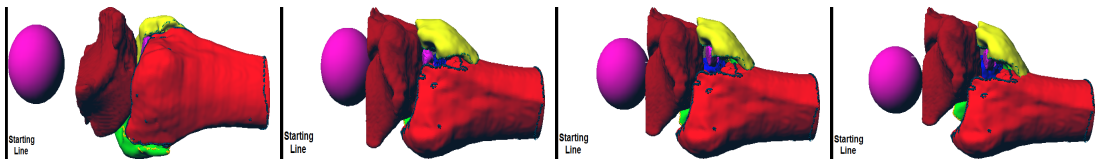
(a) (b) (c) (d)

Figure 38: shows axial view snapshots for experiment-1 of the fracture simulation for a spherical strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.



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

Figure 39: shows coronal view snapshots for experiment-1 of the fracture simulation for a spherical strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.



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

Figure 40: shows saggital view snapshots for experiment-1 of the fracture simulation for a spherical strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.

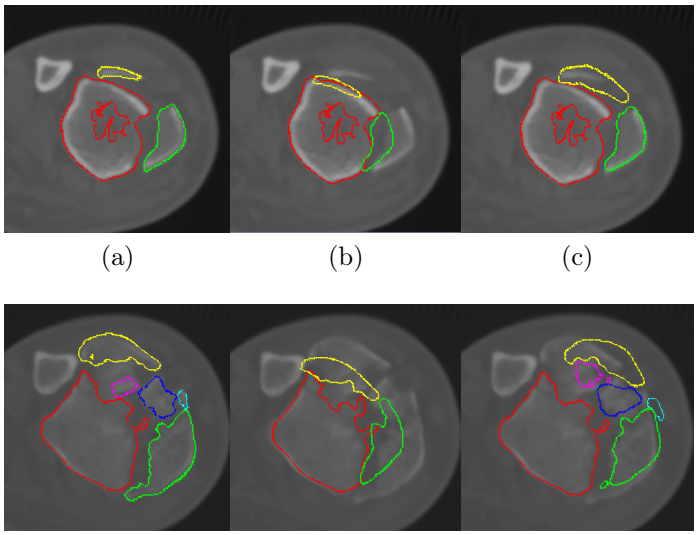


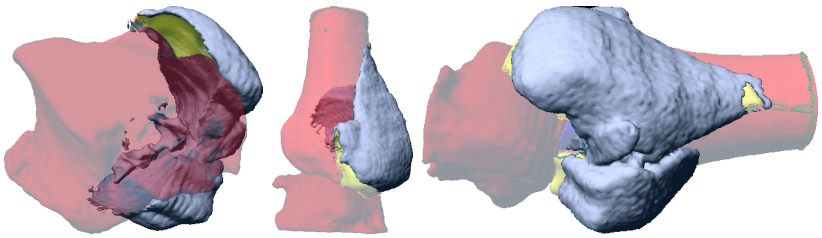
Figure 41: shows contours of the six fractured fragments measured from the CT image, reduced fragments, and fracture fragments at the end of the simulation for the first value of $\hat{\Theta}$. The contours are shown at two different slices of the CT image: slice 80 (top row) and slice 108 (bottom row). Slice 80 shows contours for three of the six fragments, while slice 108 shows the contours for all six fragments.

i.e., $t = \{0\text{s}, 0.5\text{s}, 1\text{s}, 1.5\text{s}\}$. These figures show the most likely plausible solution obtained for how bone fragments moved from their original anatomic positions to their fractured ones in a fracture event.

Figure (41) shows contours of the fractured fragments measured from the CT image, reduced fragments, and fracture fragments at the end of the simulation for the first value of $\hat{\Theta}$. The contours are shown at two different slices of the CT image: (1) slice 80 that shows the contours for three fragments and (2) slice 108 that shows the contours for all six fragments. This figure is provided to compare visually the positions and orientations of the fragments computed via the simulation with their measured ones within the fracture CT image. Table (2) shows the translation error for each fragment. Fragment 1 has a 0 error since it is kept fixed during the simulation.

Table 2: shows the computed translation error for each fragment $\|\mathbf{B}_i - \mathbf{B}_i(t_f|\Theta)\|$ for the most likely value for $\hat{\Theta}$ for a spherical strike object to generate a fracture simulation. The first row indicates the fragment ID while the second row reports the translation error for each fragment. The color code in the first row indicate the color of the corresponding fragment contour in figure (41). Fragment 1 has a 0 error since it is kept fixed during the simulation. This bone represents the base bone that is used in the reconstruction system.

Fragment ID	1(Red)	2 (Green)	3 (Yellow)	4(Blue)	5 (Pink)	6 (Turquoise)
$\|\mathbf{B}_i - \mathbf{B}_i(t_f\|\Theta)\|$(mm)	0	19	119	112	96	22



(a) (b) (c)

Figure 42: shows a 3D representation of the estimated void region in soft tissue in Experiment-1. Figures (a-c) show the void region surface in solid light blue from three different view perspectives: axial, coronal, and saggital, respectively. The void region surface is shown with respect to other bone fragment surfaces (in faded colors) in their estimated original anatomic positions in order to visualize its shape and location in the ankle.

This bone represents the base bone that is used in the reconstruction system. The average translation error for all fragments, i.e., $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$, is equal to 61mm as computed in equation (29). This figure and table help the user to understand more the translation error of the simulation result.

Figure (42) shows a 3D representation for the surface of the estimated void in soft tissue due to the movements of fracture fragments during the fracture event. The surface of the void region is shown from three different view perspectives: axial, coronal, and saggital. The surface is drawn with respect to other bone fragment surfaces in their estimated original anatomic positions in order to visualize the shape and location of the void region in the fractured ankle. Figure (43) shows a chart for

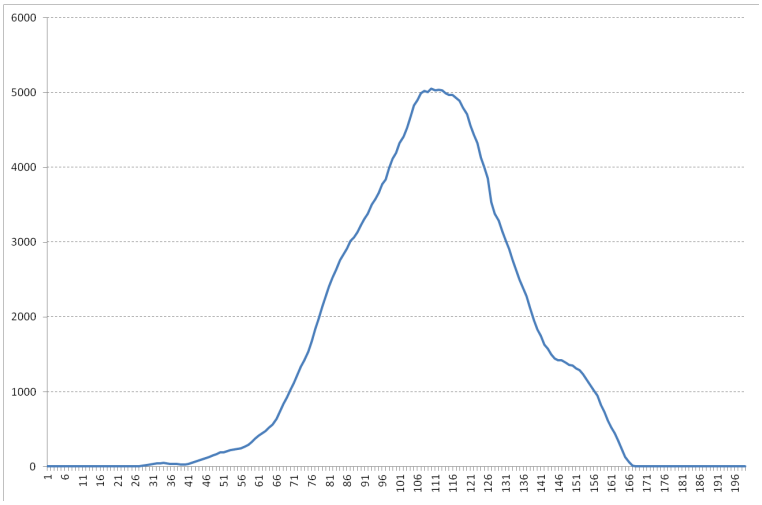
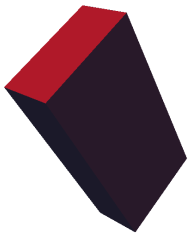


Figure 43: A chart for the void region in soft tissue that is estimated in Experiment-1. The x-axis represents the slice index while the y-axis represents the number of pixels of the void region per slice. These pixels are the ones that the bone fragments passed through during a fracture simulation.

the estimated number of pixels for the void region per slice in the fracture image. This chart provides a quantitative estimate for the void region. The information conveyed by this chart and the 3D representation of the void region in soft tissue may help the user in locating the most damaged region of the soft tissue in the real fractured ankle.

6.2 Experiment-2: 3D Rectangular Strike Object

This section presents the experimental result for estimated fracture simulations for a fracture event that are generated using a 3D rectangular strike object. Figure (44) shows the shape of the 3D rectangular strike object. The dimension of the 3D rectangle (width, height, length) is (10mm, 20mm, 40mm). Table (3) shows the top three possible values for $\hat{\Theta}$ that are highly likely to generate a fracture pattern similar to that measured in the fracture image. The values are sorted in a descending order according to the likelihood score value. Figures (45-47) show snapshots for the fracture simulation that is generated by the top value of $\hat{\Theta}$ from three different

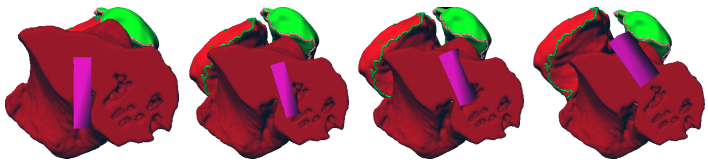


(a)

Figure 44: shows strike object shape for experiment-2. The strike object shape is a 3D rectangle. The dimension of the 3D rectangle (width, height, length) is (10mm, 20mm, 40mm). This shape of the strike object is used to hit virtually the virtual limb model of the fracture event.

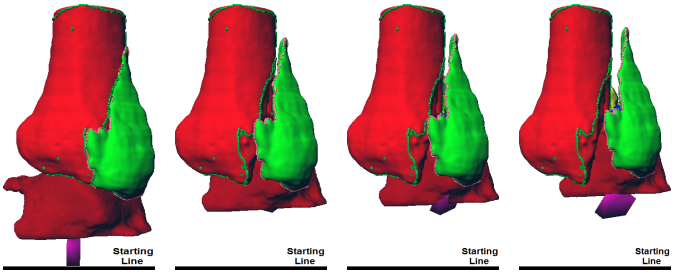
Table 3: List of the top three possible values for $\hat{\Theta}$ for a 3D rectangular strike object that are highly likely to generate a fracture pattern similar to that measured in the fracture image. Each row in the table, show the values of the variables in $\hat{\Theta}$ as well as the likelihood score. The values for $\hat{\Theta}$ are sorted in a descending order with respect to the likelihood score.

$\hat{\Theta}$	m (g)	s	v (mm/s)	**d** (theta,phi) in rad	**p** (x,y,z)	LKS($\hat{\Theta}$)
1	89	0.5	99	(2.356,-.219)	(37,41,87)	-6.2
2	109	.5	101	(2.26,-.03)	(36,39,84)	-7.4
3	94	1.0	71	(2.18,-.17)	(37,39,89)	-9.2



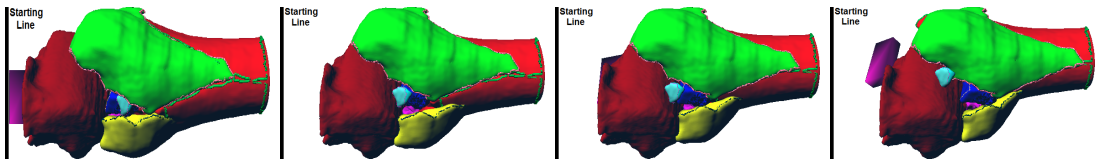
(a) (b) (c) (d)

Figure 45: shows axial view snapshots for experiment-2 of the fracture simulation for a rectangular prism strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.



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

Figure 46: shows coronal view snapshots for experiment-2 of the fracture simulation for a rectangular prism strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.



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

Figure 47: shows saggital view snapshots for experiment-2 of the fracture simulation for a rectangular prism strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.

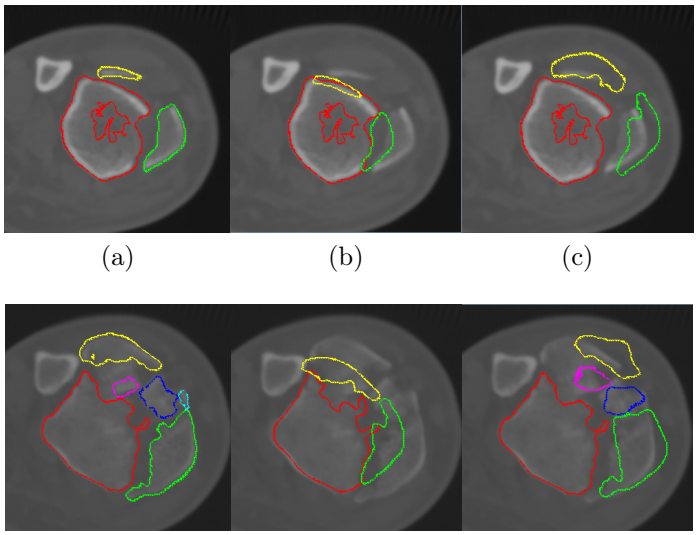


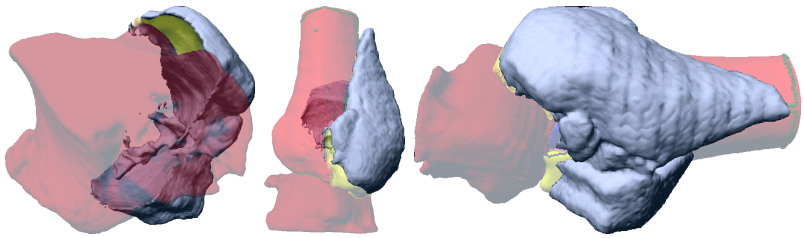
Figure 48: shows contours of the six fractured fragments measured from the CT image, reduced fragments, and fracture fragments at the end of the simulation for the first value of $\hat{\Theta}$. The contours are shown at two different slices of the CT image: slice 80 (top row) and slice 108 (bottom row). Slice 80 shows contours for three of the six fragments, while slice 108 shows the contours for all six fragments.

view perspectives: axial, coronal, and sagittal. Each figure consists of four snapshots captured at four evenly spaced time samples for the fracture simulation, i.e., $t = \{0\text{s}, 0.5\text{s}, 1\text{s}, 1.5\text{s}\}$. These figures show the most likely plausible solution obtained for how bone fragments moved from their original anatomic to their fractured ones in a fracture event.

Figure (48) shows contours of the fractured fragments measured from the CT image, reduced fragments, and fracture fragments at the end of the simulation for the first value of $\hat{\Theta}$. The contours are shown at two different slices of the CT image: (1) slice 80 that shows the contours for three fragments and (2) slice 108 that shows the contours for all six fragments. This figure is provided to compare visually the positions and orientations of the fragments computed via the simulation with their

Table 4: shows the computed translation error for each fragment $\|\mathbf{B}_i - \mathbf{B}_i(t_f|\Theta)\|$ for the most likely value for $\hat{\Theta}$ for a 3D rectangular strike object to generate a fracture simulation. The first row indicates the fragment ID while the second row report the translation error for each fragment. The color code in the first row indicate the color of the corresponding fragment contour in figure (48). Fragment 1 has a 0 error since it is kept fixed during the simulation. This bone represents the base bone that is used in the reconstruction system.

Fragment ID	1(Red)	2 (Green)	3 (Yellow)	4(Blue)	5 (Pink)	6 (Turquoise)
$\|\mathbf{B}_i - \mathbf{B}_i(t_f\|\Theta)\|$(mm)	0	12	126	107	83	56



(a) (b) (c)

Figure 49: shows a 3D representation of the estimated void region in soft tissue in Experiment-2. Figures (a-c) show the void region in soft tissue in solid light blue from three different view perspectives: axial, coronal, and saggital, respectively. The void region surface is shown with respect to other bone fragment surfaces (in faded colors) in their estimated original anatomic positions in order to visualize the shape and location of the void region in soft tissue in the ankle.

measured ones within the fracture CT image.Table (4) shows the translation error for each fragment. Fragment 1 has a 0 error since it is kept fixed during the simulation. This bone represents the base bone that is used in the reconstruction system. The average translation error for all fragments, i.e., $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$, is equal to 64mm as computed in equation (29). This figure and table help the user to understand more the translation error of the simulation result.

Figure (49) shows a 3D representation for the surface of the estimated void region in soft tissue due to the movements of fracture fragments during the fracture event. The surface of the void region in soft tissue is shown from three different view perspectives: axial, coronal, and saggital. The surface is drawn with respect to other bone fragment

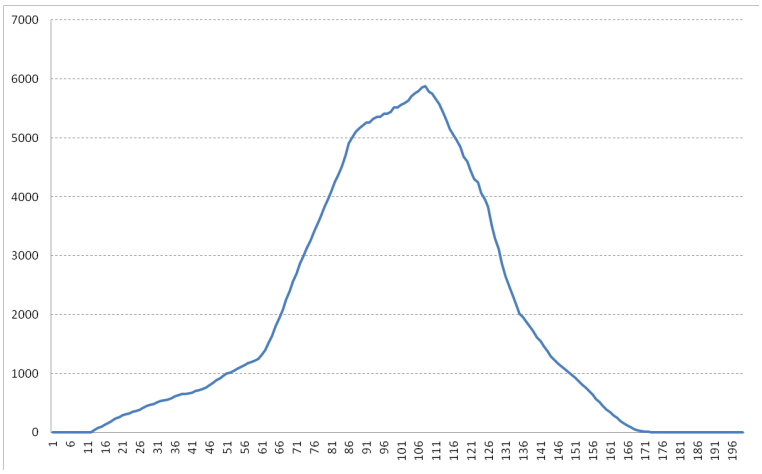


Figure 50: A chart for void region in soft tissue that is estimated in Experiment-2. The x-axis represents the slice index while the y-axis represents the number of pixels of void region per slice. These pixels are the ones that the bone fragments passed through during a fracture simulation.

surfaces in their estimated original anatomic positions in order to visualize its shape and location of the void region in the fractured ankle. Figure (50) shows a chart for the estimated number of pixels for the void region per slice in the fracture image. This chart provides a quantitative estimate for the void region in soft tissue. The information conveyed by this chart and the 3D representation of the void region in soft tissue may help the user in locating the most damaged region of the soft tissue in the real fractured ankle.

6.3 Experiment-3: Cylindrical Strike Object

This section presents the experimental result for estimated fracture simulations for a fracture event that are generated using a rectangular cylindrical strike object. Figure (51) shows the shape of the cylindrical strike object. The dimension of the cylinder (diameter, length) is (10mm, 40mm). Table (5) shows the top three possible values for $\hat{\Theta}$ that are highly likely to generate a fracture pattern similar to that measured

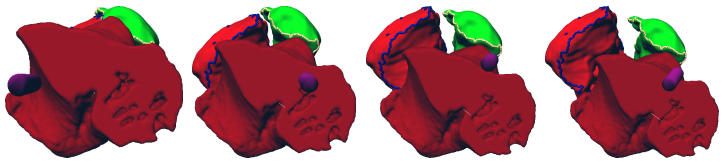
(a)

Figure 51: shows strike object shape for experiment-3. The strike object shape is cylinder. The dimension of the cylinder (diameter, length) is (10mm, 40mm). This shape of the strike object is used to hit virtually the virtual limb model of the fracture event.

Table 5: List of the top three possible values for $\hat{\Theta}$ for a cylindrical strike object that are highly likely to generate a fracture pattern similar to that measured in the fracture image. Each row in the table, show the values of the variables in $\hat{\Theta}$ as well as the likelihood score. The values for $\hat{\Theta}$ are sorted in a descending order with respect to the likelihood score.

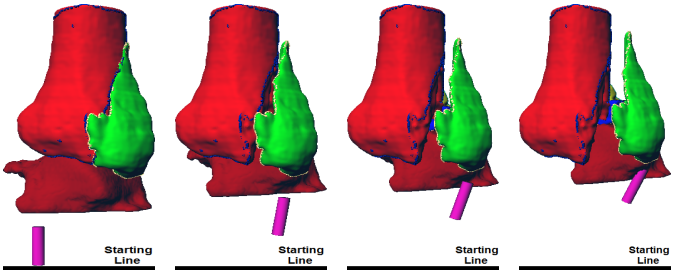
$\hat{\Theta}$	m (g)	s	v (mm/s)	**d** (theta,phi) in rad	**p** (x,y,z)	LKS($\hat{\Theta}$)
1	79	0.5	80	(2.01,-.17)	(42,38,89)	-6.4
2	94	.57	99	(2.01,.00)	(40,42,89)	-7.2
3	79	0.86	89	(2.35,-.08)	(42,39,86)	-10.1

in the fracture image. The values are sorted in a descending order according to the likelihood score value. Figures (52-54) show snapshots for the fracture simulation that is generated by the top value of $\hat{\Theta}$ from three different view perspectives: axial, coronal, and sagittal. Each figure consists of four snapshots captured at four evenly spaced time samples for the fracture simulation, i.e., $t = \{0\text{s}, 0.5\text{s}, 1\text{s}, 1.5\text{s}\}$. These figures show the most likely plausible solution obtained for how bone fragments moved from their original anatomic to their fractured ones in a fracture event.



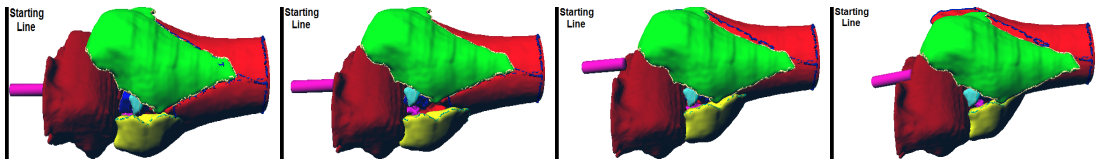
(a) (b) (c) (d)

Figure 52: shows axial view snapshots for experiment-3 of the fracture simulation for a cylindrical strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.



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

Figure 53: shows coronal view snapshots for experiment-3 of the fracture simulation for a cylindrical strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.



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

Figure 54: shows saggital view snapshots for experiment-3 of the fracture simulation for a cylindrical strike object at four different keyframes. The time interval for the fracture simulation is 1.5s. Figures (a to d) show snapshots of the fracture simulation at 0s, 0.5s, 1.0s, and 1.5s, respectively.

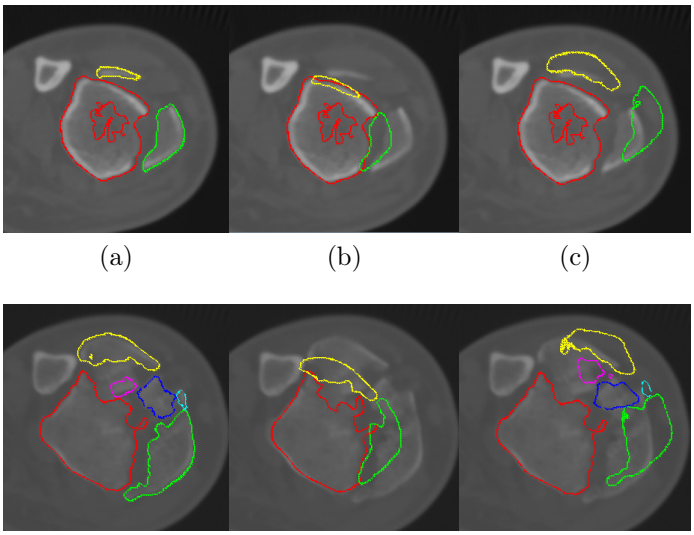
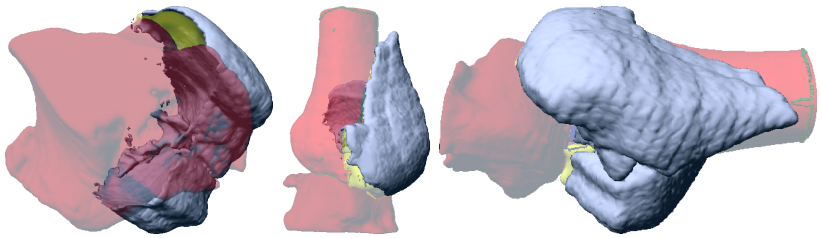


Figure 55: shows contours of the six fractured fragments measured from the CT image, reduced fragments, and fracture fragments at the end of the simulation for the first value of $\hat{\Theta}$. The contours are shown at two different slices of the CT image: slice 80 (top row) and slice 108 (bottom row). Slice 80 shows contours for three of the six fragments, while slice 108 shows the contours for all six fragments.

