Medial Techniques for Automating Finite Element Analysis

by

Jessica Renee Crawford Crouch

A Dissertation submitted to the faculty of The University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill

2003

Approved by:

Stephen M. Pizer, Advisor

Edward Chaney, Reader

Guido Gerig, Reader

Sarang Joshi, Reader

Carol Lucas, Reader

Julian Rosenman, Reader

JESSICA RENEE CRAWFORD CROUCH. Medial Techniques for Automating Finite Element Analysis. (Under the direction of Stephen M. Pizer.)

ABSTRACT

Finite element analysis provides a principled method for modeling physical deformation and is an ideal tool to apply when a medical imaging problem requires the simulation of tissue deformation. The drawback to using finite element analysis for imaging problems is the significant amount of labor and computational time that is often required to construct a finite element model and solve the resulting large system of equations. In particular, challenging questions include

1) how to generate a mesh from an image,

2) how to derive boundary conditions from images of deformed objects, and

3) how to solve the system of finite element equations as efficiently as possible.

These problems make the finite element method prohibitively expensive for some applications and thus make it unlikely to be used in a clinical setting.

The use of medial models called m-reps opens the door to techniques that provide solutions for these challenges.

1) A new meshing algorithm is presented that automatically generates a hexahedral mesh from an m-rep segmentation of an image.

2) M-rep segmentations of an object in different stages of deformation can be used to establish an approximate correspondence between points on the surfaces of the different object configurations. This is sufficient to initialize finite element displacement boundary conditions that enable the computation of a mapping between two shape configurations.

3) An m-rep based mesh subdivision algorithm is presented which together with a multi-grid solution algorithm to improves solution speed for the finite element system of equations.

Taken together, these m-rep based algorithms allow the finite element method to be applied to non-rigid registration problems in an automatic and efficient way.

The methods have been tested using CT images of a phantom pelvis. Deformation accuracy estimates were obtained by calculating the difference between (a) the observed locations of a constellation of brachytherapy seeds in an image of a deformed prostate and (b) the predicted seed locations that result from applying the finite element deformation to an image of the undeformed prostate. The average error was of the same order as the seed segmentation error that was due to image resolution.

Acknowledgments

The teaching, encouragement, and support of many people have enabled me to complete this dissertation and the Ph.D. course of study. First, I would like to thank Steve Pizer for the energy and dedication he brings to both research and teaching. I have learned a great deal from his instruction and his example. I would also like to thank the members of my committee – Steve Pizer, Ed Chaney, Sarang Joshi, Guido Gerig, Carol Lucas, and Julian Rosenman – for lending their expertise and experience and for investing time in my education and research. Additionally, I would like to thank Gig Mageras and Marco Zaider at Memorial Sloan-Kettering Cancer Center who have been valuable collaborators in this research.

I am grateful for the rich collaborative environment provided by the members of the Medical Image Display and Analysis Group (MIDAG) at the University of North Carolina at Chapel Hill. The MIDAG graduate students, faculty, and staff have assisted me in many ways and on many occasions. In particular Tim Quigg, Gregg Tracton, and Graham Gash have provided valuable administrative and technical support for this work.

I have been financially supported through two programs. The Lucent Foundation's Graduate Research Program for Women (GRPW) funded the first four years of my graduate education and provided me with a valuable learning experience at Bell Labs and a mentor, Brian Kernighan. NIH grant CA P01 47982 funded the final two years of this work.

I am grateful for the instruction and guidance I received as an undergraduate from the computer science faculty at the University of Richmond that prepared me for graduate school. In particular Gary Greenfield, my honors thesis adviser at the University of Richmond, first helped me delve into research and encouraged my pursuit of a graduate degree.

I would also like to express appreciation for the encouragement and support provided by my family and friends from outside the UNC community. Shannon Pollard, my roommate of three years and good friend, has provided me with a sympathetic ear and has been an invaluable sounding board and source of encouragement since we first met on candidate's day. She and her husband Robert have graciously hosted me on a number of occasions. My entire family and most especially my parents, Jackie and Frances Crawford, have supported me, believed in me, and helped me find the perseverance to finish this work. My husband, Eric, has been unfailingly patient throughout this process. His confidence in me and his encouragement have sustained me for the past six years. Finally, I am grateful to God for the opportunities I have had to work, learn, and interact with everyone through this process.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivating Application: Brachytherapy

For cancer patients, the success of radiation treatment is dependent first of all on the development of a good treatment plan. A desirable plan provides a high dose of radiation to tumor regions and minimizes the amount