Figure (55) shows contours of the fractured fragments measured from the CT image, reduced fragments, and fracture fragments at the end of the simulation for the first value of $\hat{\Theta}$. The contours are shown at two different slices of the CT image: (1) slice 80 that shows the contours for three fragments and (2) slice 108 that shows the contours for all six fragments. This figure is provided to compare visually the positions and orientations of the fragments computed via the simulation with their measured ones within the fracture CT image. Table (6) shows the translation error for each fragment. Fragment 1 has a 0 error since it is kept fixed during the simulation. This bone represents the base bone that is used in the reconstruction system. The average translation error for all fragments, i.e., $\|\mathbf{B} - \mathbf{B}(t_f|\Theta)\|$, is equal to 67mm as computed in equation (29). This figure and table help the user to understand more

Table 6: shows the computed translation error for each fragment $\|\mathbf{B}_i - \mathbf{B}_i(t_f|\Theta)\|$ for the most likely value for $\hat{\Theta}$ for a cylindrical strike object to generate a fracture simulation. The first row indicates the fragment ID while the second row report the translation error for each fragment. The color code in the first row indicate the color of the corresponding fragment contour in figure (55). Fragment 1 has a 0 error since it is kept fixed during the simulation. This bone represents the base bone that is used in the reconstruction system.

Fragment ID	1(Red)	2 (Green)	3 (Yellow)	4(Blue)	5 (Pink)	6 (Turquoise)
$\|\mathbf{B}_i - \mathbf{B}_i(t_f\|\Theta)\|$(mm)	0	35	89	105	127	45



(a) (b) (c)

Figure 56: shows a 3D representation of the estimated void region in soft tissue in Experiment-3. Figures (a-c) show the void region in soft tissue in solid light blue from three different view perspectives: axial, coronal, and saggital, respectively. The void region in soft tissue surface is shown with respect to other bone fragment surfaces (in faded colors) in their estimated original anatomic positions in order to visualize the shape and location of void region in soft tissue in the fractured ankle.

the translation error of the simulation result.

Figure (56) shows a 3D representation for the surface of the estimated void region in soft tissue due to the movements of fracture fragments during the fracture event. The surface of the void region in soft tissue is shown from three different view perspectives: axial, coronal, and saggital. The surface is drawn with respect to other bone fragment surfaces in their estimated original anatomic positions in order to visualize the shape and location of the void region in soft tissue in the fractured ankle. Figure (57) shows a chart for the estimated number of pixels for void region per slice in the fracture image. This chart provides a quantitative estimate for the void region in soft tissue. The information conveyed by this chart and the 3D representation of the void region

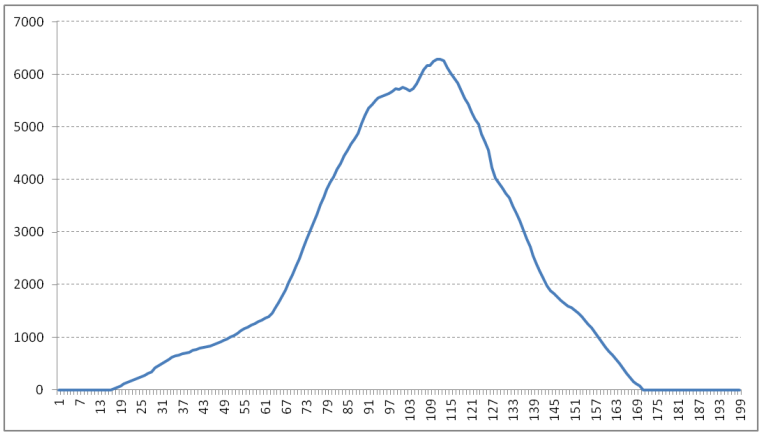


Figure 57: A chart for the void region in soft tissue that is estimated in Experiment-3. The x-axis represents the slice index while the y-axis represents the number of pixels of the void region in soft tissue per slice. These pixels are the ones that the bone fragments passed through during a fracture simulation.

in soft tissue may help the user in locating the most damaged region of the soft tissue in the real fractured ankle.

6.4 Discussion

This suction discusses the results and the distinctive aspects of the experiments and the generated fracture simulations using the system. The results as shown in the snapshots of fracture simulations indicate that it is possible to construct a virtual dynamics model for the fractured limb, to break that model, and to generate a fracture simulation for a fracture event.

The translation errors reported in the tables for fracture fragments indicate that the system was able to virtually move the fragments from their estimated anatomical positions to fractured ones that are close to the measured positions of fracture fragments in the fracture CT image. It is noticeable that the error for larger fragments is smaller than the error for smaller fragments. That is because a large fragment has a large surface that allows it to interact more with other virtual objects such as soft

tissue compared to small fragments. So a large fragment may have more virtual collisions with other objects than a small fragment which may pass through gaps between the virtual objects. This note is clear in figures for CT images with fragment contours. The contours for large fragments at the end of a fracture simulation are close to their corresponding contours for measured fragments in contrast to small fragment contours.

The relatively high error values can be explained, in part, by four reasons: curse of dimension, imperfection of used model, values of the fixed parameters, and limitations of the physics engine. Curse of dimension causes the error surface to have too many local minima that make it hard for the system to find the absolute minimum without checking every possible solution in the space of this surface and this is impractical. The used model is imperfect because it does not include other elements that exist in the fractured limb and interact with other elements during the fracture event, for example, tendon and ligament. These missing elements may improve the quality of the results. The values of the fixed parameters represent estimated physical attributes of the fractured limb and these attributes may vary for different people. Adding these parameters to the search process may improve the results but it is going to increases the dimensions of the search space and this is not desirable. The used physics engine, Bullet, has some technical limitations that does not allow the system to search the whole space of solutions for example, Bullet does not handle speed values that are too high because it may miss collision surfaces in its computation. These reasons can be overcome by reducing the dimension of the search space to the most important unknown parameters, using a more accurate model of the limb, using a search method

that check so many possible solutions, and using a more sophisticated commercial physics engine.

The reported void region in soft tissue provides an estimate about its location in the real fractured limb. The 3D representation helps in visualizing its shape with respect to the reconstructed bone fragments, while the chart helps in localizing the void region along the axial axis. The void region may help the user in estimating the damaged areas in soft tissue in a fractured limb.

The results of fracture simulations for this prototypical system indicate that it is possible to construct a virtual dynamics model for the fractured limb, to break that model, and to generate a fracture simulation for a fracture event. The system was able to generate a list of plausible solutions that a user can explore to find the one that is most likely generated the fracture event as presented in the fracture CT image.

CHAPTER 7: BONE FRAGMENT SEGMENTATION ALGORITHM

In this chapter, a novel bone fragment segmentation algorithm is presented. The bone fragment segmentation algorithm uses the PWT algorithm to extract bone fragment regions from a CT image. The proposed algorithm seeks to address two issues: (1) how to accurately segment bone fragments that touch, i.e., are in close proximity to one another, and (2) how to accurately segment bone fragment tissue from other tissues. It does this by developing novel probabilistic models for both situations and then integrating these models into a single probabilistically-driven version of the classical watershed transform, referred to as the PWT. This chapter begins by explaining the PWT and it's key variations away from the classical watershed transform.

7.1 Probabilistic Watershed Transform

Th PWT is an efficient approach to classify image pixels by propagating classifications decisions from high confidence areas to lower confidence areas following the path with maximum confidence. Confidence is measured by probability and a high confidence area is referred to as a marker. A path integrates connectivity information that describes all of the pixels intervening between a pixel itself and the marker in consideration. Integrating this information is important for reliable classification. A path is propagating information from a marker through a connected set of data to the pixel into consideration so it is always propagating information through the data

rather than classifying the data independent of any connectivity. In general, computing this is hard. The PWT algorithm is adopted so that to compute the maximum probability path. This Integrates the information about the data intervening between the marker and the pixel into consideration where this information can be in arbitrary units or context since the PWT operates on probability distributions.

The PWT algorithm consists of three steps:

1. Classify the input image pixels into three sets: markers, data, and background,

2. Compute class-conditional probabilistic reliefs, i.e., cost images,

3. Compute watershed: Expand the image markers in the order of slowest cost ascent until all non-markers pixels belong to some marker.

Marker computation uses reliable data as a basis for estimating unreliable data. This allows to propagate high probability solutions into areas with lower probability where the correct classification is less certain. The segmentation algorithm starts by classifying image pixels into three sets: markers, data, and background. The algorithm starts by computing a binary image referred to as the marker image. This image is generated by classifying each image pixel into two classes: (1) marker pixels and (2) non-marker pixels. The classifier conservatively labels pixels as marker pixels, i.e., marker pixels are only those pixels that are highly likely to include the semantic information of interest for segmentation. After the marker image is computed the connected components of the marker image is computed to generate a collection of initial foreground regions which are referred to as markers. The image pixels that are non markers are classified into two sets: data and background. Data set is the image

pixels that are assumed to be part of the objects and will be processed by the PWT algorithm. Background set is the image pixels that are assumed to not be part of any object. Background pixels will not be processed by the PWT.

Using the computed markers and the input image, a set of cost images is computed which is an interim step that enables the algorithm to efficiently compute the image watershed from the initial markers. Cost images are defined to make propagation decisions from high confidence areas to lower confidence areas through the path of maximum probability. Cost images specify the penalty for expanding the markers through the different image pixels. The number of cost images is equal to the number of markers, i.e., one cost image for each marker. These cost images are generated from probability distributions which define the stochastic relation between each pixel and the best segmentation label for the pixel. The use of probabilistic trend allows integrating costs of various decisions without having to worry about the units that these measurement were made in. The typical watershed transform works on intensity alone so the watershed transform for color images, for example, is difficult because it is difficult to know which intensity is more important as a unit of red, as a unit of green, or as a unit of blue.

The watershed transform computes the geodesics across the probability distributions such that the total cost of the path from a pixel to the associated marker has maximum probability. The algorithm propagates classification decisions from area of high confidence to areas of lower confidence. Markers are the areas with the highest confidence. Those are neighboring regions, every time you move away from the marker boundary the confidence of the classification decreases. So the algorithm propagates

classification decisions out through paths of maximum probability starting by high confidence pixel first. Then to classify a pixel, the algorithm does not take only the information of that pixel but the information of pixels along the path. Computing this is hard. The PWT is an efficient approach to compute this path. A path integrates connectivity information that describes all of the pixels intervening between the pixel itself and the marker in consideration.

The discussion of PWT in this section is developed as follows. The concept of watershed transform is presented in section (7.1.1). Classical watershed transform is discussed with its drawbacks in section (7.1.2). Improved watershed transform algorithm is discussed with its problems in section (7.1.3). PWT concept and algorithm are presented with detailed explanation in section (7.1.4).

7.1.1 Watershed Transform Algorithm

The watershed algorithm is a region-growth oriented segmentation method having origins taken from the field of mathematical morphology. The algorithm segments an image into regions using the geometric shape of the image values as the basis of the segmentation. The name "watershed" is derived from an analogy which likens the process of computing these regions to the problem of finding a watershed in a topographic relief. In a topographical context, "watershed" denotes the topographic regions divided by ridges that will drain water to different bodies of water. The watershed algorithm computes these regions treating the image intensities as the altitude of the topographic relief.

Much of the terminology for this algorithm is borrowed from that used in geology

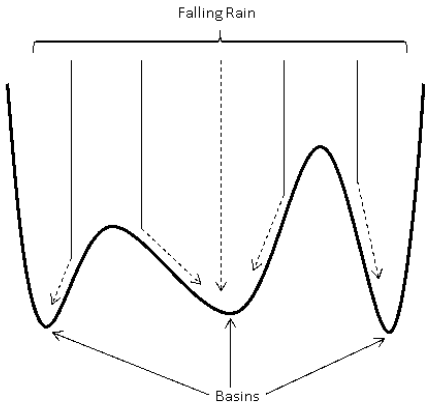


Figure 58: Illustration of Rain Falling flooding process in watershed transforms. The water drops follow the landscape downhill forming lakes at the bottom of the valleys which are called basins.

for watersheds. For example, the lowest point in each region is called a catchment basin (CB) or simply a basin, see figure (58) . Within images, catchment basins correspond to local minima of the image intensity. The algorithm works by first locating these catchment basins within the image and subsequently raising the *"*water level*"* in the topographic relief. A variety of terminologies are used in watershed related publications for the "water level." These include the following terms: height, level, altitude, and intensity. In this dissertation we refer to the "water level" as the height. The height of the water is uniform over the relief and increases with uniform speed. Neighboring basins eventually fill with "water" and merge when the water height exceeds the altitude of the ridges that separate the two basins. When this occurs, dams are built at the locations where water coming from different basins meet to keep them separate. As a result, the landscape is divided into regions and the dams formed to separate the catchment basins define the the watershed lines or *"*watershed*,"* see figure (59)

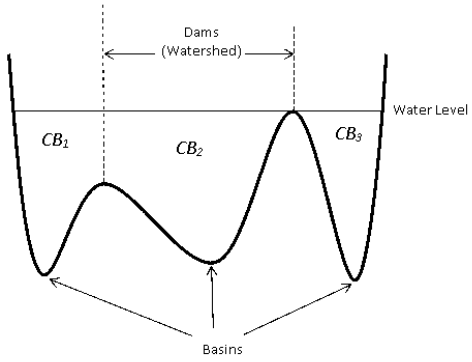


Figure 59: Illustration of watershed transform, dams are built at the points the water coming from two different lakes may meet, the union of dams represent the watershed lines. CB refers to catchment basin.

In images, the intensity values define the altitudes of a topographic relief where high intensities (bright areas) represent high altitudes and low intensities (dark areas) represent low altitudes, see figure (11). Computing the watershed of an image is a two-part problem: (1) locating local intensity minima (catchment basins) and (2) associating each image pixel with one of the detected catchment basins. Local minima can be found by looking for sign changes in the first and second order discrete differentials of the image intensity surface. The found local minima are called markers. The second problem must be solved by finding the path of fastest descent from each image pixel to some catchment basin (which must be a regional minimum). A simple approach to solve this problem is to use the discrete approximation of the gradient to determine the path of fastest descent on the surface until a catchment basin is reached. This "direct" method for computing the watershed proceeds by starting at a non-marker pixel and traversing the image by visiting a sequence of adjacent pixels until a marker pixel is reached. At each point in the path, the adjacent neighbor is chosen by following the negated intensity surface gradient. Let $i = f(\mathbf{x})$ define an

intensity function where intensity i occurs at position $\mathbf{x}$. In this case, the path of fastest descent at the point $\mathbf{x}$ is given by traversing to the "downhill" neighboring pixel position referred to as $\hat{\mathbf{y}}(\mathbf{x})$. The "downhill" pixel is selected using equation (42).

$$\hat{\mathbf{y}}(\mathbf{x}) = \arg_{\mathbf{y}} \left\{ \max_{\mathbf{y} \in NG(\mathbf{x}), f(\mathbf{y}) < f(\mathbf{x})} \left\langle -\nabla f(\mathbf{x}), \frac{\mathbf{y} - \mathbf{x}}{||\mathbf{y} - \mathbf{x}||} \right\rangle \right\} \tag{42}$$

where $NG(\mathbf{x})$ denotes the neighbors of point $\mathbf{x}$, i.e., the set of locations which share an edge (or face in 3D) with the point $\mathbf{x}$. Unfortunately, this approach is computationally costly due to the following two factors: (1) spatially-close pixels will often follow the nearly the same descent path and (2) for complex surfaces, these paths can create long and circuitous descent paths before terminating at a marker. Also, this initial version of the watershed algorithm often divided semantic objects within the image into many segments; a problem known as over-segmentation. Over-segmentation is generally undesirable as the user must merge these segments to produce a viable segmentation. Oversegmentation issues exist for several reasons: (1) noise, (2) semantics. Noise is a random perturbation of the image intensity due to the sensing process. Noise typically introduce local minima which are not indicative of semantic contents. For watershed algorithm, these minima create distinct regions that do not have a semantic meaning. A semantic object may consist of components which introduce multiple local minima due to natural variation in the texture of the object surface. In such cases, the object will be divided into multiple parts and this is not desirable, see figure (12).

7.1.2 Classical Watershed Transform Algorithm

The classical watershed algorithm first introduced in 1991 by Meyer [51]. Meyer addressed the issues of computation cost and over-segmentation by modifying the topographic relief. The modification was successful in eliminating many of the local minima responsible for over-segmentation and simplified the computation of the path of steepest descent as described in equation (42). The resulting algorithm is less prone to over-segmentation, albeit this is still an issue, and provides the same watershed result using a fraction of the computational cost of the direct solution referenced above. This work is referred to as the "classical watershed." Meyer's work in this regard is the basis of contemporary research in this area including the approach proposed in this dissertation. Meyer's algorithm is as follows:

1. Compute the image markers and non-marker pixels,
2. Compute the cost image,
3. Expand the image markers in the order of slowest cost ascent until all non-markers pixels belong to some marker.

The algorithm starts by computing a binary image referred to as the marker image. This image is generated by classifying each image pixel into two classes: (1) marker pixels and (2) non-marker pixels. The classifier conservatively labels pixels as marker pixels, i.e., marker pixels are only those pixels that are highly likely to include the semantic information of interest for segmentation. Classification is done in this way for two reasons (1) the marker image is intended to be a coarse labeling of the final

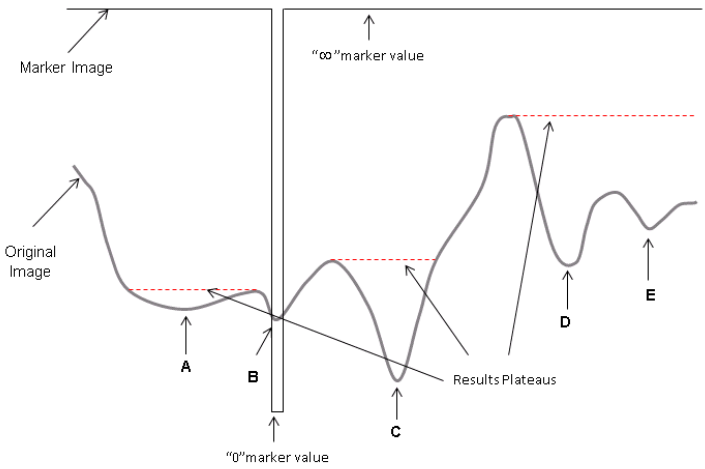


Figure 60: Minimum imposition into 1D image. The original image has local minima at A, B, C, D, and E. The marker image has a single zero at minimum B. The red plateaus show the filled minima after imposition. The result image has only one local minima at B.

segmented image and (2) each contiguous group of marker pixels will ultimately define a distinct watershed region having a unique label. After the marker image is computed the connected components of the marker image is computed to generate a collection of initial foreground regions which are referred to as markers.

Using the computed markers and the input image, a cost image is computed which is an interim step that enables the algorithm to remove local minima in the intensity image and efficiently compute the image watershed from the initial markers. Local minima are removed by setting each marker as a minimum and subsequently assigning non-marker pixels to have monotonically increasing costs with respect to the markers. This process is referred to as imposing minima, see figure (60) . The watershed is then computed by growing the markers into adjacent non-marker pixels with lowest cost first. This growth strategy selects the lowest cost pixel, with respect to the starting markers, from the set of pixels adjacent to the image markers and merges it into its

adjacent marker which is the "downhill" pixel described in equation (42).

The cost image simplifies the computation of the image watershed by re-using path computations between spatially close pixels and by shortening long and possibly circuitous paths between non-marker pixels and their markers. The cost image is created using two criteria: (1) regional minima must occur at each marker and (2) each pixel belonging to the watershed of a given marker will have a least-cost path in the cost image from the non-marker pixel position to a position on the given marker boundary corresponds to the path of fastest descent as described in equation (42).

These criteria are satisfied by a carefully orchestrated sequence of mathematical constructions. The first of which is to define a cost function that attributes a cost for traversing the path edge that connects two spatially adjacent pixels. Let $(\mathbf{x}_i, \mathbf{x}_j)$ denote two neighboring (adjacent) pixels in the image. The cost of traversing the edge connecting $\mathbf{x}_i$ to $\mathbf{x}_j$ is defined in equation (43).

$$cost(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} f(\mathbf{x}_i) - f(\hat{\mathbf{y}}(\mathbf{x}_i)) & f(\mathbf{x}_i) > f(\mathbf{x}_j) \\ f(\mathbf{x}_j) - f(\hat{\mathbf{y}}(\mathbf{x}_j)) & f(\mathbf{x}_i) < f(\mathbf{x}_j) \\ \frac{f(\mathbf{x}_i) - f(\hat{\mathbf{y}}(\mathbf{x}_i)) + f(\mathbf{x}_j) - f(\hat{\mathbf{y}}(\mathbf{x}_j))}{2} & f(\mathbf{x}_i) = f(\mathbf{x}_j) \end{cases} \tag{43}$$

Note that the cost definition (43) takes the point from of the pair $(\mathbf{x}_i, \mathbf{x}_j)$ that has highest intensity and uses the intensity difference between that point and its "downhill" neighbor as the cost. This definition guarantees that the least-cost path in the cost image from a non-marker pixel position to a position on a given marker boundary corresponds to the path of fastest descent as described in equation (42).

Averaging occurs when the intensities at these points are equal. The cost between two neighboring pixels is equal at least to the difference between their intensities or higher. The least cost of traversing the edge connecting $\mathbf{x}_i$ to $\mathbf{x}_j$, i.e., $cost(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_j) - f(\mathbf{x}_i)$, is obtained only when $\mathbf{x}_i$ is the "downhill" neighboring pixel of $\mathbf{x}_j$, i.e., $\mathbf{x}_i = \hat{\mathbf{y}}(\mathbf{x}_j)$. This definition for the cost allows for a least-cost path to be computed which, due to the properties of (43), generate a path corresponds to the path of fastest descent in the cost image and this is desired.

Since cost focuses merely on the change in the intensity (also referred to as the topographic height/altitude) Meyer refers to cost as the *topographic distance* between neighboring pixels. This analogy is then extended to include multiple transitions between adjacent pixels to construct a path. The path between two image positions $\mathbf{x}_i$ and $\mathbf{x}_j$ is represented as a sequence of L image positions referred to as $\pi(\mathbf{x}_1, \mathbf{x}_L) = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ where the positions have been re-indexed in the order suggested by the path sequence. The topographic distance of the path π is referred to as $TD^{\pi}(\mathbf{x}_1, \mathbf{x}_L)$ which is defined in equation (44).

$$TD^{\pi}(\mathbf{x}_1, \mathbf{x}_L) = \sum_{i=1}^{L-1} cost(\mathbf{x}_i, \mathbf{x}_{i+1}) \tag{44}$$

For each non-marker pixel there will exist a very large number of paths between that pixel and the markers of the image. Yet, each pixel will have at least one path of fastest descent within the cost image. This path may be computed by following the negative of the cost image gradient, i.e., let $\mathbf{x}_1$ denote the non-marker start pixel and subsequently select $\mathbf{x}_{i+1}$ using the recurrence relation $\mathbf{x}_{i+1} = \hat{\mathbf{y}}(\mathbf{x}_i)$ as defined in

equation (42) until a marker pixel is reached, at which point we define the marker pixel to have path index L. This path is referred to as the geodesic of the topographic distance function between the points $\mathbf{x}_1$ and $\mathbf{x}_L$ and the topographic distance of this path is denoted $TD(\mathbf{x}_1, \mathbf{x}_L)$ as it is a shortest-path criterion using the topographic distance as a metric as shown in equation (45).

$$TD(\mathbf{x}_1, \mathbf{x}_L) = \min_{\pi \in \Omega(\mathbf{x}_1, \mathbf{x}_L)} TD^{\pi}(\mathbf{x}_1, \mathbf{x}_L) \tag{45}$$

where $\Omega(\mathbf{x}_1, \mathbf{x}_L)$ is the set of all paths from $\mathbf{x}_1$ to $\mathbf{x}_L$. In [52], it is proven that paths of the fastest descent are the geodesics of the topographic distance functions and they are the paths of minimal cost.

Proposition: The topographic distance for a path of fastest descent is equal to the difference of the function values between their terminal points .

Proof: Let π^* be a path of fastest descent from $\mathbf{x}_1$ to $\mathbf{x}_L$ and $\pi^* = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ be the set of points belonging to π^* such that $\mathbf{x}_{i+1} = \hat{\mathbf{y}}(\mathbf{x}_i) \, \forall i \in [0, L-1]$. Let f be a function of $\mathbf{x}$, then, $\mathbf{x}_{i+1}$ is the point with the lowest function value in the neighborhood of $\mathbf{x}_i$ since $\mathbf{x}_{i+1} = \hat{\mathbf{y}}(\mathbf{x}_i)$. Therefore, the cost for walking from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$ is equal to the difference between the function values for the two points, i.e., $cost(\mathbf{x}_i, \mathbf{x}_{i+1}) = f(\mathbf{x}_i) - f(\mathbf{x}_{i+1})$. The topographic distance along path π^* is equal to the sum all costs between adjacent points along the path, i.e., $TD^{\pi^*}(\mathbf{x}_1, \mathbf{x}_L) = \sum_{i=1}^{L-1} cost(\mathbf{x}_i, \mathbf{x}_{i+1}) = \sum_{i=1}^{L-1} f(\mathbf{x}_i) - f(\mathbf{x}_{i+1}) = f(\mathbf{x}_1) - f(\mathbf{x}_L)$. Hence, the topographic distance for a path of fastest descent is equal to the difference of the function values between their terminal points, i.e., $TD^{\pi^*}(\mathbf{x}_1, \mathbf{x}_L) = f(\mathbf{x}_1) - f(\mathbf{x}_L)$.

Proposition: Paths of fastest descent are geodesics of the topographic distance function.

Proof: Let π be a path from $\mathbf{x}_1$ to $\mathbf{x}_L$ and $\pi = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ be the set of points belonging to π. Let $\overline{\mathbf{x}_i, \mathbf{x}_{i+1}}$ be a line segment in the path π, then, the cost between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ is equal to the difference between their function values or higher as defined in equation (43), i.e., $cost(\mathbf{x}_i, \mathbf{x}_{i+1}) \geq f(\mathbf{x}_i) - f(\mathbf{x}_{i+1})$. The cost of the line segment $\overline{\mathbf{x}_i, \mathbf{x}_{i+1}}$ is equal to the difference between their function values, i.e., $cost(\mathbf{x}_i, \mathbf{x}_{i+1}) = f(\mathbf{x}_i) - f(\mathbf{x}_{i+1}), iff\ \mathbf{x}_{i+1} = \hat{\mathbf{y}}(\mathbf{x}_i)$. If the path π is not the path of fastest descent, then, there will be at least a line segment $\overline{\mathbf{x}_i, \mathbf{x}_{i+1}}$ for which $\mathbf{x}_{i+1} \neq \hat{\mathbf{y}}(\mathbf{x}_i)$ and $cost(\mathbf{x}_i, \mathbf{x}_{i+1}) > f(\mathbf{x}_i) - f(\mathbf{x}_{i+1})$. Therefore, $TD^{\pi}(\mathbf{x}_1, \mathbf{x}_L) = \sum_{i=1}^{L-1} cost(\mathbf{x}_i, \mathbf{x}_{i+1}) > f(\mathbf{x}_1) - f(\mathbf{x}_L) = TD^{\pi^*}(\mathbf{x}_1, \mathbf{x}_L)$, where π^* is the path of fastest descent from $\mathbf{x}_1$ to $\mathbf{x}_L$ according to Proposition 7.1.2. Hence, the paths of fastest descent are geodesics of the topographic distance function.

Proposition 7.1.2 and Proposition 7.1.2 are also shown true in [52]. Constructing the cost function using the amount of rise at the point along the gradient direction is analogous to the cost constructed from lower slope in [52]. As time has passed this approach for computation of the watershed has been coined "geodesic reconstruction" as it seeks to reconstruct the watershed by "growing" the marker regions along the geodesic paths of the topographic distance surface (the cost image) from the bottom up. Geodesic reconstruction is computationally efficient as it needs to traverse each pixel only once to assign it the correct label where the "direct" method may traverse the same pixel many times. Work in [64] analyzed this approach for computing the watershed and proved that using the method is equivalent the more computationally

intensive direct method for computing the watershed. This work is summarized in the definition below:

Definition: Let f denote an intensity function that is strictly increasing from its minima defined in domain $\mathbb{D}$ with regional minima set $\mathbf{M} = \{M_k\}_{k \in K}$ where K is its index set. Let $f(\mathbf{x})$ denote the value of an intensity function at the point $\mathbf{x}$ in the image. Each regional minimum M_i in the image has a catchment basin denoted by $CB(M_i)$ and assigned a label i. $CB(M_i)$ is the set of points that are topographically closer to M_i than any other regional minimum M_j:

$$CB(M_i) = \{\mathbf{x} | f(M_i) + TD(\mathbf{x}, M_i) \underset{\forall j \neq i, j \in K}{\leq} f(M_j) + TD(\mathbf{x}, M_j))\} \tag{46}$$

Watershed, denoted $wshed$, of f is the set of image points that do not belong to any catchment basin:

$$wshed(f) = \mathbb{D} \cap \left[\bigcup_{\forall i \in K} CB(M_i) \right]^C \tag{47}$$

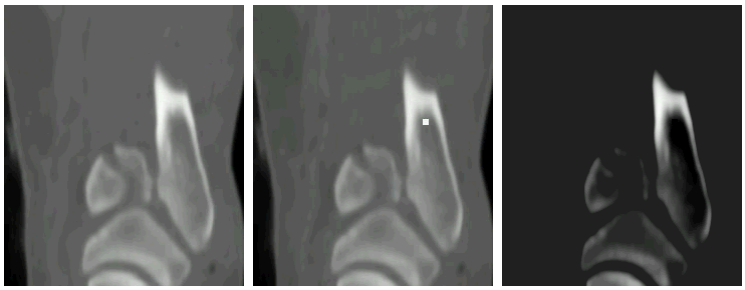
Let W be a distinct value that does not belong to K, i.e. $W \notin K$ and label watershed. Then, the watershed transform of function f is a mapping $Label : \mathbb{D} \rightarrow K \cup \{W\}$, such that:

$$Label(\mathbf{x}) = \begin{cases} i & if\, \mathbf{x} \in CB(M_i) \\ \\ W & if\, \mathbf{x} \in wshed(f) \end{cases} \tag{48}$$

A regional minimum is either a single point or a set of connected points that have the same intensity value.

The watershed transform generates a unique catchment basin for each minimum, each catchment basin is given a unique label, each pixel in the image is given the label of the catchment basin which has the shortest-path to the pixel using the topographic distance as a metric as described in equation (45). The pixel is assigned the watershed label, i.e., W, in the case where two or more catchment basins have the shortest-path to the pixel using the topographic distance as a metric.

Meyer's algorithm is efficient but suffers from several shortcomings that limit its application. These shortcomings include the following: (1) the cost function defined in equation (43) only considers intensity/the geometry of the intensity surface, (2) loss of image information, and (3) no special class for the background. The cost as defined in equation (43) focuses merely on the change in the intensity between neighboring pixels using one cost function. In practice, objects may have more semantic information, e.g., color, that distinguish them from each other. Also, each marker may have its own semantic, e.g., probabilistic information, which help in determining the pixels that should be merged to it. Such information cannot be incorporated in the intensity for one cost function causing the watershed algorithm to incorrectly segment the objects regions in the image. The loss of image information may reduce the watershed algorithm accuracy in segmenting the objects region from the image. Image data is modified to have regional minima at the markers positions in the cost image. This modification may remove the boundaries and reduce the contrast between the objects regions which cause the watershed algorithm to incorrectly segment the objects regions in the image and this is not desired, see figure (61) . No special class for the background reduces the watershed algorithm accuracy in segmenting objects



(a) (b) (c)

Figure 61: Information loss due to marker imposition, (a) Original image, (b) The marker to be imposed position on the original image, (c)The result of imposing the marker as minimum on the original image, a lot of details are lost and it is clearly apparent the right side of the image, the details on the right side of the original image are were deleted in order to impose marked minimum.

that have diffused boundaries with the background. In the classical watershed algorithm, the image pixels are classified into two classes only: marker and non marker. Therefore, a marker should be placed in every background region regardless of its size in order to segment the background area from the objects otherwise unmarked background areas will be considered part of the surrounding objects. Background markers may add many markers to the markers set. Image data is modified to have regional minima at every marker position in the cost image. The loss of image information is proportional with the number of markers. This loss of information may reduce the watershed algorithm accuracy in segmenting the objects region from the image.

7.1.3 Improved Watershed Transform Algorithm

The improved watershed transform algorithm was introduced in 2004 by Grau et al. [34]. Grau addressed the issues of considering only the intensity for one cost function and the loss of image information by modifying the watershed transform to utilize a set of probability functions that encode prior information about the objects.

The modification was successful in introducing multiple cost images which encode the stochastic dependence between the intensity of two neighboring pixels and the boundary of the unknown object. Also, the algorithm avoided the loss of image information by eliminating the need of imposing minima prior the watershed computation. The resulting algorithm utilizes prior information about the objects and provides watershed result less sensitive to noise. This work is referred to as the "improved watershed".

The improved watershed algorithm is as follows:

1. Compute the image markers and non-marker pixels,
2. Compute the cost images,
3. Expand the image markers in the order of slowest cost ascent until all non-markers pixels belong to some marker.

Similar to classical watershed, the algorithm starts by computing a binary image referred to as the marker image. This image is generated by classifying each image pixel into two classes: (1) marker pixels and (2) non-marker pixels. The classifier conservatively labels pixels as marker pixels, i.e., marker pixels are only those pixels that are highly likely to include the semantic information of interest for segmentation. After the marker image is computed the connected components of the marker image is computed to generate a collection of initial foreground regions which are referred to as markers.

Using the computed markers and the input image, a set of cost images is computed which is an interim step that enables the algorithm to introduce prior knowledge about the objects and efficiently compute the image watershed from the initial markers. The

number of cost images is equal to the number of markers, i.e., one cost image for each marker. Cost images are computed using the probability functions which represent the stochastic dependence between the intensity of two neighboring pixels and the boundary of the unknown object. The watershed is then computed by growing the markers into adjacent non-marker pixels in order of lowest cost ascent. This growth strategy selects the pixel with lowest cost from the set of pixels adjacent to the image markers and merges it into its adjacent marker.

The improved watershed is computed by a sequence of mathematical constructions. The first of which is to define multiple cost functions, one for each marker, where each cost function attributes the cost for traversing the path edge that connects two spatially adjacent pixels with respect to the marker. Let $(\mathbf{x}_i, \mathbf{x}_j)$ denote two neighboring (adjacent) pixels in the image such that $\mathbf{x}_i$ belongs to the marker with label$k$. The cost of traversing the edge connecting $\mathbf{x}_i$ to $\mathbf{x}_j$ is defined in equation (49).

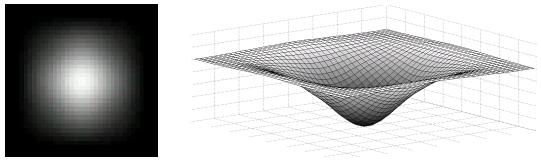
$$cost_k(\mathbf{x}_i, \mathbf{x}_j) = p_k(\mathbf{x}_i, \mathbf{x}_j) \tag{49}$$

where $p_k(\mathbf{x}_i, \mathbf{x}_j)$ is a probability function that represents the probability of having an edge between the $\mathbf{x}_i$ to $\mathbf{x}_j$ given that $\mathbf{x}_i$ belongs to marker with label k. The probability functions are computed using prior information about the objects. In [34], the probability function, $p_k(\mathbf{x}_i, \mathbf{x}_j)$, is defined to be the difference between the posterior probabilities of the edge points $\mathbf{x}_i$ and $\mathbf{x}_j$ where the posterior probability is computed using the prior information provided by Pott's model, which is a generalization of Ising model for more than two states, and atlas.

Improved watershed algorithm suffers from a major shortcoming that limit its application. The shortcoming is its dependency on prior information of the objects. Grau states that the success of the algorithm largely relies on the existence of a prior distribution about the objects of interest. The absence of prior information causes the algorithm to be sensitive to noise and not able to segment areas with low contrast boundaries. Noise introduces local variations to the image intensity are not indicative of semantic contents. The improved watershed detects these variations between the markers as contours. Prior information defines the places of the expected contours in the image. The absence of prior information will lead to detect contours that do not correspond to actual object boundaries. Low contrast boundaries are obtained when the signal to noise ratio is not high enough at the boundary of interest. The improved watershed detects contours with high variations between the markers. Prior information defines the places of the expected contours in the image and increases the variation around them to improve the boundaries detection. The absence of prior information will lead to inaccurate detection of object boundaries and this is not desirable. In many applications, the prior information is not available because of the large variation and randomicity in the objects of interest characteristics such as shapes, positions, orientation, texture, intensity, and color. For this reason, new approaches for segmentation are needed for these application which incorporate better models for objects structures.

7.1.4 Probabilistic Watershed Transform Algorithm

The Probabilistic Watershed Transform (PWT) modifies the watershed transform to allow the introduction of a set of probability distributions which represent the stochastic relation between each pixel and the best segmentation label for the pixel. The PWT is similar to Grau's algorithm in utilizing a set of functions and different than Grau's algorithm in what the probability distributions represent. Probability distributions in the PWT represent the stochastic dependence between the properties of each pixel in the image and the properties of each catchment basin in order to over come the need for a prior distribution and the dependency on local information. In practical applications, each object may have its own semantic information about its surface properties, texture variation, relative intensities, and geometry. Definition (7.1.2) of watershed transform does not allow the introduction of this information. Probability distributions are able to encode semantic information about the objects, e.g., color and likelihood information. PWT extends rain fall analogy to probability distributions by taking the image point probability values as the altitudes of a topographic relief where high probability values represent low altitudes and low probability values represent high altitudes. Regional maxima of the probability distributions correspond to the bottoms of the catchment basins of the landscape, see figure (62). . The cost images are generated by negating these probability distributions. So the minima of the cost images are located at the maxima of probability distributions. In this work, the computation of watershed is conducted by using the probability distributions notation directly since they are the negative of cost images. In this case,



(a) (b)

Figure 62: PWT model, (a) probability distribution with one white blob, (b) topographical relief representation of the probability distribution. The bright area is the region of high probability values in the probability distribution and dark area is the region of low probability values. The high probability values represent the low altitudes in the relief surface and the low probability values represent the high altitudes. The catchment basin corresponds to the bright regions in the probability distribution.

for probability distribution f_k, the path of fastest descent at the point $\mathbf{x}$ is given by traversing to the "downhill" neighboring pixel position referred to as $\hat{\mathbf{y}}_k(\mathbf{x})$ following the probability distribution gradient. The "downhill" pixel is selected using equation (50):

$$\hat{\mathbf{y}}_k(\mathbf{x}) = \arg_{\mathbf{y}} \left\{ \max_{\mathbf{y} \in NG(\mathbf{x}), f_k(\mathbf{y}) > f_k(\mathbf{x})} \left\langle \nabla f_k(\mathbf{x}), \frac{\mathbf{y} - \mathbf{x}}{||\mathbf{y} - \mathbf{x}||} \right\rangle \right\} \tag{50}$$

Each probability distribution has a single maximum, the collection of maxima of the probability distributions are the pixels that are highly likely to include the semantic information of interest for segmentation.

The PWT algorithm addresses the issue of not having a background set in the classical watershed algorithm by creating a unique set that contains the background points and not processing it with the markers. The PWT algorithm is as follows:

1. Classify the input image pixels into three sets: markers, process data, and background,

2. Generate the cost images,

3. Expand the image markers in the order of slowest cost ascent until all non-markers pixels belong to some marker.

The segmentation algorithm starts by classifying image pixels into three sets: markers, process data, and background. Similar to classical watershed, the algorithm starts by computing a binary image referred to as the marker image. This image is generated by classifying each image pixel into two classes: (1) marker pixels and (2) non-marker pixels. The classifier conservatively labels pixels as marker pixels, i.e., marker pixels are only those pixels that are highly likely to include the semantic information of interest for segmentation. After the marker image is computed the connected components of the marker image is computed to generate a collection of initial foreground regions which are referred to as markers, $\mathbf{M} = \{M_k\}_{k\in[1,K]}$. The image pixels that are non markers are classified into two sets: process data and background. Process data set, D, is the collection of image pixels that are assumed to be part of the objects and will be processed by the PWT algorithm. Background set, BG, is the collection of image pixels that are assumed to not be part of any object. Background pixels will not be processed by the PWT.

Using the computed markers and the input image, a set of cost images is computed which is an interim step that enables the algorithm to efficiently compute the image watershed from the initial markers. The number of cost images is equal to the number of markers, i.e., one cost image for each marker. Cost images are computed using the probability distribution, $F = \{f_k\}_{k\in[1,K]}$, which define the stochastic relation between each pixel and the best segmentation label for the pixel. Depending on the

application, probability distributions can be either manually provided by the user or automatically calculated from the set of markers and input image data. The number of probability distributions is equal to the number of markers, i.e., one distribution for each marker. Each probability distribution has only one maximum at the associated marker and monotonically decreasing elsewhere with respect to the marker. A cost image is generated for each probability distribution such that it has local minimum at the associated marker location and subsequently assigning non-marker pixels to have monotonically increasing costs with respect to the marker. The watershed is then computed by growing the markers into adjacent non-marker pixels with lowest cost first. This growth strategy selects the pixel with lowest cost, with respect to the starting markers, from the set of pixels adjacent to the image markers and merges it into its adjacent marker.

The PWT is computed by a carefully orchestrated sequence of mathematical constructions. The first of which is to define multiple cost functions, one for each marker, where each cost function attributes the cost for traversing the path edge that connects two spatially adjacent pixels with respect to the marker. Let $(\mathbf{x}_i, \mathbf{x}_j)$ denote two neighboring (adjacent) pixels in the image such that $\mathbf{x}_i$ belongs to the marker with label$k$, M_k, for distribution function f_k. The cost of traversing the edge connecting $\mathbf{x}_i$ to $\mathbf{x}_j$ is defined in equation (51).

$$cost_k(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} f_k(\hat{\mathbf{y}}(\mathbf{x}_i)) - f_k(\mathbf{x}_i) & f_k(\mathbf{x}_i) < f_k(\mathbf{x}_j) \\ f_k(\hat{\mathbf{y}}(\mathbf{x}_j)) - f_k(\mathbf{x}_j) & f_k(\mathbf{x}_i) > f_k(\mathbf{x}_j) \\ \frac{f_k(\hat{\mathbf{y}}(\mathbf{x}_i)) - f_k(\mathbf{x}_i) + f_k(\hat{\mathbf{y}}(\mathbf{x}_j)) - f_k(\mathbf{x}_j)}{2} & f_k(\mathbf{x}_i) = f_k(\mathbf{x}_j) \end{cases} \tag{51}$$

Note that the cost definition (51) with respect to label k takes the point from of the pair $(\mathbf{x}_i, \mathbf{x}_j)$ that has the minimum probability and uses the probability difference between that point and its "downhill" neighbor as the cost. This definition guarantees that the least-cost path in the cost image from a non-marker pixel position to a position on a given marker boundary corresponds to the path of fastest descent in the topographic relief as described in equation (50). Averaging occurs when the probability at these points are equal. The cost between two neighboring pixels is equal at least to the difference between their probability values or higher. The least cost of traversing the edge connecting $\mathbf{x}_i$ to $\mathbf{x}_j$, i.e., $cost_k(\mathbf{x}_i, \mathbf{x}_j) = f_k(\mathbf{x}_j) - f_k(\mathbf{x}_i)$, is obtained only when $\mathbf{x}_j$ is the "downhill" neighboring pixel of $\mathbf{x}_i$, i.e., $\mathbf{x}_j = \hat{\mathbf{y}}(\mathbf{x}_i)$. This definition for the cost allows for a least-cost path to be computed which, due to the properties of (51), generate a path corresponds to the path of fastest descent in the cost image for marker M_k and this is desired.

Cost in PWT focuses merely on the change in the probability values (also referred to as the topographic height/altitude) and the cost between neighboring pixels is referred to as the topographic distance. This analogy is then extended to include multiple transitions between adjacent pixels to construct a path. The path between two image positions $\mathbf{x}_i$ and $\mathbf{x}_j$ is represented as a sequence of L image positions

referred to as $\pi(\mathbf{x}_1, \mathbf{x}_L) = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ where the positions have been re-indexed in the order suggested by the path sequence. The topographic distance of the path π with respect to label k is referred to as $TD_k^{\pi}(\mathbf{x}_1, \mathbf{x}_L)$ which is defined in equation (52).

$$TD_k^{\pi}(\mathbf{x}_1, \mathbf{x}_L) = \sum_{i=1}^{L-1} cost_k(\mathbf{x}_i, \mathbf{x}_{i+1}) \tag{52}$$

For each non-marker pixel there will exist a very large number of paths between that pixel and the markers of the image. Yet, each pixel will have at least one path of fastest descent within the cost image. This path may be computed by following the cost image gradient with respect to label k, i.e., let $\mathbf{x}_1$ denote the non-marker start pixel and subsequently select $\mathbf{x}_{i+1}$ using the recurrence relation $\mathbf{x}_{i+1} = \hat{\mathbf{y}}_k(\mathbf{x}_i)$ as defined in equation (50) until a marker pixel is reached, at which point we define the marker pixel to have path index L. This path is referred to as the geodesic of the topographic distance function between the points $\mathbf{x}_1$ and $\mathbf{x}_L$ with respect to label k and the topographic distance of this path is denoted $TD_k(\mathbf{x}_1, \mathbf{x}_L)$ as it is a shortest-path criterion using the topographic distance as a metric as shown in equation (53).

$$TD_k(\mathbf{x}_1, \mathbf{x}_L) = \min_{\pi \in \Omega(\mathbf{x}_1, \mathbf{x}_L)} TD_k^{\pi}(\mathbf{x}_1, \mathbf{x}_L) \tag{53}$$

where $\Omega(\mathbf{x}_1, \mathbf{x}_L)$ is the set of all paths from $\mathbf{x}_1$ to $\mathbf{x}_L$. As proven in section (7.1.2), the paths of the fastest descent are the geodesics of the topographic distance functions and they are the paths of minimal cost.

To compute the PWT catchment basins, a unique catchment basin $CB(M_k)$ is

created for each marker M_k, the catchment basins are computed according to the following definition of the PWT:

Definition: Let $\{(f,m)_k\}_{k\in[1,K]}$ be a set of probability functions with their regional maxima where K is the number of catchment basins. f_k has a single regional maximum m_k and it is defined in the domain $\mathbb{D}$. f_k is strictly decreasing from its regional maximum. Each regional maximum m_i has a catchment basin denoted by $CB(m_i)$. $CB(m_i)$ is the set of points that are topographically closer to m_i than any other regional maximum m_j:

$$CB(m_i) = \{\mathbf{x} | -f_i(m_i) + TD_i(\mathbf{x}, m_i) \underset{\forall j\neq i, j\in[1,K]}{\leq} -f_j(m_j) + TD_j(\mathbf{x}, m_j))\} \tag{54}$$

Watershed, denoted $Wshed$, of $F = \{f_k\}_{k\in[1,K]}$ is the set of image points that do not belong to any catchment basin:

$$Wshed(F) = \mathbb{D} \cap \left[\bigcup_{\forall i\in[1,K]} CB(m_i) \right]^C \tag{55}$$

Let W be a distinguish value that does not belong to K, i.e. $W \notin K$ and label watershed. Then, probability watershed transform of set F is a mapping $Label : \mathbb{D} \rightarrow K \cup \{W\}$, such that:

$$Label(\mathbf{x}) = \begin{cases} i & if\, \mathbf{x} \in CB(m_i) \\ \\ W & if\, \mathbf{x} \in Wshed(F) \end{cases} \tag{56}$$

So, PWT assigns unique labels to the points belong to different catchment basins and assigns a distinguish label W to the points of watershed for set F.

In order to define the computational steps of PWT algorithm, let $\{(f, M)_k\}_{k\in[1,K]}$, D, and BG be the set of cost images associated with their markers, process data, and background sets, respectively. The provided PWT algorithm combines geodesic reconstruction with PWT calculations to improve the efficiency. The computation steps are as follows:

1. Initialization: Each marker M_i is assigned a unique label i. All markers points are grouped into subset S and all data points in D are grouped into subset $\overline{S}$. An empty set WS is created, i.e. $WS = \{\emptyset\}$, to store watershed points. BG holds the background points.

2. Select the point $\mathbf{x} \in \overline{S}$ for which

$$\underset{\mathbf{y}\in S,\, \mathbf{x}\in\overline{S},\, \mathbf{x}\in N_G(\mathbf{y})}{f_k(\mathbf{x})} = \underset{\mathbf{v}\in S,\, \mathbf{z}\in\overline{S},\, \mathbf{z}\in N_G(\mathbf{v})}{\max} f_l(\mathbf{z}) \tag{57}$$

   where k and l are the labels assigned earlier to points $\mathbf{y}$ and $\mathbf{v}$, respectively. Point $\mathbf{x}$ is assigned label k and removed from $\overline{S}$: $\overline{S} = \overline{S}\backslash\{\mathbf{x}\}$ and added to S : $S = S \cup \{\mathbf{x}\}$. Then, go to Step 3. If $\mathbf{x}$ is not found do: $BG = BG \cup \overline{S}$, END.

3. For each neighbor $\mathbf{z}$ of $\mathbf{x}$ belonging to S , i.e. $\mathbf{z} \in N_G(\mathbf{x})$, $\mathbf{z} \in S$; if $Label(\mathbf{z}) \neq Label(\mathbf{x})$ do: $Label(\mathbf{x}) = WATERSHED$, remove $\mathbf{x}$ from S : $S = S\backslash\{\mathbf{x}\}$, and add $\mathbf{x}$ to WS : $WS = WS \cup \{\mathbf{x}\}$. If $\overline{S}$ is empty, i.e. $\overline{S} = \{\emptyset\}$, END; else go to Step 4.

4. For each neighbor $\mathbf{z}$ of $\mathbf{x}$ belonging to $\overline{S}$ with $\mathbf{x}$ belonging to S, i.e. $\mathbf{z} \in N_G(\mathbf{x})$, $\mathbf{z} \in \overline{S}$, $\mathbf{x} \in S$; If $f_k(\mathbf{z}) > f_k(\mathbf{x})$ do: $f_k(\mathbf{z}) = f_k(\mathbf{x})$. Return to Step 2.

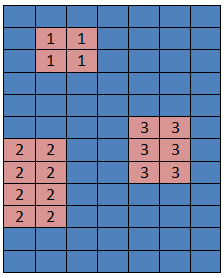


Figure 63: Illustration picture of 2D image with three regional minima $\{m_1, m_2, m_3\}$. Each regional minimum m_i is assigned a distinguish label i and its points are labeled with the same label. The minima points are grouped into set S, colored pink, and all other points are grouped into set $\overline{S}$, colored blue.

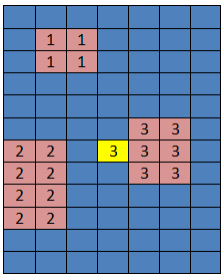
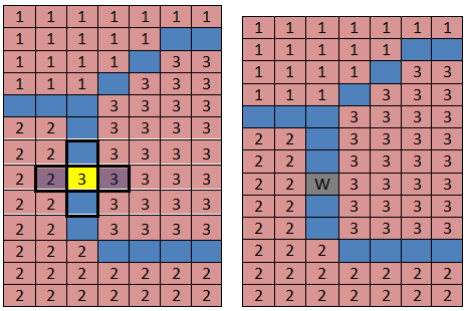


Figure 64: Illustration picture of 2D image shows the selected point $\mathbf{x}$, in yellow, that satisfies equation (57). Point $\mathbf{x}$ is assigned label 3 since point $\mathbf{y}$ belongs to regional minimum m_3.

The algorithm starts by assigning a unique label for each marker and label its points with the same label. The markers points are grouped into set S and data points in D are grouped into set $\overline{S}$. Notice that no point in $\overline{S}$ is assigned a label. An empty set WS is created to store watershed points, i.e. points with watershed label $WATERSHED$. Background points are kept in BG, refer to step 1, see figure (63) . Then, point $\mathbf{x}$ in set $\overline{S}$ which is a neighbor to point $\mathbf{y}$ in set S with the maximum probability function value is selected, assigned the label of point $\mathbf{y}$, removed from set $\overline{S}$, and added to set S, refer to step 2. The maximum probability function value is selected. This step ensures the label expansion process is following the path of fastest ascent by making each point $\mathbf{x}$ gets its label from the neighbor $\hat{\mathbf{y}}_k(\mathbf{x})$, see figure (64) .



(a) (b)

Figure 65: Illustration picture of 2D image. (a) shows the 4-connected neighborhood in thick borders of the selected point, **x**, in yellow, and the neighborhood points that belong to set S, in dark pink. Point **x** has label 3 and one of its neighbors has label 2 so point **x** is a watershed point (b) shows point **x** as a watershed point, in gray, with label W.

If point **x** is not found, i.e., the remaining points in $\overline{S}$ have no labeled neighbor belongs to S, then, move the remaining points in $\overline{S}$ to BG because they are not connected to any marker and stop running the algorithm. If all the neighbors of point **x** belonging to S do not have the same label of point **x**, then, point **x** is assigned the watershed label $WATERSHED$, removed from S, and added to WS, refer to step 3. This step ensures the construction of watershed lines and not expanding their labels by removing the watershed points from set S and store them in WS, see figure (65) . If set $\overline{S}$ is empty the algorithm ends, otherwise, all the neighbors of point **x** belonging to $\overline{S}$ with **x** belonging to S that have probability function values greater than the probability function value at **x** get the probability function value at point **x**, refer to step 4. This step is the implicit implementation of the transformation process to make f_k monotonically decreasing from its regional maximum M_k by transforming all other maxima of f_k into non-regional maxima plateaus. The algorithm keeps running until all the points in set $\overline{S}$ are removed, see figure (66) .

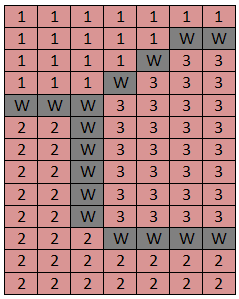


Figure 66: Illustration picture of 2D image shows the final result of PWT algorithm. The different catchment basins are separated by watershed lines. Watershed lines are labeled by W and colored gray.

PWT algorithm utilizes a set of probability functions which can encode semantic information about the objects. PWT algorithm does not require the original image to be modified to process, therefore, there is no loss in image information. PWT algorithm assigns labels to data points that are connected to markers only and do not process background points so an object label will not be expanded to an object that is not connected to it and background region will not be expanded to objects.

The PWT algorithm addresses the issue of the dependency on prior information by using probability distributions which define the stochastic relation between each pixel and the best segmentation label for the pixel. The PWT also resolves the issues of considering only the intensity for one cost function and the loss of image information by modifying the watershed transform to utilize a set of probability distributions. The modification was successful in introducing multiple cost images which encode the stochastic dependence between the properties of each pixel in the image and the properties of each catchment basin. Also, the algorithm avoided the loss of image information by eliminating the need of imposing minima prior the watershed computation. The introduction of the background set makes the algorithm avoids the

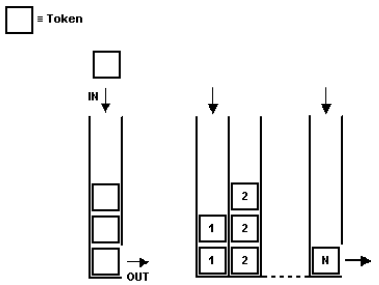


Figure 67: Illustration picture for queue and ordered queue. (a) A queue (FIFO) token are inserted from top and removed from the bottom where each token represents and image data point and carries the its coordinate information. (b) An ordered queue of N-Level so the number of queues is N, all queues are open from the top but, only the queue with the highest priority level available can be emptied.

need to mark every background area in the image and processing them. Also, the background set will improve the PWT speed and accuracy. The speed is improved because the background pixels are not processed in the algorithm so the number of pixels that are going to be processed by PWT algorithm in the image is reduced. The accuracy is improved because the probability of wrong expansion of the background labels to the object of interest is reduced by not processing the background pixels.

7.2 PWT Algorithm With Ordered Queue

PWT algorithm is implemented using an ordered queue where the probability distributions determine the order in which the points are processed at in ordered queue. Ordered queues are very efficient structures for fast computation of morphological transformations, for example, watershed transform and geodesic reconstruction. An ordered queue is a set of queues, a queue is a First In First Out (FIFO) buffer that contains tokens and they are removed in the same order they were added, see figure (67a) . Each token represents an image data point, e.g. a pixel, and it carries two

information: the coordinate information of the point, i.e., (x, y, z), and catchment basin information associated with the token.

Ordered queue is a collection of queues with different priority levels. The number of queues equals to the number of possible priority levels that may exist in an application. Each queue has a single unique priority level. All queues are open at their top and a token can be added to the corresponding priority queue at any time. Only the queue with the highest priority level available at the moment of processing can be emptied and cleared. The removed token from the queue is the one that has been added to it first, see figure (67b). The process of the ordered queue starts by emptying the highest priority queue. Once the queue of the highest priority is empty, it is removed from the ordered queue and the process starts emptying the next queue until all queues are removed. Once a queue is removed it cannot be created again and a low priority queue cannot be processed before all the higher priority queues are emptied and removed.

The PWT algorithm is implemented using an ordered queue. The ordered queue is initialized by adding the outer boundary points of the markers. For each marker M_k, the process points with at least a neighbor inside M_k are added to the ordered queue. Points are added to the ordered queue according to their priorities. The token at the highest priority queue is removed. Let $\mathbf{x}$ be the point corresponds to the token coordinate information and k be the label of the catchment basin associated with the token. If point $\mathbf{x}$ is already assigned a label then it is ignored. Otherwise, if the neighborhood of point $\mathbf{x}$ contains only points with label k or $WATERSHED$, point $\mathbf{x}$ is assigned label k, if not, it is assigned label $WATERSHED$. If point $\mathbf{x}$

is assigned label k , then, each neighbor process point $\mathbf{y}$ of $\mathbf{x}$ which is not labeled is added to the ordered queue with priority $f_k(\mathbf{y})$ if $f_k(\mathbf{y}) < f_k(\mathbf{x})$, otherwise, with priority $f_k(\mathbf{x})$. The ordered queue process continues until all queues are emptied and removed.

The implementation of PWT algorithm with ordered queue has many advantages which are: (1) It marks the watershed line, (2) The queues contains only the borders of the growing regions so it needs less memory, (3) Watershed line never expands and has always a width of 1 point, and (4) Watershed line is contiguous using the opposite connectivity of the one used in the PWT algorithm, for example, in 2D images, watershed line is 8-connected when the algorithm runs using 4-connected, see figure (66), and 4-connected if the algorithm runs using 8-connected.

7.3 Segmenting Bone Fragments Using the PWT Algorithm

The bone fragment segmentation algorithm follows steps of the PWT algorithm to extract bone fragments within CT images. The algorithm starts by classifying image pixels into three classes: (1) cortical bone tissue, (2) non-cortical bone tissue, and (3) non-bone tissue. The classification process is simplistic as it is needed only to differentiate bone tissue from objects other than soft tissue which, in typical bone fracture CT data, is typically only open air. For this reason the segmentation approach classifies the pixels of the image into three categories: (1) cortical pixels, (2) non-cortical pixels, and (3) non-bone tissue pixels. These regions are estimated using a global threshold with two threshold values: (1) cortex T_{cortex} and (2) bone T_{bone}. These values are estimated using a prior distribution for bone tissue, $p(w_{bone})$, in CT

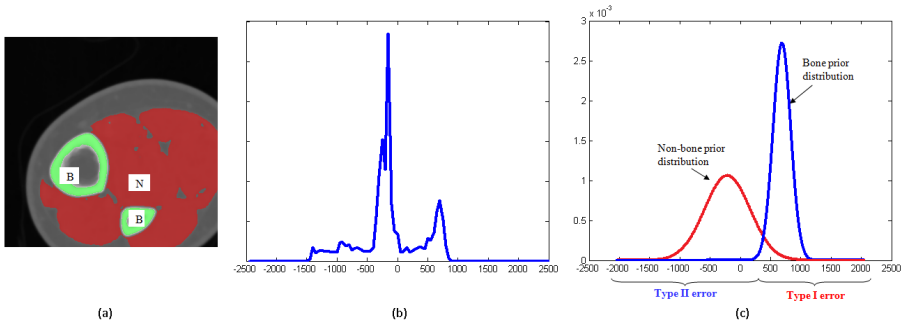


Figure 68: The prior bone distribution that is used to estimate the bone area in a CT image. (a) An original CT image where bone area, denoted by B, and non-bone area, denoted by N, are manually segmented. The CT values in the image are represented 12-bit numbers range from -2048 to 2047. (b) The histogram of pixel intensities for the segmented bone and non-bone areas. (c) The Gaussian distributions for bone prior, and non-bone prior. For the bone prior distribution, the Gaussian function has a mean = 692 HU with a standard deviation = 146HU. For the non-bone prior distribution, the Gaussian function has a mean = -212 HU with a standard deviation = 375HU.

images.

The prior distribution is specified as a function of intensity. Figure (68) shows the generated prior distribution for bone tissue as well as a prior distribution for non-bone tissue. The prior distributions are generated using a known profile for CT intensities for bone tissues within the human body to determine the likelihood that a pixel from a CT image is a pixel coming from bone tissue. The prior distributions are approximated by two Gaussian functions. One is the prior for bone tissue centered at the average of intensities for bone pixels and the second is the prior for non-bone tissue centered at the average of intensities for non-bone tissue pixels. The centers for bone tissue pixels and non-bone tissue pixels are found to be 692HU and -212HU, respectively. These findings support the CT values reported for bone tissue and soft tissue for all human found in [30]. To generate the prior distributions, each CT image is manually segmented by a user to extract two regions: bone region and non-bone

region. Bone region is a set of pixels that a user is certain they correspond to a bone, see figure (68 a). Non-bone region is a set of pixels that a user is certain they do not correspond to a bone and they have relatively high intensities compared to other non-bone areas, see figure (68 a). The histogram distribution of pixel intensities is computed for each region. Then, a Gaussian distribution is fit to the computed histogram distribution for each region such that the mean is set to the average intensity in the region and the variance is set to the variance of intensities in the region. The Gaussian distribution that is generated for bone region is referred as bone prior, $p(w_{bone})$. While the Gaussian distribution generated for non-bone region is referred to as non-bone prior. In this work, these Gaussian distribution are applied to human bone across all genders types and ages. To generate a Gaussian distribution for a specific type, CT images for that type should be used in the generation process of prior distributions.

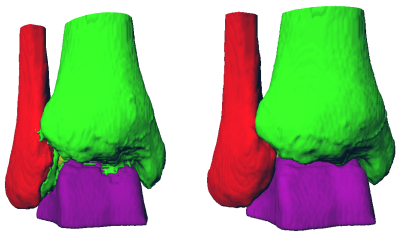
The classification process is accomplished by two stages of classification: (1) classify bone and non-bone tissue and then (2) sub-dividing the group of classified bone tissue into cortical bone tissue and non-cortical bone tissue. Classification (cortical bone /non-cortical bone) is likely to minimize Type 1 errors given the distribution $p(w_{bone})$. Classification (bone/non-bone) is likely to minimize Type 2 errors given $p(w_{bone})$. Minimizing Type 1 and Type 2 errors involves threshold segmentations, with T_{cortex} and T_{bone} thresholds, determined by multiplying $p(w_{bone})$ by a constant and then finding regions where the distribution is $>$ or $<$ some chosen power or significance level. For minimizing Type 1, the constant that multiplies $p(w_{bone})$ is expected to be < 1 shifting T_{cortex} threshold to higher probabilities of $p(w_{bone})$. For minimizing Type

2, the constant is expected to be > 1 shifting T_{bone} threshold in opposite direction. How far the constant deviates from 1 determines how conservative (classification (1)) or how liberal (classification (2)) the classification is. The significance of classification (1) should be small. The power of classification (2) should be as high as possible.

Pixels marked as candidate-bone tissue are given bone fragment class labels using the PWT algorithm. The PWT algorithm uses a collection of probability models for computing bone fragment class labels. These models are separated into two tiers. The first tier of the model computes the likelihood of a bone fragment class, w_{frag_k}, for all bone pixels. The second tier of the model estimates the geometric context of the pixel of consideration and represents the likelihood that a pixel is located in an "isthmus" of bone tissue, refer to section (3.2.4).

Fragment likelihood is constructed as a product of two independent distributions: (1) a fragment tissue probability which expresses the likelihood the observed intensity comes from the k^{th} fragment and (2) a fragment position probability which expresses the likelihood the observed pixel position is a position in the k^{th} fragment. The product of the two represents the joint likelihood of the k^{th} fragment class given a pixel at position $\mathbf{x}$ with intensity $I(\mathbf{x})$. Fragment tissue probability is a Laplace distribution with a mean set by the mean intensity of all pixels within a specified distance of the marker region for the fragment. Fragment position probability is a discrete distribution obtained by computing the distances between each fragment cortical pixels, denoted by M_k, and the pixel of consideration, $\mathbf{x}$, where the probability of the k^{th} fragment class is taken as its percentage of the total of these distances.

Context likelihood is a discrete distribution obtained by computing the distance



(a) (b)

Figure 69: An example to demonstrate the capability of context likelihood in reducing the leak problem, i.e., incorrectly label some parts of a bone fragment with labels of other nearby fragments due to the loss of clear edge information specially in close proximity areas. The segmentation is applied to segment tibia (green), fibula (red) and talus (purple) bones of an ankle. (a) Segmentation result without the use of context likelihood, some parts of fibula and talus are wrongly labeled as tibia. (b) Segmentation result using the context likelihood.

between the edge of bone area and the pixel of consideration, $\mathbf{x}$, where the probability of the k^{th} fragment class is taken as its percentage of the pre-specified minimum expected depth of fragment of interest, T_{depth}. This percentage is useful to specify fragment class probability for pixels located in an "isthmus" of bone tissue. A pixel is assumed to be located inside "isthmus" of bone tissue if it is distance to the edge of bone area is less than T_{depth}. Context likelihood is used to enhance the segmentation result and reduce the leak problem through "isthmus" bone areas. Figure (69) shows an example of segmentation results with and without the use of context likelihood. Context likelihood helps in reducing the leak problem between bone fragments.

Fragment likelihood and context likelihood are combined to form a collection of probability distributions for the PWT algorithm, one for each bone fragment, f_k. Each probability distribution has only one maximum at the location of its corresponding marker and monotonically decreasing elsewhere with respect to the marker. The cost images are generated by negating these probability distributions. So the

minima of the cost images are located at the maxima of probability distributions.

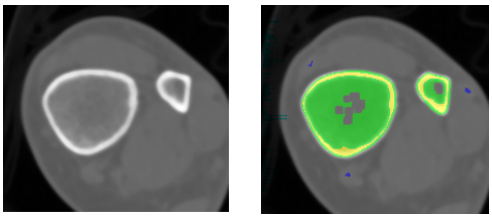
The PWT computes the geodesics across the probability distributions such that the total cost of the path from a pixel to the associated marker has maximum probability. The algorithm propagates classification decisions from area of high confidence, i.e., markers of cortical bone areas, to areas of lower confidence, i.e., non cortical bone areas.

The bone fragment segmentation algorithm takes as input a CT image of the limb (I) and a set of user settings. The output of this algorithm is a labeled image where each unique label corresponds to a unique bone fragment.

7.3.1 Image Pixel Classification

Image pixels classification is the first step in bone fragment segmentation algorithm. This step takes two inputs: (1) the CT image I, and (3) the size threshold T_{size}. The outputs of this step are three: (1) a set of markers, $\mathbf{M}$, (2) the set of non-cortical bone pixels, D, and (3) the set of background pixels, BG. This is accomplished by two stages of classification: (1) classify bone and non-bone (background) tissue and then (2) sub-dividing the group of classified bone tissue into cortical bone tissue and non-cortical bone tissue.

In the first classification stage, the algorithm uses a global threshold method to classify image pixels into one of two classes: (1) bone pixels and (2) background pixels. A pixel is considered a bone pixel if it has an intensity that is equal to the bone threshold, T_{bone}, or higher, i.e., $X_{bone} = \{\mathbf{x}|I(\mathbf{x}) \geq T_{bone}\}$ where $I(\mathbf{x})$ is the image intensity for pixel $\mathbf{x}$. Otherwise, it is considered a background pixel, i.e., $BG = \overline{X}_{bone}$,



(a) (b)

Figure 70: Non marker pixel classification, (a) original image, (b) the PWT data sets: markers area **M** is highlighted in yellow, bone data area D in green and blue, and background area BG in gray. The image is generated using $T_{cx} = 600$HU and $T_{bone} = 90$HU.

see figure (70) .

The bone threshold T_{bone} is determined using the prior distribution for bone tissue, $p(w_{bone})$. The value of T_{bone} is specified by finding the intersection point between bone prior distribution and non-bone prior distribution. In order to minimize Type 2, $p(w_{bone})$ is multiplied by a constant that is > 1 to shift the threshold to lower probabilities of $p(w_{bone})$. In this work, the constant is set to 100 so that the intersection point is approximated to 180HU which is the bone threshold, i.e., $T_{bone}= 140$. Users may change the constant value as appropriate to fit the imaging devices used in their application.

In the second classification stage, the algorithm uses a global threshold method to classify bone pixels X_{bone} into one of two classes: (1) cortical bone pixels and (2) non cortical bone pixels. A pixel is considered a cortical bone pixel if it has an intensity that is equal to the cortical bone threshold, T_{cortex}, or higher, i.e., $X_{cortex} = \{\mathbf{x}|I(\mathbf{x}) \geq T_{cortex}, \mathbf{x} \in X_{bone}\}$. Otherwise, it is considered a non cortical bone pixel, i.e., $D = X_{bone} \cap \overline{X}_{cortex}$.

The cortical bone threshold T_{cortex} is determined using the prior distribution for

bone tissue, $p(w_{bone})$. The value of T_{cortex} is specified by finding the intersection point between bone prior distribution and non-bone prior distribution. In order to minimize Type 1, $p(w_{bone})$ is multiplied by a constant that is < 1 to shift the threshold to higher probabilities of $p(w_{bone})$. In this work, the constant is set to 0.05 so that the intersection point is approximated to 600HU which is the cortical bone threshold, i.e., T_{cortex}= 600. Users may change the constant value as appropriate to fit the imaging devices used in their application.

The set of cortex pixels X_{cortex} is decomposed into disjoint union, i.e., $X_{cortex} = \bigcup_i M_i$, where M_i is a connected component. These connected components are intended to be a coarse labeling of the final segmented image. This is based on the assumption that bone fragments are distinguished by their cortical bone tissue. So that each of these connected components is called a marker and approximate the location of a bone fragment in the image. Each marker will ultimately define a distinct PWT region having a unique label. So, the number of provided markers is the estimated number of bone fragments in the image.

Markers which have a size greater than or equal to the maximum size of a false positive marker T_{size} are added to from a marker set called $\mathbf{M}$, i.e., $\mathbf{M} = \{M_i, |M_i|) \geq T_{size}, M_i \in X_{cortex}\}$ where $|M_i|$ is the number of pixels in M_i. This process filters out false markers due to noise or markers that correspond to bone fragments of insignificance. Markers with size less than T_{size} are added to the set of non-cortical bone pixels, i.e., $D = D \cup \{M_i, |M_i|) < T_{size}, M_i \in X_{cortex}\}$. The set $\mathbf{M} = \{M_1, M_2, ..., M_K\}$ includes connected components which mark the unknown bone fragments in the CT image I, where K is the number of the detected markers.

7.3.2 Probability Distributions Computation

Probability distributions computation is the second step of the bone fragment segmentation algorithm. This step takes three inputs: (1) the CT image I, (2) the set of markers $\mathbf{M}$, (3) the set of bone pixels D, (4) the set of background pixels BG, and (5) the depth threshold T_{depth}. The output of this step is a set of probability distributions that define the stochastic relationship between the properties of each bone pixel and the properties of each of the provided markers. For the bone fragment segmentation problem, a probability distribution $f_k(\mathbf{x})$ determines how likely a pixel $\mathbf{x}$ comes from the k^{th} bone fragment which has class w_k. Probability distributions computation is based on the joint probability of two independent likelihood models: (1) fragment-based probability model and (2) context-based probability model. Fragment-based probability model is a unique likelihood distribution that is based on intensity and position information of the image pixels in order to increase the algorithm robustness to image inhomogeneities. This model is the basic model to compute probability distributions to segment bone fragments. Context-based probability model is the likelihood that a bone pixel has an "isthmus"/non-"isthmus" bone area property. This model is an enhancement to the basic model to cope with the leak problem that may occur through narrow bone areas. Probability distributions computation consists of three stages:

1. Fragment-based probability computation,

2. Context-based probability computation,

3. PWT probability distributions computation.

At the end of these stages, the algorithm computes the probability distributions that are needed to segment bone fragments from the CT image using the PWT algorithm.

7.3.2.1 Fragment-Based Probability Computation

Fragment-based probability computation is the first stage of probability distribution computation. In this stage, the algorithm takes three inputs: (1) the CT image, (2) the set of markers $\mathbf{M}$, and (3) the set of non-cortical bone data D. The out of this stage is the likelihood of bone fragment class k, denoted by w_{frag_k}, given the position $\mathbf{x}$, the intensity $I(\mathbf{x})$, the k^{th} marker, i.e., M_k, and the set of marker regions $\mathbf{M}$. This likelihood is referred to as fragment probability. The fragment probability is based on the product of two likelihoods: (1) position-based likelihood and (2) intensity-based likelihood. The position-based and intensity-based likelihoods are assumed to be conditionally independent. The fragment probability distribution is computed in equation (58):

$$p(w_{frag_k}|I(\mathbf{x}),\mathbf{x},k,\mathbf{M}) = \frac{p(w_{frag_k}|I(\mathbf{x}),k,\mathbf{M})\,p(w_{frag_k}|\mathbf{x},k,\mathbf{M})}{p(I(\mathbf{x}),\mathbf{x},\mathbf{M})} \tag{58}$$

where $p(w_{frag_k}|I(\mathbf{x}),\mathbf{x},k,\mathbf{M})$ is the fragment probability. $p(w_{frag_k}|I(\mathbf{x}),k,\mathbf{M})$ is the intensity-based likelihood that determines the probability of w_{frag_k} given $I(\mathbf{x})$, the k^{th} marker, and the set of markers $\mathbf{M}$. $p(w_{frag_k}|\mathbf{x},k,\mathbf{M})$ is the position-based likelihood that determines the probability of w_{frag_k} given $\mathbf{x}$, the k^{th} marker, and the set of markers $\mathbf{M}$. $p(I(\mathbf{x}),\mathbf{x},\mathbf{M})$ is a normalization factor. Probability is computed for bone pixels only, i.e., $\mathbf{x} \in D$. The computation of probability distribution is based on

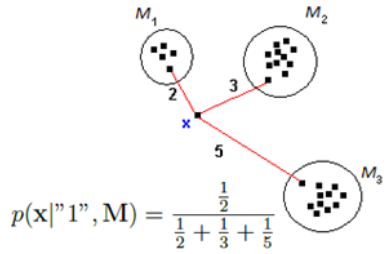


Figure 71: An example of how to compute the position likelihood for pixel $\mathbf{x}$ given the first marker and the set of markers $\mathbf{M} = \{M_1, M_2, M_3\}$. $p(\mathbf{x}|"1", \mathbf{M})$ is the ratio of the reciprocal of the distance between point $\mathbf{x}$ and marker M_1 to the sum of the reciprocals of the individual distances between point $\mathbf{x}$ and all markers, refer to equation (59).

spatial and intensity information that are obtained from inside the image to reduce heterogeneity. Intensity distribution inside the bone region may vary significantly from one dataset to another depending on the parameters of acquisition, the imaged body part, and the nature of the bone tissue for a patient.

The position-based likelihood, $p(w_{frag_k}|\mathbf{x}, k, \mathbf{M})$, is the likelihood of w_{frag_k} given the position $\mathbf{x}$, the k^{th} marker, and the set of markers $\mathbf{M}$. The position-based likelihood, $p(w_{frag_k}|\mathbf{x}, k, \mathbf{M})$, is computed as the ratio of the reciprocal of the distance between point $\mathbf{x}$ and marker M_k for the k^{th} bone fragment to the sum of the reciprocals of the individual distances between point $\mathbf{x}$ and all markers. The computation of $p(w_{frag_k}|\mathbf{x}, k, \mathbf{M})$ is shown in equation (59).

$$p(w_{frag_k}|\mathbf{x}, k, \mathbf{M}) = \frac{\frac{1}{dist(\mathbf{x}, M_k)}}{\sum_{i=1}^{K} \frac{1}{dist(\mathbf{x}, M_i)}} \tag{59}$$

where $dist(\mathbf{x}, M_i)$ is the shortest Euclidean distance between pixel $\mathbf{x}$ and marker M_i, see figure (71) . $dist(\mathbf{x}, M_i)$ is defined in equation (60):

$$dist(\mathbf{x}, M_i) = \min_{\forall \mathbf{x_j} \in M_i} \{||\mathbf{x} - \mathbf{x_j}||\} \tag{60}$$

where $\|\cdot\|$is the Euclidean distance. The computation of the position likelihood is based on the assumption that a bone pixel is most likely part of the bone fragment region which has the closest cortical pixels to that bone pixel. This assumption is based on the fact that bone structures are continuous, smooth and mostly covered by cortical tissue. Cortical tissue is represented by pixels with high intensity values in a CT image. These pixels are estimated in a CT image to create markers for bone fragment regions, refer to section (7.3.1). The spatial distance between a pixel and a marker reflects how close the pixel is to cortical pixels of the corresponding bone fragment. The position-based likelihood is considered a shape based probability for a bone fragment class.

Intensity-based likelihood, $p(w_{frag_k}|I(\mathbf{x}), k, \mathbf{M})$, is the likelihood of w_{frag_k} given the image intensity $I(\mathbf{x})$, the k^{th} marker, and the set of markers $\mathbf{M}$. The intensity-based likelihood, $p(w_{frag_k}|I(\mathbf{x}), k, \mathbf{M})$, is computed in equation (61).

$$p(w_{frag_k}|I(\mathbf{x}), k, \mathbf{M}) = \frac{1}{2}\exp(-|I(\mathbf{x}) - \mu_k(\mathbf{x})|) \tag{61}$$

where $\mu_k(\mathbf{x})$ is the location parameter of the Laplace distribution. The computation of intensity likelihood is based on the assumption that bone structures typically exhibit an exponential decaying of intensity in CT images from their cortex. So that, Laplace distribution provides a convenient way to model exponential distributions. Laplace distribution has relatively large values around $\mu_k(\mathbf{x})$, i.e., $|I(\mathbf{x}) - \mu_k(\mathbf{x})| \approx 0$, and has relatively small values for intermediate and large values of $|I(\mathbf{x}) - \mu_k(\mathbf{x})|$. The intensity-based likelihood is considered a tissue properties based probability for a bone fragment class.

The location parameter of the Laplace distribution $\mu_k(\mathbf{x})$ represents the expected intensity at point $\mathbf{x}$ due to the existence of marker M_k. $\mu_k(\mathbf{x})$ is computed in equation (62):

$$\mu_k(\mathbf{x}) = \frac{1}{|LS(d)|} \sum_{\mathbf{x}_i \in LS(d)} I(\mathbf{x}_i) \tag{62}$$

where d is the distance between pixel $\mathbf{x}$ and the k^{th} marker M_k, i.e., $d = dist(\mathbf{x}, M_k)$. $LS(d)$ is the set of all pixels in the bone region that satisfy $dist(\mathbf{x}_i, \mathbf{M}) - d = 0$. $LS(d)$ is computed as in equation (63).

$$LS(d) = \{\mathbf{x}_i | dist(\mathbf{x}_i, \mathbf{M}) - d = 0, \forall x_i \in D\} \tag{63}$$

The distance between a pixel $\mathbf{x}_i$ and the set of markers $\mathbf{M}$, i.e., $dist(\mathbf{x}_i, \mathbf{M})$, is calculated in equation (64):

$$dist(\mathbf{x}_i, \mathbf{M}) = \min_{\forall \mathbf{x_j} \in \mathbf{M}} ||\mathbf{x}_i - \mathbf{x_j}|| \tag{64}$$

where $\|\cdot\|$is the Euclidean distance. The computation of $\mu_k(\mathbf{x})$ is based on the assumption that the intensity values follow similar decaying curves as pixels move away from markers for different parts of the image, so that bone pixels that have the same distance to the markers set are expected to have the same intensity value, see figure (72) . This assumption is based on the fact that bones exhibit smooth structural changes and will have similar structure in CT scans which usually image small areas of the body.

The normalization factor, $p(I(\mathbf{x}), \mathbf{x}, \mathbf{M})$, ensures that the likelihood probability in the left-hand side of equation (58) is a valid probability which sums over all possible

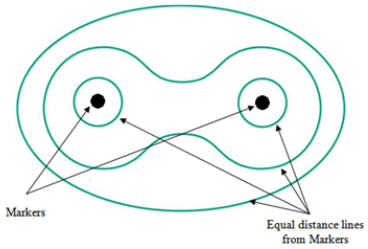


Figure 72: An illustration picture to show the equal distance lines from markers. Along each line, pixels are assumed to have the same intensity value.

hypotheses to one. This normalization factor is computed in equation (65):

$$p(I(\mathbf{x}), \mathbf{x}, \mathbf{M}) = \sum_{k=1}^{K} p(w_{frag_k} | I(\mathbf{x}), k, \mathbf{M})\, p(w_{frag_k} | \mathbf{x}, k, \mathbf{M}) \tag{65}$$

where K is the number of markers. The normalization factor is the sum of all likelihood multiplications for all bone fragment classes. Once the intensity-based likelihood, position-based likelihood, and the normalization factor are computed, the fragment probability is computed according to equation (58).

7.3.2.2 Context-Based Probability Computation

Context-based probability computation is the second stage of probability distribution computation. This stage takes three inputs: (1) the set of non-cortical bone pixels D, (2) the set of background pixels BG, (3) the set of markers $\mathbf{M}$, and (4) the depth threshold T_{depth}. There are two outputs of this stage: (1) non-"isthmus" context likelihood and (2) non-"isthmus" likelihood. These likelihoods are used to enhance the segmentation results for bone fragments within CT images.

The non-"isthmus" context likelihood measures how likely the geometric shape around pixel $\mathbf{x}$ assembles a non-"isthmus" bone area. The algorithm computes this likelihood by the percentage of the distance between a pixel and the edge of bone

area to the cut-off distance T_{depth}. The non-“isthmus” context likelihood is computed in equation (66):

$$p(c_{NIS}|\mathbf{x}) = \begin{cases} 1, & DT(\mathbf{x}) \geq T_{depth} \\ \frac{DT(\mathbf{x})}{T_{depth}}, & otherwise \end{cases} \tag{66}$$

where $p(c_{NIS}|\mathbf{x})$ is the non-“isthmus” context likelihood given $\mathbf{x}$, while, $DT(\mathbf{x})$ is the pixel distance from the boundary of the bone region and T_{depth} is the depth threshold. T_{depth} is set according to the expected minimum width of a bone fragment of interest for the user. The computation of $p(c_{NIS}|\mathbf{x})$ probability is based on the assumption that pixels located in ”isthmus” areas are very close to the boundary of the bone region because these areas have a small depth. So, the further a pixel is from the boundary of the bone region the lower its probability to assemble an”isthmus” bone area shape.

The non-”isthmus” likelihood defines how likely a bone pixel is located outside an ”isthmus” bone area. The non-”isthmus” likelihood is computed in equation (67).

$$p(w_{NIS}|\mathbf{x}) = \begin{cases} 1, & DT(\mathbf{x}) \geq T_{depth} \\ 0, & otherwise \end{cases} \tag{67}$$

where $p(w_{NIS}|\mathbf{x})$ is the non-“isthmus” likelihood given $\mathbf{x}$. The algorithm computes the non-”isthmus” likelihood as a binary value based on the distance of the pixel from the boundary of bone region. The algorithm uses T_{depth} as a cut-off value to assume whether the pixel $\mathbf{x}$ is located inside or outside an “isthmus” bone area.

The boundary of a bone region is determined by a collection of pixels which have at least a neighbor pixel that belongs to the background. The boundary of a bone

region, denoted as BBR, is computed in equation (68).

$$BBR = \{\mathbf{x} | \mathbf{x} \in D, \exists(\mathbf{y} \in NG(\mathbf{x}), \mathbf{y} \in BG)\} \tag{68}$$

where $NG(\mathbf{x})$ is the set of pixels that are neighbors to pixel $\mathbf{x}$.

The distance of a pixel from the boundary of a bone region is defined as the length of the shortest path that connects a bone pixel to the boundary such that this path does not include any marker pixel. Let $\mathbf{x}$ be a bone pixel and $\mathbf{y}$ be a boundary pixel, i.e., $\mathbf{x} \in D$ and $\mathbf{y} \in BBR$. Also, let $\pi(\mathbf{x}, \mathbf{y}) = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$ denote a path with length l that connects between $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{x}_1 = \mathbf{x}$, $\mathbf{x}_l = \mathbf{y}$, and $\mathbf{x}_i \notin \mathbf{M}$. Then, the distance of bone pixel $\mathbf{x}$ from the boundary BBR is computed in equation (69):

$$DT(\mathbf{x}) = \min_{\forall \mathbf{y} \in BBR} \left[\min_{\pi \in \Omega(\mathbf{x}, \mathbf{y})} |\pi(\mathbf{x}, \mathbf{y})| \right] \tag{69}$$

where $\Omega(\mathbf{x}, \mathbf{y})$ is the set of all paths from $\mathbf{x}$ to $\mathbf{y}$ and $|\pi(\mathbf{x}, \mathbf{y})|$ is the length of the path which corresponds to the number of pixels along the path. The computation of distances for bone pixels from the boundary of the bone region are performed using the wavefront propagation algorithm for distance transform described in [62].

7.3.2.3 PWT Probability distributions computation

PWT Probability distributions computation is the third stage of probability computation. In this stage, the algorithm takes three inputs: (1) fragment-based probability, (2) non-“isthmus” context likelihood, and (3) non-”isthmus” likelihood. The output of this algorithm is a set of probability distributions $\{f_k\}$ that computes how likely a bone fragment class w_{frag_k} given $\mathbf{x}$. The PWT probability distributions are

computed in equation (70):

$$f_k(\mathbf{x}) = p(w_{frag_k}|\mathbf{x}) = \frac{1}{2}\left[p(w_{NIS}|\mathbf{x})(p(w_{frag_k}|I(\mathbf{x}),\mathbf{x},k,\mathbf{M})+1)+(1-p(w_{NIS}|\mathbf{x}))p(c_{NIS}|\mathbf{x})\right] \quad (70)$$

where $f_k(\mathbf{x})$ is the probability function associated with marker M_k for pixel $\mathbf{x}$ and $\frac{1}{2}$ is a normalization factor. $f_k(\mathbf{x})$ is determined by the probability of whether bone fragment class k is true at pixel $\mathbf{x}$ or not. The algorithm uses the $p(w_{NIS}|\mathbf{x})$ to switch between fragment-based probability and non-"isthmus" context likelihood. If the $p(w_{NIS}|\mathbf{x})$ of that pixel is 1, the pixel is assumed to be a non-"isthmus" pixel, otherwise it is assumed an "isthmus" pixel. For a non-"isthmus" pixel, the algorithm computes the PWT probability depending on the fragment probability only, because the intensity value for this pixel is assumed to be reliable, i.e., assumed to have small blurring noise. Adding1 to the fragment probability shifts the PWT probability value at this pixel to the upper half of the probability range. For an "isthmus" pixel, the algorithm computes the probability by depending on the non-"isthmus" context likelihood only because $p(w_{NIS}|\mathbf{x}) = 0$ at this pixel. The multiplication by $\frac{1}{2}$ in equation (70) is a normalization factor to ensure that the PWT probability values are between 0 and 1. According to equation (70), the probability value for a non-"isthmus" pixel is between 0.5 and 1, while for an "isthmus" pixel, the probability value is between 0 and 0.5, see figure (73) . This ensures that non-"isthmus" pixels have higher probabilities than "isthmus" pixels.

7.3.3 Performing the PWT Algorithm

Performing the PWT algorithm is the fourth step of the bone fragment segmentation algorithm. This step takes four inputs: (1) the set of markers $\mathbf{M}$, (2) the set of

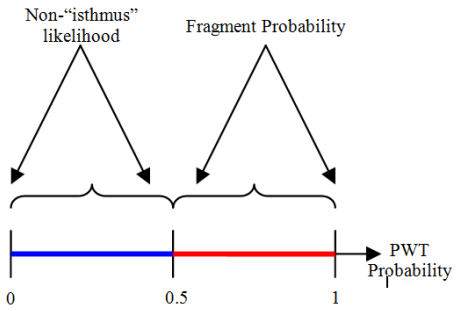


Figure 73: An illustration picture to show the mapping of fragment probability and non-"isthmus" context likelihood to generate probability distributions for the PWT algorithm. Fragment probability is mapped to the upper half of the probability range while the non-"isthmus" context likelihood is mapped to the lower half.

bone pixels D, (3) the set of background pixels BG, and (4) the set of probability distributions $\{f_k\}$. The output of this step is a collection of labels, one for each bone pixel. The collection of bone pixels that share the same label form a unique segment which represents a unique bone fragment in the image, see figure (74).

The PWT algorithm is performed as described in section (7.1.4). The algorithm starts by assigning a unique label for each marker and label its pixels with the same label. Then, the labels of the markers are expanded into bone pixels according to their probability distributions. Pixels with higher probability values are processed before pixels with lower probability values. Once the PWT algorithm finishes its operations, the collection of the labeled pixels represent the estimated bone fragments in the image and the background set BG contains the pixels that do not belong to any bone fragment.

The PWT algorithm assigns labels to none "isthmus" pixels before assigning labels to "isthmus" pixels because none "isthmus" pixels have higher probabilities. This order of assignment reduces the expansion of any incorrect labels for pixels in the narrow bone areas to none "isthmus" pixels. For "isthmus" pixels, the PWT algorithm

(a) (b)

Figure 74: Bone fragment segmentation result using the PWT algorithm, (a) original image, (b) segmentation results, each unique label is represented with a unique color. The white area represents the background points in BG.

expands labels by adding an additional layer of pixels neighboring those already labeled ones at each step, starting from the farthest pixels from the boundary of the bone region since they have higher probabilities, refer to equation (70). This order of assignment causes "isthmus" pixels to be divided evenly between neighboring segments.

CHAPTER 8: RESULTS OF BONE FRAGMENT SEGMENTATION ALGORITHM

This chapter describes the methods used to quantitatively evaluate the bone fragment segmentation 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 over 2D images and 3D images. For 2D images, the quantitative evaluation is performed by comparing results from a collection of segmentation algorithms, including results from the bone fragment 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 images. For 3D images, the quantitative evaluation is performed by comparing results from the bone fragment segmentation with results established as the ground truth. The segmentation evaluation techniques described in [7, 63, 53, 20] are used to evaluate the bone fragment segmentation using the PWT approach. These results allow users to compare this approach with other leading segmentation methods.

The patient data that is used for segmentation performance analysis is provided by [46]. The used data set includes three cases for tibia fractures where each case includes an image of a broken limb and an image of an unbroken limb. These fracture cases range from low energy fracture events such as 1.5 foot fall, to high energy fracture events such as a 50 mph car accident. The images are 3D CT scans in DICOM format

and 16-bit are used to express the CT numbers. Each volume contains of 81 to 302 contiguous axial slices with a slice thickness of 5mm. 2D images are generated by selecting slices from the 3D scans from axial, sagittal, and coronal view perspectives to represent normal and abnormal cases of intact and fractured shapes of bone fragments.

The proposed segmentation algorithm evaluation approach requires a definition of a ground truth segmentation for each image. For 2D images, the ground truth segmentation is generated by employing a human to manually segment each image. The set of human generated segmentations were taken as a collection of 2D "ground truth" examples. For 3D images, work in [46] extracted surfaces for tibia bone fragments for the cases of the dataset by employing humans to manually segment each image using a watershed segmentation tool. The set of the generated surfaces were taken as a collection of 3D "ground truth" examples. Evaluation is accomplished by comparing the segmentations produced by automatic algorithms with each of the "ground truth" examples.

A set of ten metrics serve to score the difference between two segmentations of a given image. 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 an image and ground truth segmentation of that image, then comparing segmentations generated by automatic algorithms with the ground truth segmentation. Using this comparison strategy, the bone fragment segmentation algorithm was ranked with respect to other leading segmentation algorithms using the same metrics.

8.1 Methodology for Evaluation

The evaluation for the proposed bone fragment segmentation using the PWT algorithm uses ten evaluation metrics taken from [7, 63, 53, 20] 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 nine metrics measure the segmentation difference by comparing regions of two segmentations using nine 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 bone fragment segmentation algorithm which are analyzed in detail afterward.

8.1.1 Experimental Setup for Comparative Evaluations

Comparative evaluations experiments were conducted to evaluate the performance of the bone fragment segmentation algorithm on 2D images and 3D images. For 2D images, the experiment compares the performance of the proposed bone fragment segmentation algorithm with other segmentation algorithms. While for 3D images, the experiment computes the evaluation metric that measures the segmentation difference with the ground truth examples by comparing the boundaries of two segments using a boundary difference score.

For 2D images, the experiment compares the bone fragment segmentation algorithm using the PWT algorithm, referred to as the PWT algorithm for abbreviation, with five segmentation algorithms: (1) Level set, [42], (2) Active contours, [100], (3) Region, [43], (4) Threshold, [58], and (5) Watershed, [50]. The implementations to

generate segmentations for the level set, active contours and region algorithms were provided by the work in [42], [100], and [43] respectively. The implementation for each of these algorithms takes as input a 2D image and provides as output closed contours for the segmented regions where each enclosed region represents a unique segment. The implementations to generate segmentations for threshold and watershed algorithms were provided by MATLAB software. For the threshold algorithm, the implementation takes as input a 2D image and provides as output a set of disjoint regions where each region represents a unique segment. For the watershed algorithm, the implementation takes two inputs a 2D image to be segmented and a marker image. The marker image specifies estimated locations for bone fragment segments. To compare fairly with other segmentation algorithms, a 2D image that contains the bone region estimated by the PWT algorithm was provided as input for other segmentation algorithms to generate their best result. Furthermore, the estimated locations for bone fragment segments by the PWT algorithm are provided as markers to the watershed algorithm. If multiple segments were generated by a segmentation algorithm for a single ground truth segment, i.e., oversegmentation, these segments are combined to simplify the computation of metrics and to measure the maximum region that a segmentation algorithm is able to segment regardless of the number of generated segments.

For 3D images, the experiment compares segmentations for the PWT algorithm with the ground truth examples provided by the work in [46]. The ground truth examples are surfaces for tibia fragments generated by triangulating segments that have been manually generated by a human. The comparison process generates surfaces for

the segments generated by the PWT algorithm. Then, it computes the evaluation metric that measures the segmentation difference by comparing the boundaries of the two surfaces using a boundary difference score.

8.1.2 Compute Evaluation Metrics

The performance analysis uses ten metrics proposed in [7, 63, 53, 20] to evaluate how well the PWT 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[101]. There are two extreme situations in the segmentation results: (1) the segmentation algorithm segments every pixel into a different segment, (2) the segmentation algorithm segments the whole image into a single segment. In these two circumstances, the values for each metric will not present the true quality of the results.

Segmentation comparison metrics quantify differences between two segmentation results. There are ten comparison metrics are used: (1) accuracy, (2) sensitivity, (3) specificity, (4) overlap, (5) precision, (6) recall, (7) F_1 score, (8) Hamming distance, (9) region rank, and (10) cut discrepancy. These metrics are taken from [7, 63, 53, 20] to compare segmentation results of an automated algorithm with ground truth segmentation. The first nine metrics compare segmentations by measuring differences between regions of two segmentations. While, the remaining metric compares segmentations by measuring differences between boundaries of two segmentations. These

metrics are used to evaluate how well a segmentation algorithm performs relative to a ground truth segmentation.

8.1.2.1 Region Based Segmentation Comparison Metrics

Region based metrics measure the difference between two segmentation results by comparing the regions of their segments. Region based metrics consists of nine metrics: (1) accuracy, (2) sensitivity, (3) specificity, (4) overlap, (5) precision, (6) recall, (7) F_1 score, (8) Hamming distance, and (9) region rank. These metrics compare two segmentation regions point-wise according to points membership in two segmentations of an image. Assume an image that represents objects and background is segmented by two algorithms producing the segmentations $S_1 = \{S_1^1, S_1^2, ..., S_1^m\}$ and $S_2 = \{S_2^1, S_2^2, ..., S_2^n\}$ with m and n segments, respectively, so that S_2 is the ground truth segmentation. S_1^i denotes the i^{th} segment region from segmentation S_1 , and S_2^i is the i^{th} segment region for S_2. Using this notation, the region based metrics are defined as follows.

Accuracy metric is the ratio of the shared points between two segmentations to the total number of points in an image [7], i.e.:

$$Accuracy = \frac{\sum_i ||S_2^i \cap S_1^{t(i)}||}{||S||}, \tag{71}$$

where " $\cap$ " is the set intersection operator, $||X||$ is the size of set X, and $t(i)$ is the index of the best corresponding segment in S_1 for segment S_2^i, i.e., $t(i) = \arg_k\{\max S_2^i \cap S_2^k\}$. $t(i)$ decides the index of the region from segmentation S_1 which has the largest overlap with region i from segmentation S_2. $||S||$ is a normalization

factor to the metric and it denotes the size of the set S which represents the total area of an image. Accuracy metric values have a range between 0 and 1, where 0 means objects and background are completely not segmented while 1 means objects and background are perfectly segmented. Accuracy metric, as defined in equation (71), indicates how good a segmentation algorithm is in accurately segmenting both objects and background. Accuracy metric provides a good evaluation for the performance of a segmentation algorithm when objects and background areas have similar sizes. However, accuracy metric has a shortcoming in cases where there is a big difference between the sizes of objects and background. In such cases, measurements of large areas are dominating so that measurements of small areas have a negligible effect on the computed accuracy metric value. For example, if the size of objects is too small compared to a background size, i.e., , then the accuracy metric is evaluated to:

$$\lim_{\sum_{i\neq b}||S_2^i||\to 0} Accuracy = \frac{||S_2^b \cap S_1^{t(b)}||}{||S_2^b||},$$

where b denotes the index of the background segment from segmentation S_2. S_2^b denotes the background segment region from segmentation S_2. $\sum_{i\neq b}||S_2^i||$ is the total size of segments for objects other than the background from segmentation S_2. In this case, the accuracy metric value is high if most of background pixels are labeled correctly by a segmentation algorithm regardless of the correctness of other segments of objects and this is not desired. This example occurs frequently in problems of segmenting bone fragments from CT images where bone fragments occupy small regions compared to the background region.

Sensitivity and specificity are two metrics defined in order to overcome the afore-

mentioned shortcoming of the accuracy metric by evaluating the segmentation of objects and background separately. Sensitivity metric is the ratio of the shared points of object segments between two segmentations to the total number of points in object regions in an image. Sensitivity metric is computed as:

$$Sensitivity = \frac{\sum_{i \neq b} ||S_2^i \cap S_1^{t(i)}||}{\sum_{i \neq b} ||S_2^i||}, \tag{72}$$

where b denotes the index of the background segment from segmentation S_2. $t(i)$ decides the index of the region from segmentation S_1 which has the largest overlap with region i from segmentation S_2. $\sum_{i \neq b} ||S_2^i||$ is the total size of segments for objects other than the background from segmentation S_2. From equation (72), sensitivity metric is equivalent to the accuracy metric when the size of a background approaches zero i.e. $\lim_{||S_2^b|| \to 0} Accurecy$, where S_2^b S_2^b denotes the background segment region from segmentation S_2. Sensitivity metric values have a range between 0 and 1, where 0 means objects are completely not segmented while 1 means all points in object regions from segmentation S_2 are inside object segments from segmentation S_1, i.e., ground truth segments from S_2 are completely inside their corresponding segments in S_1. Sensitivity metric measures the performance of a segmentation algorithm on extracting objects area without a background. A shortcoming of the sensitivity metric is that high values do not indicate a good segmentation. High sensitivity metric values occur when most points in object regions from ground truth segmentation S_2 are inside object segments from segmentation S_1. Having most points in object regions from segmentation S_2 are inside object segments from segmentation S_1 does not

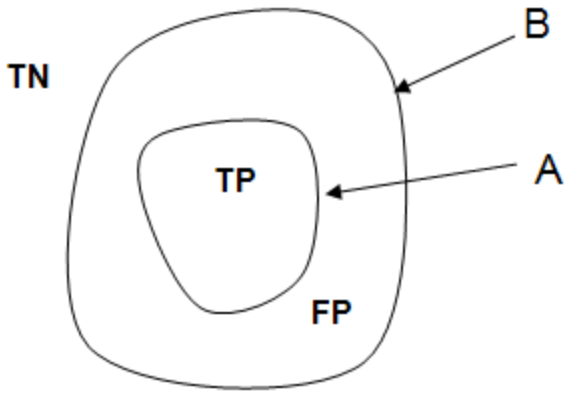


Figure 75: Classification result of two binary images A and B compared point-wise with image A as a reference to represent the truth model of the object. Image B represents an automatically segmented object that is larger than the truth model and completely covers it, therefore, $N_{FN} = 0$ and $Recall = 1$.

mean most points in object regions from segmentation S_1 are inside object segments from segmentation S_2. For example, figure (75) shows a case where a segmentation algorithm that has a leak problem in a background, i.e., incorrectly labels background pixels as object pixels, generates a segment that is larger than its corresponding ground truth segment while it covers completely the ground truth segment. In this case, $||S_2^i \cap S_1^{t(i)}|| = ||S_2^i||$ so that $Sensitivity = 1$. So, a high sensitivity metric value is obtained for a bad segmentation and this not desirable.

Specificity metric is the ratio of the shared points of a background segment between two segmentations to the number of points in background region in an image. Specificity metric is computed as:

$$Specificity = \frac{||S_2^b \cap S_1^{t(b)}||}{||S_2^b||}, \tag{73}$$

where b denotes the index of the background segment from segmentation S_2. $t(b)$ decides the index of the region from segmentation S_1 which has the largest over-

lap with background region b from segmentation S_2. From equation (73), specificity metric is equivalent to the accuracy metric when the size of objects approaches zero i.e. $\lim_{\sum_{i \neq b} ||S_2^i|| \to 0} Accurecy$. Specificity metric values have a range between 0 and 1, where 0 means the background is completely not segmented and 1 means it is perfectly segmented. Specificity metric measures the algorithm performance in extracting background area without objects. In a case where small objects exist on a large background and a small part of background pixels are labeled as objects, the effect of background errors is small on the specificity metric, while, incorrectly labeled pixels of background may have a considerable bad effect on the correctness of object segments and this is not desirable.

Overlap metric is defined as the ratio of the shared points of object segments between two segmentations to the number of points in the union of object segments of the two segmentations. Overlap metric is computed as:

$$Overlap = \frac{\sum_{i \neq b} ||S_2^i \cap S_1^{t(i)}||}{\sum_{i \neq b} ||S_2^i \cup S_1^{t(i)}||}, \tag{74}$$

where b denotes the index of the background segment from segmentation S_2. $t(i)$ decides the index of the region from segmentation S_1 which has the largest overlap with region i from segmentation S_2. " $\cup$ " is the set union operator. Overlap metric values have a range between 0 and 1, where 0 means objects are completely not segmented and 1 means objects are perfectly segmented. The definition of overlap metric overcomes the shortcoming of the sensitivity metric by considering in its computation the intersection points between segments of objects from segmentation S_1 with back-

ground segment from segmentation S_2. This is achieved by setting the normalization factor to the size of the union of object segments from S_1 and S_2 instead of the size of segments from S_2 alone. Segments from S_1 include correct and incorrect labeled pixels for objects. High values of the overlap metric indicate a good segmentation of objects in an image. However, these values do not indicate what kind of problems a segmentation algorithm has, for example, a leak problem and a problem of segmenting only parts of objects, referred to as partial segmentation. Furthermore, the overlap metric does not evaluate the performance of a segmentation algorithm in segmenting a background.

Precision metric is the ratio of the shared points of object segments between two segmentations to the total number of points in segmented object regions. Precision metric is computed as:

$$Precision = \frac{\sum_{i \neq b} ||S_2^i \cap S_1^{t(i)}||}{\sum_{i \neq b} ||S_1^{t(i)}||}, \tag{75}$$

where b denotes the index of the background segment from segmentation S_2. $t(i)$ decides the index of the region from segmentation S_1 which has the largest overlap with region i from segmentation S_2. $\sum_{i \neq b} ||S_2^{t(i)}||$ is the total number of points in segmented object regions other than the background from segmentation S_1. Precision metric values have a range between 0 and 1, where 0 means objects are completely not segmented and 1 means all points in object regions from segmentation S_1 are inside object segments from segmentation S_2, i.e., segments of S_1 are completely inside their corresponding ground truth segments in S_2. A shortcoming of the precision

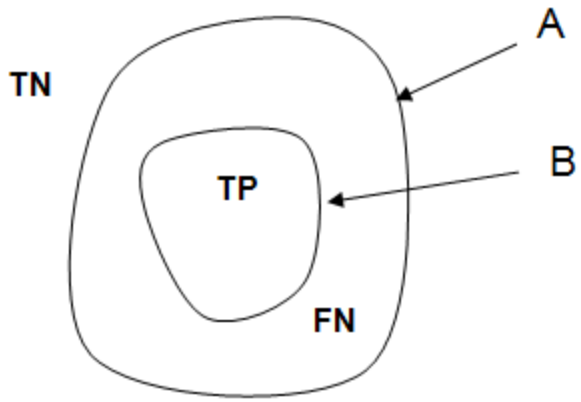


Figure 76: Classification result of two binary images A and B compared point-wise with image A as a reference to represent the truth model of the object. Image B represents an automatically segmented object that is completely inside the truth model object, therefore, $N_{FP} = 0$ and $Precesion = 1$.

metric is that high values do not indicate a good segmentation. High precision metric values occur when most points in object regions from segmentation S_1 are inside object segments from segmentation S_2. Having most points in object regions from segmentation S_1 are inside object segments from segmentation S_2 does not mean most points in object regions from segmentation S_2 are inside object segments from segmentation S_1. For example, figure (76) shows a case where a segmentation algorithm extracts part of a ground truth segment only, i.e., partial segmentation. In this case, $||S_2^i \cap S_1^{t(i)}|| = ||S_1^{t(i)}||$ so that $Precesion = 1$. So, a high precision metric value is obtained for a bad segmentation and this not desirable.

Recall metric is the ratio of the shared points of object segments between two segmentations to the total number of points in object regions in an image. Recall metric is computed as:

$$Recall = \frac{\sum_{i \neq b} ||S_2^i \cap S_1^{t(i)}||}{\sum_{i \neq b} ||S_2^i||}, \tag{76}$$

where b denotes the index of the background segment from segmentation S_2. $t(i)$ decides the index of the region from segmentation S_1 which has the largest overlap with region i from segmentation S_2. Recall metric as defined in equation (76) is the sensitivity metric as defined in equation (72). Recall metric values have a range between 0 and 1, where 0 means completely not segmented while 1 means all points in object regions from segmentation S_2 are inside object segments from segmentation S_1, i.e., ground truth segments from S_2 are completely inside their corresponding segments in S_1. The shortcomings for the recall metrics are similar to the shortcomings for the sensitivity metric.

F_1 score metric is a harmonic average of the precision metric and the recall metric. F_1 score metric is defined as:

$$F_1 = \frac{2 \times Precesion \times Recall}{Precesion + Recall} = \frac{2 \sum_{i \neq b} ||S_2^i \cap S_1^{t(i)}||}{\sum_{i \neq b} ||S_2^i|| + \sum_{i \neq b} ||S_1^{t(i)}||}, \tag{77}$$

where b denotes the index of the background segment from segmentation S_2. $t(i)$ decides the index of the region from segmentation S_1 which has the largest overlap with region i from segmentation S_2. This equation indicates no preference for either precision metric or recall metric, both metrics have equal importance. F_1 score metric values have a range between 0 and 1, where 0 means completely not segmented while 1 means objects and background are perfectly segmented. High values for F_1 score metric indicate a good segmentation, however, it does not indicate what kind of problems a segmentation algorithm has, for example, a leak problem and partial segmentation.

Hamming Distance (HD) metric measures the difference between two segmentation results in terms of the shared point membership of the two segmentations [20]. The directional Hamming distance from S_1 to S_2, $D_H(S_1 \Rightarrow S_2)$, is defined as:

$$D_H(S_1 \Rightarrow S_2) = \sum_i ||S_2^i \setminus S_1^{t(i)}|| \tag{78}$$

Where " $\setminus$ " is the set difference operator, and $t(i)$ is the index of the best corresponding segment in S_1 for segment S_2^i, i.e., $t(i) = \arg_k\{\max S_2^i \cap S_2^k\}$. $t(i)$ decides the index of the region from segmentation S_1 which has the largest overlap with region i from segmentation S_2. $D_H(S_1 \Rightarrow S_2)$ is the sum of the areas of the non-overlapping parts of S_1 segments which are found corresponding to some segments in S_2. Similarly, $D_H(S_2 \Rightarrow S_1)$ is the sum of the areas of the non-overlapping parts of S_2 segments which are found corresponding to some segments in S_1. Then, two measures are computed which are: the missing rate, R_m, and the false alarm rate, R_f. R_m and R_f are defined as follows:

$$R_m(S_1, S_2) = \frac{D_H(S_1 \Rightarrow S_2)}{||S||}, \tag{79}$$

$$R_f(S_1, S_2) = \frac{D_H(S_2 \Rightarrow S_1)}{||S||}, \tag{80}$$

where $||S||$ is a normalization factor to the metric and it denotes the size of the set S which represents the total area of objects. The Hamming distance is defined as the average of the missing rate and the false alarm rate, i.e.,:

$$HD(S_1, S_2) = 1 - \frac{1}{2}(R_m(S_1, S_2) + R_f(S_1, S_2)) \tag{81}$$

Where $HD(S_1, S_2)$ is the Hamming distance between S_1 and S_2. Metric values range from 0 to 1. The smaller the degree of mismatch, the closer the metric value is to 1. The Hamming distance metric relies on finding the correspondences between segments. This metric is meaningful when correspondences are correct and adds noise when they are not.

Region Rank (RR) metric is the average of the seven unique aforementioned region based metrics: (1) accuracy, (2) sensitivity (recall), (3) specificity, (4) overlap, (5) precision, (6) F_1score, and (7) Hamming distance. Region rank metric is computed as:

$$RR = \frac{(Accuracy + Sensitivity + Specificity + Overlap + Precision + F_1 + HD)}{7}, \tag{82}$$

where RR is the region rank metric. RR metric values have a range between 0 and 1, where 0 means objects and background are completely not segmented while 1 means objects and background are perfectly segmented. The seven unique region based metrics are used in computing the RR metric in order to use it in ranking a segmentation algorithm based on segmented regions. Each of the seven unique region based metrics indicates an aspect or more about a segmentation algorithm. Accuracy metric give a general indication of how good a segmentation algorithm is in segmenting both a background and objects. Sensitivity metric is good in indicating if a segmentation algorithm has a problem of partial segmentation but it does not indicate

if there is a leak problem. Specificity metric indicates how good a segmentation algorithm is in segmenting a background from an image but it does not indicate the effect of background segmentation errors on objects. Overlap metric indicates how good a segmentation algorithm in specifying objects in an image but it does not indicate if an algorithm has a leak problem or partial segmentation problem. Precision metric is good in indicating if a segmentation algorithm has a leak problem but it does not indicate if there is a partial segmentation problem. F_1 score metric gives a general indication if a segmentation algorithm has both a problem of partial segmentation and a leak problem but it does not specify if one of them exists only. HD metric gives another general indication if a segmentation algorithm has both a problem of partial segmentation and a leak problem but it does not specify if one of them exists only. An efficient segmentation algorithm scores high values in all region based metrics while a less efficient one scores low in one or more metrics. Hence, the usage of RR metric in ranking a segmentation algorithm based on segmented regions.

8.1.2.2 Boundary Based Segmentation Comparison Metric: Cut Discrepancy

Cut Discrepancy (CD) is a boundary-based metric which measures the difference between two segmentation results by comparing the boundaries of their segments. CD was originally proposed in [20] and it compares two segmentation boundaries by measuring the curve-to-curve distance from the set of points on one segmentation boundary to the set of points on the other segmentation boundary. CD is computed by summing the distances from the points along the cuts in the first segmentation to the closest cuts points in the second segmentation and vice versa. Assume an

image is segmented by two algorithms producing the segmentations S_1 and S_2 so that S_2 is the ground truth segmentation. Let C_1 and C_2 represent the segmentation boundaries for S_1 and S_2, respectively. The CD measures the sum of the distances between the segmentation boundaries C_1 and C_2. Suppose $\mathbf{p}_1$is a point that belongs to the boundary C_1, i.e., $\mathbf{p}_1 \in C_1$. Then, the distance between the point $\mathbf{p}_1 \in C_1$ and C_2 can be computed by finding the minimum distance between the point $\mathbf{p}_1$ and the segmentation boundary C_2 and it is defined as:

$$d(\mathbf{p}_1, C_2) = \min\{d(\mathbf{p}_1, \mathbf{p}_2),\ \forall \mathbf{p}_2 \in C_2\} \tag{83}$$

Where $d(\mathbf{p}_1, \mathbf{p}_2)$ is the Euclidean distance between the two points $\mathbf{p}_1$ and $\mathbf{p}_2$ and $d(\mathbf{p}_1, C_2)$ is the distance between the point $\mathbf{p}_1$ and C_2. The directional cut discrepancy for S_1 with respect to S_2 is defined as the mean of the point-to-boundary distances $d(\mathbf{p}_1, C_2)$, taken over all the points in the first segmentation boundary C_1 for all points $\mathbf{p}_1 \in C_1$, i.e., :

$$DCD(S_1 \Rightarrow S_2) = \frac{1}{N_1} \sum_{\forall \mathbf{p}_1 \in C_1} d(\mathbf{p}_1, C_2) \tag{84}$$

Where $DCD(S_1 \Rightarrow S_2)$ is the directional cut discrepancy for S_1 with respect to S_2. N_1 is the number of points in C_1. The cut discrepancy between S_1 and S_2 is defined as the mean of their directional cut discrepancy functions in both directions divided by the average of the Euclidean distance from a point in the surface to the centroid of the object , i.e.,:

$$CD(S_1, S_2) = \frac{DCD(S_1 \Rightarrow S_2) + DCD(S_2 \Rightarrow S_1)}{AvgRadius} \tag{85}$$

Where $CD(S_1, S_2)$ is the cut discrepancy between S_1 and S_2, and $AvgRadius$ is the average of the Euclidean distance from a point in the surface to the centroid of the object. The average of the Euclidean distance ensures the symmetry of the metric and reduces the scale effects. Metric values range from 0 to 1. The smaller the degree of mismatch, the closer the metric value is to 0. The benefit of the cut discrepancy metric is that it provides a measure of how well boundaries align.

8.1.3 Scores Used for Analysis

Collectively, there are nine 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 data set. To merge scores, the dataset 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. Accuracy as in equation (71),
2. Sensitivity (Recall) as in equation (72),
3. Specificity as in equation (73),
4. Overlap as in equation (74),
5. Precision as in equation (75),
6. F_1 score as in equation (77),

7. Hamming Distance, HD, as in equation (81),

8. Region Rank, RR, as in equation (82),

9. Cut Discrepancy, CD, as in equation (85).

The details of each score are discussed in section (8.1.2). Comparative evaluations are accomplished by observing these scores for the different segmentation approaches.

8.2 Evaluation and Analysis for 2D Images

Figure (77) shows image segmentation results for five different 2D images using seven 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 PWT algorithm. The first row of figure (77) shows the original input images, the remaining rows of figure (77) shows segmentation results for each of the seven different segmentation methods. These methods, from top-to-bottom, are 1) a human generated segmentation ("ground truth"), 2) the PWT segmentation, 3) the watershed segmentation, 4) the threshold segmentation, 5) the region segmentation, 6) the active contour segmentation, and 7) the level set segmentation. The segmentation results for these algorithms are shown in rows (2), (3), (4), (5), (6), (7), and (8) of figure (77), respectively. Table (7) shows the PWT settings that are used to segment images. The results shown suggest that the PWT algorithm can generate similar segmentation results as the ground truth examples.

Figure (78) shows a bar chart of the evaluation scores for the six automatic segmentation methods when evaluated over all the images. There are seven unique region based evaluation metrics that are reported: (1) accuracy, (2) sensitivity (recall), (3)

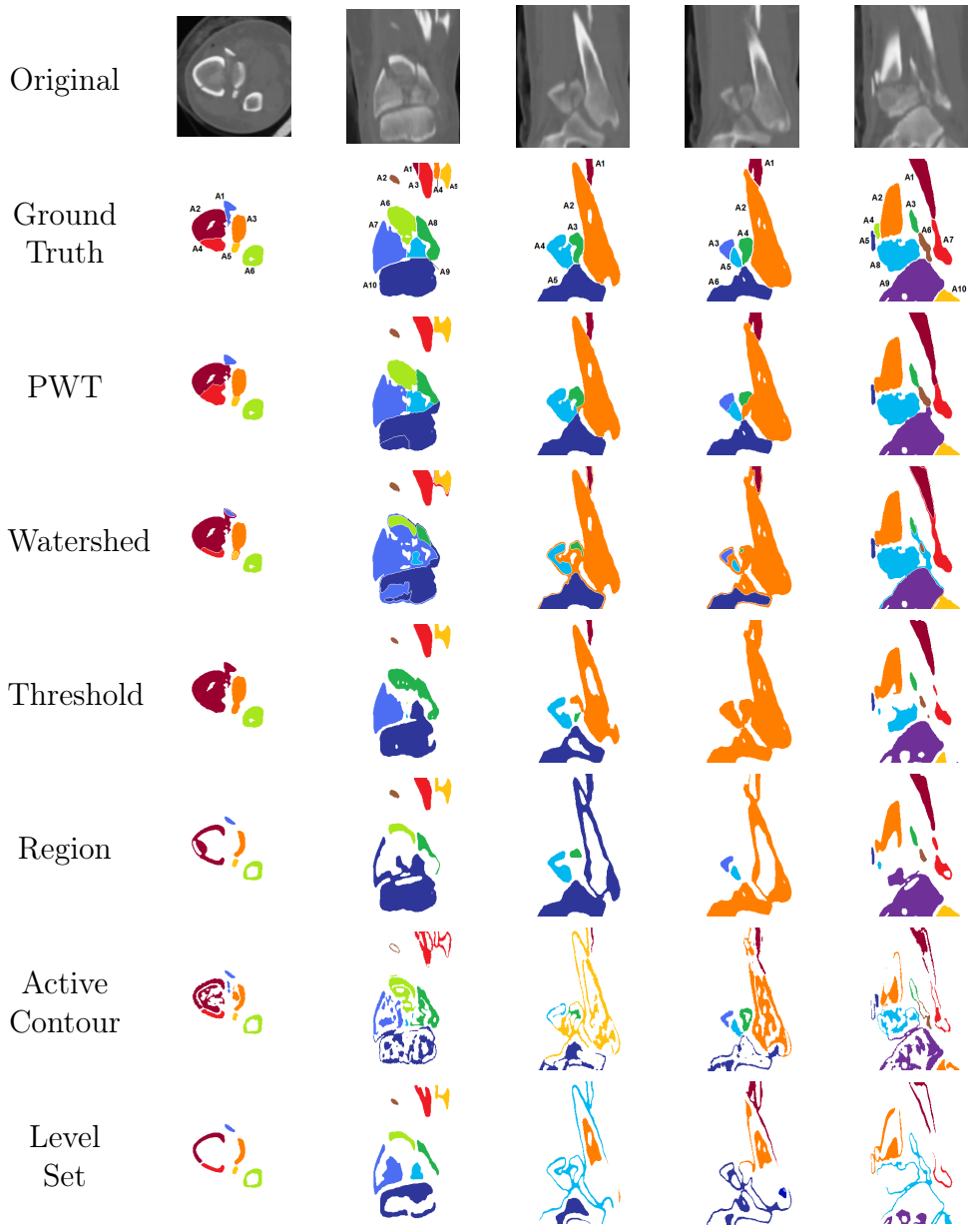


Figure 77: Image segmentation results for five different 2D CT images using seven different segmentation method. The first row shows the input images. The remaining rows show segmentation results generated by humans ("ground truth"), the PWT algorithm, the watershed algorithm , the threshold algorithm, the region algorithm [43], the active contour algorithm [100], and the level set algorithm [42], respectively.

Table 7: The settings for the PWT algorithm that are used in segmenting the original images (first row) in figure (77). The first row in the table is the number of the column of the segmented image in the indicated figure. Rows (2), (3), (4), and (5) are the used values for T_{cx}, T_{bone}, T_{Size}, and T_{depth} settings for the PWT algorithm, respectively.

Image Column#	2	3	4	5	6
T_{cx} (HU)	720	640	600	655	600
T_{bone} (HU)	140	140	140	150	140
T_{Size}(pixels)	15	15	15	15	15
T_{depth} (pixels)	15	15	15	15	15

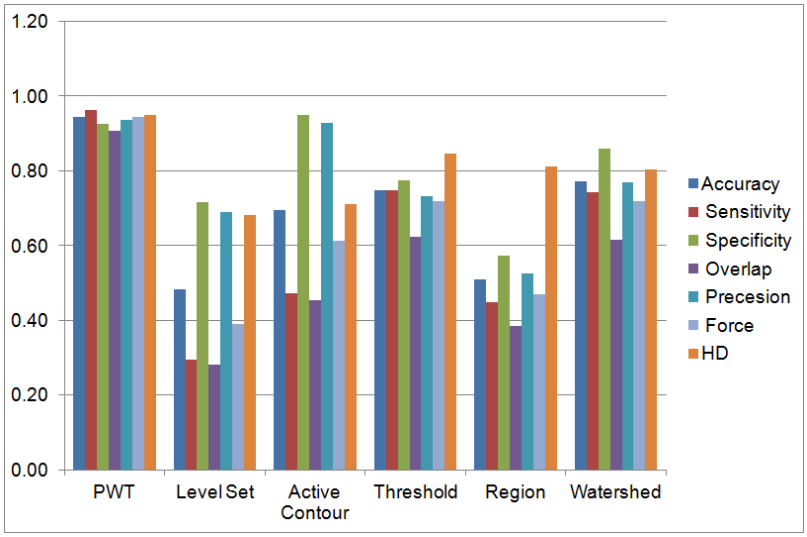


Figure 78: A chart for region based evaluation scores for segmentation results of all 2D images. The x-axis represents the region based metrics for different segmentation algorithms while the y-axis represents the measured metric value. Each algorithm is represented by group of bars. An efficient algorithm scores high values in all region based metrics while a less efficient one scores low in one or more metrics.

Table 8: Region based evaluation scores for segmentation results of all 2D images computed for the different segmentation algorithms used in performance comparison. All metrics values range from 0 to 1. The smaller the degree of mismatch, the closer the metric value to 1.

Method	Accuracy	Sensitivity	Specificity	Overlap	Precision	F_1	HD
Level Set	0.48	0.29	.72	.28	.69	.39	.68
Active Contour	.69	.47	.95	.45	.93	.61	.71
Region	.51	.45	.57	.38	.53	.47	.81
Threshold	.75	.75	.78	.62	.73	.72	.85
Watershed	.77	.74	.86	.62	.77	.72	.80
PWT	.94	.96	.93	.91	.94	.94	.95

Table 9: RR evaluation score for segmentation results of all 2D images computed for the different segmentation algorithms used in performance comparison. The metric value ranges from 0 to 1. The smaller the degree of mismatch, the closer the metric value to 1.

Method	RR
Level Set	.48
Active Contour	.66
Region	.52
Threshold	.74
Watershed	.75
PWT	.94

specificity, (4) overlap, (5) precision, (6) F_1 score, and (7) HD. Table (8) shows the measured seven region based metrics. The RR metric for segmentation algorithms is reported separately in a table and a chart. Figure (79) shows a chart of the RR metric while table (9) shows the RR metric for segmentation algorithms. The PWT segmentation has the score closest to the score for the ground truth examples among all the automatic segmentation algorithms. This indicates that the regions computed by the PWT algorithm are close to the regions specified in the ground truth segmentations. The boundary based comparison metric CD for segmentation algorithms is reported in a table and a chart. Figure (80) shows a chart of the CD metric while table (10)

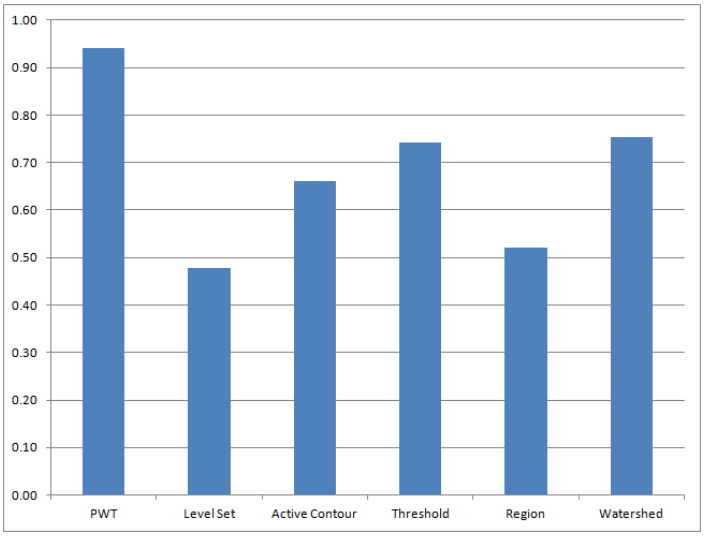


Figure 79: A chart for RR evaluation score for segmentation results of all 2D images to compare the performance of the different segmentation algorithms. The metric value ranges from 0 to 1. The smaller the degree of mismatch, the closer the metric value to 1.

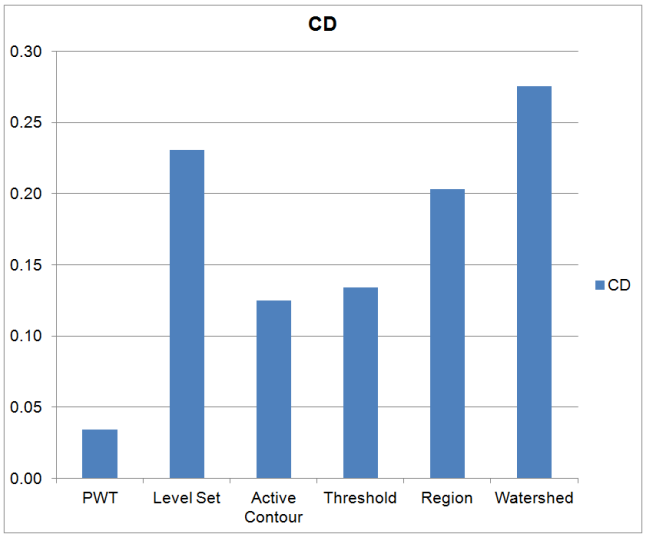


Figure 80: A chart for CD evaluation score for segmentation results of all 2D images to compare the performance of the different segmentation algorithms. The x-axis represents the CD metric for different segmentation algorithms while the y-axis represents the measured metric value. Each algorithm is represented by a single bar. The smaller the degree of mismatch, the closer the metric value to 0.

Table 10: CD evaluation score for segmentation results of all 2D images computed for the different segmentation algorithms used in performance comparison. Metric values range from 0 to 1, the smaller the degree of mismatch, the closer the metric value to 0.

Method	CD
Level Set	.23
Active Contour	.12
Region	.20
Threshold	.13
Watershed	.28
PWT	.03

shows the CD metric for segmentation algorithms. For this metric, the PWT segmentation has the score closest to the score for the ground truth examples among all the automatic segmentation algorithms. This indicates that the shapes of the regions computed by the PWT algorithm are close to the shapes of the regions specified in the ground truth segmentations. The small value of the computed CD metric for the PWT algorithm indicates that the PWT algorithm will not over-segment or under-segment most of models in the dataset. Hence, all evaluation scores indicate that the PWT algorithm tends to produce good segmentations when evaluated over all the 2D images in the test dataset.

8.2.1 Analysis

Analysis focuses on answering several important questions motivated by observations from the figures shown in (77 to 80) and tables in (8 to 10). The general questions posed are as follows:

1. Why does the PWT algorithm perform well on large fragments but not well on small ones?

2. Why does the PWT algorithm label some pixels as part of a fragment and they should not be, i.e., leak?

3. Why does the PWT algorithm generate incorrect number of segments for images in columns 2 and 5?

4. Why the level set and active contour algorithms tend to oversegment bone fragment regions?

5. Why does the watershed algorithm suffer from leak problem more than the other segmentation algorithms?

6. Why does the PWT algorithm typically out-perform the analyzed competing segmentation methods?

Answering these questions would provide a better understanding of the segmentation results as well as the strengths and weaknesses of the PWT algorithm.

The PWT algorithm performs well on large fragments but not well on small ones for two main reasons: (1) the ratio of the size of the cortex region with respect to the size of the segment is relatively small for small fragments and (2) the width of a bone region for a small fragment is relatively small. For the first reason, having a small cortex region in a fragment segment generates a relatively small marker with respect to the size of that intended segment. When the ratio of the size of a marker to the size of the intended segment is small, the pixels that belong to that intended segment may have large distances to the marker region. Large distances from a marker cause pixels at these distances to have low fragment likelihood values according to equation (58).

Pixels with low likelihood values for a marker are less likely labeled with the label of that marker especially when they are close to other markers. So, these pixels are highly likely labeled with incorrect labels. This problem is highly likely to appear for small fragments because they are usually generated from the internal side of a broken bone which mostly composed from cancellous tissue. For the second reason, having a bone region with small width increases the number of pixels that are likely to be inside "isthmus" bone areas in that region. Pixels that are assumed to be inside "isthmus" bone areas are given low probability values according to their likelihood of being inside an "isthmus" bone area, i.e., depends on non-"isthmus" context probability, refer to equation (70). For these pixels, the fragment probability which depends on intensity information and distances from markers is ignored. As a result, the pixels that are assumed to be inside "isthmus" bone areas are labeled last in the PWT algorithm and they are divided evenly between the neighboring segments causing some of these pixels to be incorrectly labeled.

The PWT algorithm may label incorrectly some bone pixels as part of a fragment while they should not be, for example, between fragments A_2 and A_3 in the image in the third column in figure (77). This issue highly likely occurs for pixels with very close distance to the background so to the boundary of the bone region. These pixels are assumed to be inside an "isthmus" bone area. Pixels that are assumed to be inside "isthmus" bone areas are given low probability values according to their likelihood of being inside an "isthmus" bone area, i.e., depends on non-"isthmus" context probability, refer to equation (70). For these pixels, the fragment probability which depends on intensity information and distances from markers is ignored. As a

result, the pixels that are assumed to be inside "isthmus" bone areas are labeled last in the PWT algorithm and they are divided evenly between the neighboring segments causing some of these pixels to be incorrectly labeled.

The PWT algorithm generates incorrect number of segments for images in columns 2 and 5. This issue occurs due to an error in detecting the correct number of markers for bone fragment segments in the marker detection stage, refer to section (7.3.1). This can be explained, in part, due to two reasons: (1) the existence of low intensity cortex areas within the cortex region and (2) the minimum intensity value considered a cortical tissue is too low value. For the first reason, the low intensity cortex areas causes algorithm to divide a single cortex region into multiple disjoint regions where each region is assumed to be a unique marker for a unique segment. In this case, too many markers are generated and a single bone fragment is represented by multiple segments, e.g., bone fragment A_{10} in the image in the second column in figure (77) is represented by two segments. For the second reason, if a too low value for the minimum intensity considered a cortical tissue is used, the marker detection algorithm may assume pixels that do not belong to a cortical tissue region as cortical pixels. These pixels may connect multiple disjoint cortex regions together. The algorithm considers these connected cortex regions as one and generates a single marker for them. Therefore, the PWT algorithm generates a single unique segment for this marker that represents multiple bone fragments, for example, bone fragments A_2 and A_4 in the image in the fifth column in figure (77) are represented by a single segment in the PWT segmentations.

The level set and active contour algorithms tend to oversegment bone fragment regions and labels small parts of the bone region more than any other algorithm. This can be explained, in part, due to the texture of bone fragments. Bone fragment regions in an image do not have a smooth intensity variation. This variation generates strong internal forces within bone regions that affect deformable models, refer to section (3.2.2.1). These internal forces move the initial contours for deformable models toward regions boundaries making a deformation process to split internal structures and textures of bone regions.

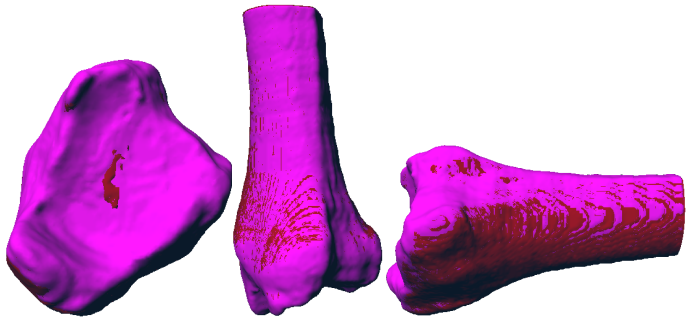
The watershed algorithm suffers from leak problem more than the other segmentation algorithm. This can be explained in part due to the loss of image information during the watershed segmentation process, refer to section (7.1.2). The watershed algorithm takes two inputs: an image and a collection of markers markers. The watershed algorithm modifies the input image to have regional minima at the specified marker positions. This modification may remove the boundaries and reduce the contrast between object regions. So that pixels at the lost boundaries may be incorrectly labeled. The labels of these incorrectly labeled pixels are then expanded to adjacent pixels. This process continues until no more pixels are none labeled, hence the problem.

The PWT algorithm typically out-perform the analyzed competing segmentation methods. This can be explained for three main reasons: (1) the use of unmodified intensity information in the computation of fragment probability, (2) the use of distance information between pixels and markers in the computation of fragment probability, and (3) the use of non-"isthmus" context probability to enhance the segmentation

result through “isthmus” bone area. For the first reason, the use of unmodified intensity information allows to not lose boundary information in the image. This allows the PWT algorithm to stop at boundaries and close proximity areas that separate between fragment regions, for example, the labeling of bone fragments A_1 and A_2 in the image in the fourth column in figure (77). For the second reason, distances between pixels and markers allows to reduce the leak problem by preventing overgrowing segments to far away pixels. A pixel is highly likely to belong to a segment that has a close marker than to belong to a segment that has a far away marker. For the third reason, the non-”isthmus” context probability is used to enhance the labeling process for the pixels that are inside ”isthmus” bone areas. These pixels usually separate between bone fragments and have inaccurate intensity information due to blurring. Therefore, these pixels are given low probability values to be processed last in the PWT algorithm. The PWT algorithm assigns these pixels labels after all pixels that are outside ”isthmus” bone areas are labeled. This procedure prevents the expansion of potentially incorrect labels to pixels outside ”isthmus” bone areas, for example, labeling of bone fragments A_2 and A_4 in the image in the first column in figure (77).

8.3 Evaluation and Analysis for 3D Images

In this section, the performance of the PWT bone fragment segmentation algorithm is quantitatively evaluated on 3D CT images. The performance is evaluated by applying the PWT algorithm on two 3D CT images of ankles to segment the tibia: one image for an unbroken tibia and a second records a fractured tibia after



(a) (b) (c)

Figure 81: The segmentation results for a 3D CT image for unbroken tibia using the PWT algorithm and its "ground truth" segmentation. The PWT segmentation result is shown in red, while the "ground truth" segmentation is shown in pink. They are shown together to allow for visual comparison of the results and to better understand the strengths and weaknesses of the PWT algorithm.

an injury. The estimated surfaces by the algorithm are compared to ground truth surfaces provided by the work in *[46]*. The performance is measured by computing the cut discrepancy metric which evaluates average distance between the points of the estimated surfaces and their corresponding points in the ground truth surfaces.

8.3.1 Unbroken Tibia Case

Figure (81) shows segmentation result for a 3D CT image for unbroken tibia using the PWT algorithm. The images show the segmentation result of the PWT algorithm for the tibia bone (in red) together with its "ground truth" segmentation (in pink) from three different view perspectives: axial, coronal, and saggital. They are shown together to allow for visual comparison of the results and to better understand the strengths and weaknesses of the PWT algorithm. The "ground truth" serves as the benchmark against which the PWT segmentation is compared.

The segmentation result of the PWT algorithm indicates that one region was generated to represent the unbroken tibia. The surface of this region looks very similar

to the ground truth one. The cut discrepancy metric for the segmentation result is 0.05. This low value confirms the similarity between the segmented region and the ground truth one.

8.3.2 Fractured Tibia Case

Figures (82-85) show segmentation result for a 3D CT image for a fractured tibia that was broken into three fragments using the PWT algorithm. Figure (82) shows all fragments together while each fragment is viewed separately in figures (83-85). The segmentation result is shown from three different view perspective: axial, saggital, and coronal. The segmentation result and the human generated segmentation, i.e., "ground truth", are shown together to allow for visual comparison of the results and to better understand the strengths and weaknesses of the PWT algorithm. The "ground truth" serves as the benchmark against which the PWT segmentation is compared. Different segmented regions are shown in different colors.

The segmentation result indicates that five regions were generated to represent the three fracture tibia fragments. The cut discrepancy metric for each fragment is shown in table (11). There are two notes about the PWT segmentation algorithm: (1) it may not perform well on segmenting small bone fragments and (2) it may generates an incorrect number of bone fragments.

For the first note, the proposed segmentation algorithm may not perform well in segmenting small bone fragment for two main reasons: (1) the ratio of the size of the cortex region with respect to the size of the segment is relatively small for small fragments and (2) the width of a bone region for a small fragment is relatively small.

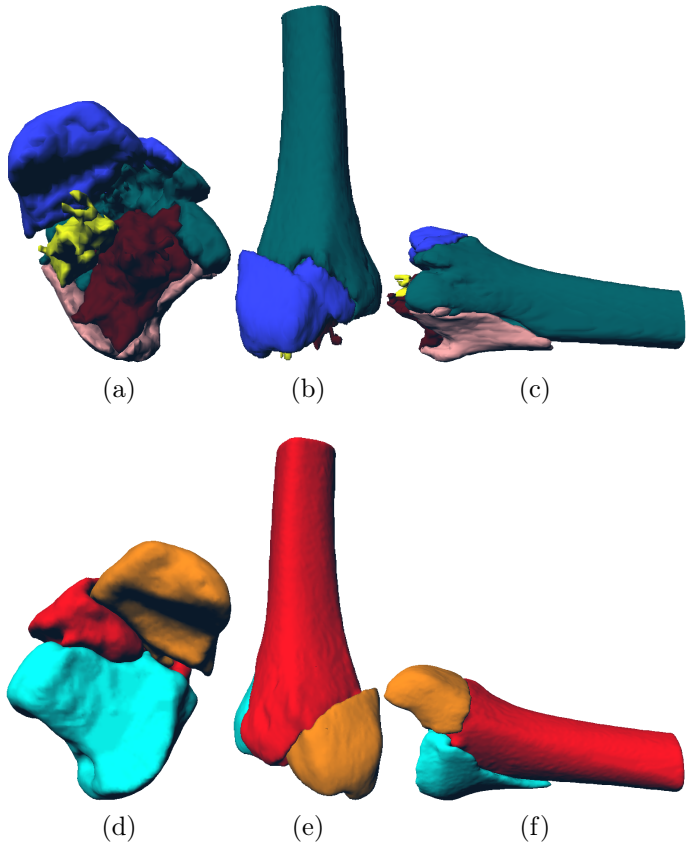
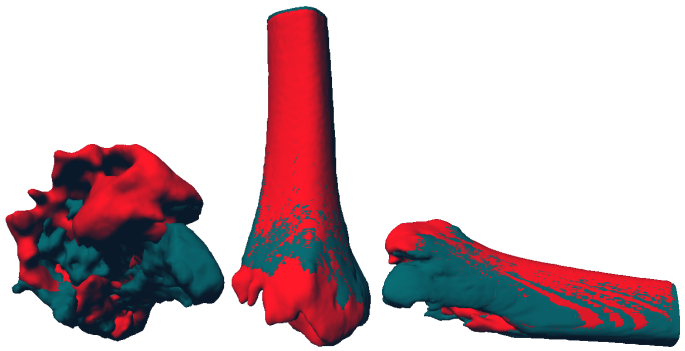


Figure 82: The segmentation results for a 3D CT image for a broken tibia using the PWT algorithm and its "ground truth" segmentation. Figures (a-c) show the segmentation result for all fractured tibia fragments from three different view perspective: axial, saggital, and coronal, respectively. Figures (d-f) show the human generated segmentation which is treated as "ground truth" from three different view perspective: axial, saggital, and coronal, respectively. Different segmented regions are shown in different colors.

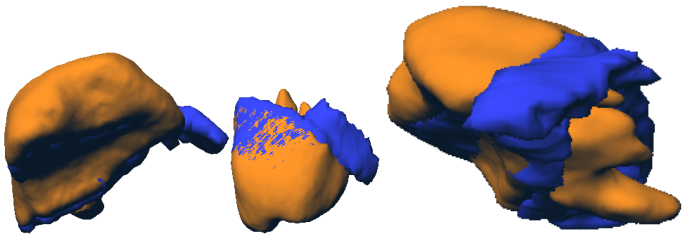
Table 11: The cut discrepancy metric values measured for the segmentation result for a 3D CT image for a broken tibia that was fractured into three fragments. The first, second, and third fragments are shown in figures (83-85), respectively.

Fragment	CC
First	0.09
Second	0.22
Third	0.38



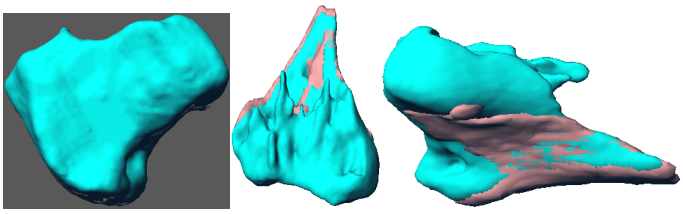
(a) (b) (c)

Figure 83: The segmentation results for the first fragment of three within a 3D CT image for a broken tibia using the PWT algorithm and its corresponding "ground truth" segmentation. Figures (a-c) show the segmentation result from three different view perspective: axial, saggital, and coronal, respectively. The PWT segmentation result is shown in dark blue, while the "ground truth" segmentation is shown in red.



(a) (b) (c)

Figure 84: The segmentation results for the second fragment of three within a 3D CT image for a broken tibia using the PWT algorithm and its corresponding "ground truth" segmentation. Figures (a-c) show the segmentation result from three different view perspective: axial, saggital, and coronal, respectively. The PWT segmentation result is shown in blue, while the "ground truth" segmentation is shown in orange.



(a) (b) (c)

Figure 85: The segmentation results for the third fragment of three within a 3D CT image for a broken tibia using the PWT algorithm and its corresponding "ground truth" segmentation. Figures (a-c) show the segmentation result from three different view perspective: axial, saggital, and coronal, respectively. The PWT segmentation result is shown in blue, while the "ground truth" segmentation is shown in pink.

For the first reason, having a small cortex region in a fragment segment generates a relatively small marker with respect to the size of that intended segment. When the ratio of the size of a marker to the size of the intended segment is small, the pixels that belong to that intended segment may have large distances to the marker region. Large distances from a marker cause pixels at these distances to have low fragment likelihood values according to equation (58). Pixels with low likelihood values for a marker are less likely labeled with the label of that marker especially when they are close to other markers. So, these pixels are highly likely labeled with incorrect labels. This problem is highly likely to appear for small fragments because they are usually generated from the internal side of a broken bone which mostly composed from cancellous tissue. For the second reason, having a bone region with small width increases the number of pixels that are likely to be inside "isthmus" bone areas in that region. Pixels that are assumed to be inside "isthmus" bone areas are given low probability values according to their likelihood of being inside an "isthmus" bone area, i.e., depends on non-"isthmus" context probability, refer to equation (70). For these pixels, the fragment probability which depends on intensity information and distances from markers is ignored. As a result, the pixels that are assumed to be inside "isthmus" bone areas are labeled last in the PWT algorithm and they are divided evenly between the neighboring segments causing some of these pixels to be incorrectly labeled.

For the second note, the proposed segmentation algorithm may generate an incorrect number of segments due to an error in detecting the correct number of markers for bone fragment segments in the classification process stage, refer to section (7.3.1).

This can be explained, in part, due to two reasons: (1) the existence of low intensity cortex areas within the cortex region and (2) the minimum intensity value considered a cortical tissue is too low value. For the first reason, the low intensity cortex areas causes algorithm to divide a single cortex region into multiple disjoint regions where each region is assumed to be a unique marker for a unique segment. In this case, too many markers are generated and a single bone fragment is represented by multiple segments as in figures (84-85). For the second reason, if a too low value for the minimum intensity considered a cortical tissue is used, the marker detection algorithm may assume pixels that do not belong to a cortical tissue region as cortical pixels. These pixels may connect multiple disjoint cortex regions together. The algorithm considers these connected cortex regions as one and generates a single marker for them. Therefore, the PWT algorithm generates a single unique segment for this marker that represents multiple bone fragments.

8.4 Conclusion

In this chapter, the bone fragment segmentation using the PWT algorithm is quantitatively evaluated by comparing its results with ground truth segmentation results using ten evaluation metrics. A similar evaluation is done for five other algorithms on 2D images. The segmentation results are also studied to explore how the PWT algorithm segments 2D images. Overall, the evaluation scores indicate that the PWT algorithm is more similar to the ground truth than other segmentation algorithms. The geometric segmentation properties for segmented boundaries and regions generated by the PWT algorithm are also close to the ground truth. However, the PWT

algorithm has some shortcomings. Specifically, it tends to incorrectly label pixels that are very close to background and to in accurately specify the boundaries for small bone fragment regions.

CHAPTER 9: CONCLUSION AND FUTURE WORK

The presented system is capable of estimating the inverse mechanics of highly comminuted bone fractures from 3D CT images of fracture cases, which is heretofore unsolved problem in the existing systems for fractures analysis. The system estimates how a limb fractured in terms of what fracture the limb and how the bone fragments moved over time. The system represents a unique integration of image processing techniques with existing dynamics simulation tools. The system is provided with a 3D CT image of a fractured limb and another 3D CT image for an unfractured limb. The system extracts data from these images using the image processing techniques. The system uses these data to build a virtual dynamics model of the fractured limb. This model represents fracture bone fragments, soft tissues, and a strike object as they existed before the fracture event occurred. The dynamics model is assigned physical attributes to reflect physical properties of the real fractured limb. The system then estimates the fracture event by hitting the dynamics model of the limb with the strike object. The system uses dynamics simulator to approximate the physical behavior of the objects in a fracture event. The system iteratively modifies the velocity for the strike object in order to find the best fracture simulation. The estimated fracture simulation is then analyzed by a user to explore the space of plausible solutions to understand how a fracture event may have occurred. While it is intuitive that a system like this could improve surgical treatment, this has not been proven as a

clinically effective tool. However, due to the limited number of experiments, solid conclusive statements about clinical utility of the system of inverse mechanics for highly comminuted fractures cannot be made. This system is a significant advancement toward improving the treatment of comminuted tibial plafond fractures. Estimating the inverse mechanics of a fracture provides heretofore unavailable understanding of how bone fragments moved from their original anatomical positions to their fractured positions as presented by a fracture case. Such understanding may help physicians improve the accuracy of bone fragments treatment decisions.

9.1 Future Work

Although the presented prototypical system to compute the mechanics of a highly comminuted fracture offers a powerful new tool for knowing the estimated trajectory that each fragment had taken to move from the their anatomic positions to their fractured position in a fracture, there are some limitations that need to be taken into account for future research before the system can be used in a clinical setting. The most important limitation of the system is the lack of the evaluation information from real world users to the system. Ideally, the real world user should be a trained orthopedic physician. A user study should be conducted to evaluate the system. The user study should attempt to answer the following set of questions: (1) does the tool to compute the mechanics of a fracture event benefit the users? If yes, how do users benefit from it, and in what aspects? (2) Is there any difference between the paths used in manual reconstruction surgery and trajectory paths estimated by the system? If yes, what is the difference? And why there is a difference? (3) What things can be

done to further improve the system? The feedback and suggestions for improving the system from the user study should be taken into consideration as the system evolves to suit these needs. Another limitation of the current system is the assumption that a healthy bone template is available as a reference from the patient. In practice, we may not have a healthy bone template due to cost limitation that only the fractured limb is scanned or the patient may have broken both limbs [46]. Future work may investigate how to automatically generate a generic 3D template bone for each fracture case depending on the biological information of the patient, for example, age, sex, height and weight. Another limitation of the current system is that it does not consider some important types of soft tissues around the bone and their connection to the bone in the simulation process, for example, tendon and ligament. Consideration of this information may improve the accuracy of the system to generate fracture patterns similar to the one observed in the fractured limb. Another limitation of the current bone segmentation algorithm used in the system is that it does not consider the 3D gradient of intensities for bone pixels in the probability computation. This gradient information may help the algorithm to detect the boundaries of bone fragments in the CT image especially for small fragments, so that the leak problem may be reduced further. Finally, future may seek to investigate the relationship between the estimated displaced volume of the soft tissue and the severity of a fracture. The severity of a fracture helps physician in determining a sequence of best-practices for obtaining the best possible prognosis for the patient.

REFERENCES

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

[2] M. S. Atkins and B. T. Mackiewich. Handbook of medical imaging. chapter Fully automated hybrid segmentation of the brain, pages 171–183. Academic Press, Inc., Orlando, FL, USA, 2000.

[3] Z. B, W. A, and S. Y. Improving inter-fragmentary alignment for virtual 3d reconstruction of highly fragmented bone fractures. *SPIE Medical Imaging*, 2009.

[4] L. Ballerini and L. Bocchi. Bone segmentation using multiple communicating snakes. *SPIE International Symposium Medical Imaging*, pages 1621–1628, 2003.

[5] I. Bankman. *Handbook of medical imaging: processing and analysis*. Academic Press series in biomedical engineering. Academic Press, 2000.

[6] F. Barber and S. Fischer. *Surgical Techniques for the Shoulder and Elbow*. Thieme, 2011.

[7] H. Barrett and W. Swindell. *Radiological Imaging: The Theory of Image Formation, Detection, and Processing*. Number v. 1. Elsevier Science, 1996.

[8] R. Beare and G. Lehmann. The watershed transform in itk - discussion and new developments. *The Insight Journal*, January - June 2006.

[9] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *In NIPS*, pages 831–837, 2000.

[10] S. Beucher. *Segmentation d'Images et Morphologie Mathematique*. PhD thesis, Ecole Nationale Superieure des Mines de Paris, 1990.

[11] W. Birkfellner. *Applied Medical Image Processing, Second Edition: A Basic Course*. Taylor & Francis, 2014.

[12] M. Bomans, K.-H. Hohne, U. Tiede, and M. Riemer. 3-d segmentation of mr images of the head for 3-d display. *Medical Imaging, IEEE Transactions on*, 9(2):177 –183, jun 1990.

[13] N. Boukala, E. Favier, B. Laget, and P. Radeva. Active shape model based segmentation of bone structures in hip radiographs. In *Industrial Technology, 2004. IEEE ICIT '04. 2004 IEEE International Conference on*, volume 3, pages 1682 – 1687 Vol. 3, dec. 2004.

[14] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105 –112 vol.1, 2001.

[15] Y. Boykov and O. Veksler. Graph Cuts in Vision and Graphics: Theories and Applications. In N. Paragios, Y. Chen, and O. Faugeras, editors, *Handbook of Mathematical Models in Computer Vision*, chapter 5, pages 79–96. Springer US, New York, 2006.

[16] M. Cabezas Grebol, A. Oliver, X. Llado, J. Freixenet, and M. Bach Cuadra. Review on Atlas-based Segmentation of Magnetic Resonance Brain Images. *Computer Methods and Programs in Biomedicine*, (Accepted for publication), 2011.

[17] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22:61–79, 1995.

[18] V. Chalana and Y. Kim. A Methodology for Evaluation of Boundary Detection Algorithms on Medical Images. *IEEE Transactions on medical imaging*, 16(5), Oct. 1997.

[19] H. Chen and A. K. Jain. Tooth contour extraction for matching dental radiographs. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, ICPR '04, pages 522–525, Washington, DC, USA, 2004. IEEE Computer Society.

[20] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009.

[21] Y. Chen, X. Ee, W. K. Leow, and T. S. Howe. T.s.: Automatic extraction of femur contours from hip x-ray images. In *In: Proc. ICCV Workshop on Computer Vision for Biomedical Image Applications*, pages 200–209, 2005.

[22] R. R. Chowdhury A, Bhandarkar S. Virtual multi-fracture craniofacial reconstruction using computer vision and graph matching. *Computerized Medical Imaging and Graphics*, 2009.

[23] K. A. Cimerman M. Preoperative planning in pelvic and acetabular surgery: The value of advanced computerised planning modules. *Injury*, 38:442–449, 2007.

[24] D. Cofer, G. Cymbalyuk, J. Reid, Y. Zhu, W. J. Heitler, and D. H. Edwards. Animatlab: A 3d graphics environment for neuromechanical simulations. *Journal of Neuroscience Methods*, 187(2):280 – 288, 2010.

[25] P. Combettes and J.-C. Pesquet. In H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer Optimization and Its Applications, pages 185–212. Springer New York, 2011.

[26] L. Cueru, A. M. Trunfio sfarghiu, Y. Bala, B. Depalle, Y. Berthier, and H. Follet. Mechanical and physicochemical multiscale analysis of cortical bone. *Computer Methods in Biomechanics and Biomedical Engineering*, 14(sup1):223–225, 2011.

[27] M. Droske, B. Meyer, M. Rumpf, and C. Schaller. An adaptive level set method for medical image segmentation. In *Proceedings of the 17th International Conference on Information Processing in Medical Imaging*, IPMI '01, pages 416–422, London, UK, UK, 2001. Springer-Verlag.

[28] M. S. Dryden. Complicated skin and soft tissue infection. *Journal of Antimicrobial Chemotherapy*, 65(suppl 3):iii35–iii44, 2010.

[29] M. S. Farvid, T. W. K. Ng, D. C. Chan, P. H. R. Barrett, and G. F. Watts. Association of adiponectin and resistin with adipose tissue compartments, insulin resistance and dyslipidaemia. *Diabetes, Obesity and Metabolism*, 7(4):406–413, 2005.

[30] T. Feeman. *The Mathematics of Medical Imaging: A Beginners Guide*. Springer Undergraduate Texts in Mathematics and Technology. Springer, 2009.

[31] J. Feng, W.-C. Lin, and C.-T. Chen. Epicardial boundary detection using fuzzy reasoning. *Medical Imaging, IEEE Transactions on*, 10(2):187 –199, jun 1991.

[32] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[33] A. Goshtasby and D. Turner. Segmentation of cardiac cine mr images for extraction of right and left ventricular chambers. *Medical Imaging, IEEE Transactions on*, 14(1):56 –64, mar 1995.

[34] V. Grau, A. Mewes, M. Alcaniz, R. Kikinis, and S. Warfield. Improved watershed transform for medical image segmentation using prior information. *Medical Imaging, IEEE Transactions on*, 23(4):447 –458, april 2004.

[35] D. Han, J. Bayouth, Q. Song, A. Taurani, M. Sonka, J. Buatti, and X. Wu. Globally optimal tumor segmentation in PET-CT images: a graph-based cosegmentation method. *Information processing in medical imaging proceedings of the conference*, 22:245–256, 2011.

[36] D. Han, X. Wu, and M. Sonka. Optimal multiple surfaces searching for video/image resizing - a graph-theoretic approach. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1026 –1033, 29 2009-oct. 2 2009.

[37] M. Ibrahim and Sezan. A peak detection algorithm and its application to histogram-based image data reduction. *Computer Vision, Graphics, and Image Processing*, 49(1):36 – 51, 1990.

[38] A. K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[39] Y. Jiang and P. Babyn. X-ray bone fracture segmentation by incorporating global shape model priors into geodesic active contours. *International Congress Series*, 1268(0):219 – 224, 2004.

[40] Z. Kai, K. Bin, K. Yan, and Z. Hong. Auto-threshold bone segmentation based on CT image and its application on CTA bone-subtraction. In *Photonics and Optoelectronic (SOPO)*, pages 1–5, 2010.

[41] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[42] C. Li, R. Huang, Z. Ding, C. Gatenby, D. N. Metaxas, and J. C. Gore. A level set method for image segmentation in the presence of intensity inhomogeneities with application to MRI. *IEEE Trans. Image Process.*, 20(7):2007–2016, July 2011.

[43] C. Li, C.-Y. Kao, J. C. Gore, and Z. Ding. Minimization of region-scalable fitting energy for image segmentation. *Image Processing, IEEE Transactions on*, 17(10):1940–1949, Oct. 2008.

[44] K. Li, X. Wu, D. Chen, and M. Sonka. Optimal surface segmentation in volumetric images-a graph-theoretic approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(1):119 –134, jan. 2006.

[45] Z. Lin, J. Jin, and H. Talbot. Unseeded region growing for 3D image segmentation. In *Selected papers from the Pan-Sydney workshop on Visualisation - Volume 2*, VIP '00, pages 31–37, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.

[46] P. Liu. *A System For Computational Analysis And Reconstruction Of 3D Comminuted Bone Fractures*. PhD thesis, The University of North Carolina At Charlotte, 2012.

[47] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM, 1987.

[48] G. Manos, A. Cairns, I. Rickets, and D. Sinclair. Segmenting radiographs of the hand and wrist. *Computer Methods and Programs in Biomedicine*, 43(3):227 – 237, 1994.

[49] D. Marr and E. Hildreth. Theory of Edge Detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207(1167):187–217, 1980.

[50] F. Meyer. Un algorithme optimal de ligne de partage des eaux. In *Reconnaissance des Formes et Intelligence Artificielle, 8e congr'es*, pages 847–857, Lyon-Villeurbanne, 1991. AFCET.

[51] F. Meyer. Integrals and gradients of images. pages 200–211, 1992.

[52] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1):113 – 125, 1994.

[53] M. R. K. Mookiah, U. R. Acharya, C. K. Chua, L. C. Min, E. Ng, M. M. Mushrif, and A. Laude. Automated detection of optic disk in retinal fundus images using intuitionistic fuzzy histon segmentation. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 227(1):37–49, 2013.

[54] T. Mullen and E. Coumans. *Bounce, Tumble, and Splash!: Simulating the Physical World with Blender 3D*. Serious skills. Wiley, 2008.

[55] I. J. Myung. Tutorial on maximum likelihood estimation. *J. Math. Psychol.*, 47(1):90–100, Feb. 2003.

[56] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(12):1163 –1173, dec 1996.

[57] L. O'Gorman and A. C. Sanderson. The converging squares algorithm: An efficient multidimensional peak picking method. In *Proceedings of the 1983 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '83)*, volume 8, pages 112 – 115, April 1983.

[58] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, Jan. 1979.

[59] F. Paulano, J. Jiménez, and R. Pulido. 3D segmentation and labeling of fractured bone from ct images. *The Visual Computer*, pages 1–10, 2014.

[60] G. Peersman, R. Laskin, J. Davis, M. Peterson, and T. Richart. Prolonged operative time correlates with increased infection rate after total knee arthroplasty. *HSS Journal*, 2(1):70–72, 2006.

[61] R. Pohle and K. D. Toennies. Segmentation of medical images using adaptive region growing. In M. Sonka & K. M. Hanson, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 4322 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 1337–1346, July 2001.

[62] F. Porikli and T. Kocak. Fast distance transform computation using dual scan line propagation. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6496 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Feb. 2007.

[63] D. M. W. Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Technical Report SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia, 2007.

[64] J. B. T. M. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41:187–228, 2001.

[65] J. Rogowska, K. Batchelder, G. S. Gazelle, E. F. Halpern, W. Connor, and G. L. Wolf. Evaluation of selected two-dimensional segmentation techniques for computed tomography quantitation of lymph nodes. *Investigative Radiology*, 31(3):138–145, 1996.

[66] A. Rosenfeld and A. Kak. *Digital picture processing.* Number v. 1 in Computer science and applied mathematics. Academic Press, 1982.

[67] C. B. Ruff and F. P. Leo. Use of computed tomography in skeletal structure research. *American Journal of Physical Anthropology*, 29(S7):181–196, 1986.

[68] J. C. Russ. *Image Processing Handbook, Fourth Edition.* CRC Press, Inc., Boca Raton, FL, USA, 4th edition, 2002.

[69] P. K. Sahoo, S. Soltani, A. K. Wong, and Y. C. Chen. A survey of thresholding techniques. *Comput. Vision Graph. Image Process.*, 41(2):233–260, Feb. 1988.

[70] A. Sands, L. Grujic, D. Byck, J. Agel, S. Benirschke, and M. Swiontkowski. Clinical and functional outcomes of internal fixation of displaced pilon fractures. *Clin Orthop*, (347):131–7, 1998.

[71] W. Seggl, R. Szyszkowitz, and W. Grechenig. Tibial pilon fractures. *Current Orthopaedics*, 13(1):42–52, 1999.

[72] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1999.

[73] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, Jan. 2004.

[74] W. Shadid and A. Willis. Bone fragment segmentation from 3d ct imagery using the probabilistic watershed transform. In *Southeastcon, 2013 Proceedings of IEEE*, pages 1–8, April 2013.

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

[76] J. Sijbers, P. Scheunders, M. Verhoye, A. V. der Linden, D. van Dyck, and E. Raman. Watershed-based segmentation of 3d mr data for volume quantization. *Magnetic Resonance Imaging*, 15(6):679 – 688, 1997.

[77] Q. Song, M. Chen, J. Bai, M. Sonka, and X. Wu. Surface-region context in optimal multi-object graph-based segmentation: robust delineation of pulmonary tumors. In *Proceedings of the 22nd international conference on Information processing in medical imaging*, IPMI'11, pages 61–72, Berlin, Heidelberg, 2011. Springer-Verlag.

[78] Q. Song, Y. Liu, Y. Liu, P. K. Saha, M. Sonka, and X. Wu. Graph search with appearance and shape information for 3-D prostate and bladder segmentation. In *Proceedings of the 13th international conference on Medical image computing and computer-assisted intervention: Part III*, MICCAI'10, pages 172–180, Berlin, Heidelberg, 2010. Springer-Verlag.

[79] M. Sonka, V. Hlavac, and R. Boyle. Image Processing, Analysis, and Machine Vision Second Edition. *Image (Rochester, N.Y.)*.

[80] S. Umbaugh. *Computer Vision and Image Processing: A Practical Approach Using Cviptools*. Prentice Hall, 1998.

[81] M. G. Urbanchek, E. B. Picken, L. K. Kalliainen, and W. M. Kuzon. Specific force deficit in skeletal muscles of old rats is partially explained by the existence of denervated muscle fibers. *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, 56(5):B191–B197, 2001.

[82] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(6):583 –598, jun 1991.

[83] D. Waanders, D. Janssen, K. Bertoldi, K. A. Mann, and N. Verdonschot. Mixed mode loading of the cement bone interface a finite element study. *Computer Methods in Biomechanics and Biomedical Engineering*, 14(2):145–155, 2011.

[84] B. Wang and X.-Y. Xu. Minimally invasive reconstruction of lateral ligaments of the ankle using semitendinosus autograft. *Foot and Ankle International*, 34(5):711–715, 2013.

[85] J. Z. Wang, D. A. Turner, and M. D. Chutuape. Fast and interactive algorithm for segmentation of a series of related images: application to volumetric analysis of mr images of the heart. *J Magn Reson Imaging*, 2(5):575–82, 1992.

[86] L. Wang, M. Greenspan, and R. Ellis. Validation of bone segmentation and improved 3-D registration using contour coherency in CT data. *Medical Imaging, IEEE Transactions on*, 25(3):324 –334, march 2006.

[87] S. Wang and J. Siskind. Image segmentation with ratio cut. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(6):675 – 690, june 2003.

[88] S. Wegner, T. Harms, H. Oswald, and E. Fleck. The watershed transformation on graphs for the segmentation of CT images. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 498 –502 vol.3, aug 1996.

[89] Wikipedia. Bone — wikipedia, the free encyclopedia, 2013. [Online; accessed 10-March-2013].

[90] A. Willis, D. Anderson, T. Thomas, T. Brown, and J. L. Marsh. 3d reconstruction of highly fragmented bone fractures. *Med Imaging*, 6512:65121P1–65121P10, 2007.

[91] J. Wu, A. Belle, R. H. Hargraves, C. Cockrell, Y. Tang, and K. Najarian. Bone segmentation and 3D visualization of CT images for traumatic pelvic injuries. *International Journal of Imaging Systems and Technology*, 24(1):29–38, 2014.

[92] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(11):1101 –1113, nov 1993.

[93] Y. Xiaohan, J. Yla-Jaaski, O. Huttunen, T. Vehkomaki, O. Sipila, and T. Katila. Image segmentation combining region growing and edge detection. In *Pattern Recognition, 1992. Vol.III. Conference C: Image, Speech and Signal Analysis, Proceedings., 11th IAPR International Conference on*, pages 481 –484, aug-3 sep 1992.

[94] C. Xu and J. L. Prince. Generalized gradient vector flow external forces for active contours. *Signal Processing-An International Journal*, 71(2):131–139, 1998.

[95] J. Yang, R. Chiou, A. Ruprecht, J. Vicario, L. A. MacPhail, and T. E. Rams. A new device for measuring density of jaw bones. *Dentomaxillofacial Radiology*, 31(5):313–316, 2002.

[96] Y. Yin, X. Zhang, R. Williams, X. Wu, D. Anderson, and M. Sonka. Logismos-layered optimal graph image segmentation of multiple objects and surfaces: Cartilage segmentation in the knee joint. *Medical Imaging, IEEE Transactions on*, 29(12):2023 –2037, dec. 2010.

[97] H. Yousefi, M. Fatehi, M. Amian, and R. Zoroofi. A fully automated segmentation of radius bone based on active contour in wrist MRI data set. In *Biomedical Engineering (ICBME), 2013 20th Iranian Conference on*, pages 42–47, Dec 2013.

[98] Z. Yue, A. Goshtasby, and L. Ackerman. Automatic detection of rib borders in chest radiographs. *Medical Imaging, IEEE Transactions on*, 14(3):525–536, 1995.

[99] J. Zhang, C. Yan, C. Chui, and S. Ong. Fast segmentation of bone in CT images using 3D adaptive thresholding. *Computers in biology and medicine*, 40(2):231–236, 2010.

[100] K. Zhang, H. Song, and L. Zhang. Active contours driven by local image fitting energy. *Pattern Recogn.*, 43(4):1199–1206, Apr. 2010.

[101] B. Zhou. *Geometric analysis tools for mesh segmentation.* PhD thesis, The University of North Carolina At Charlotte, 2013.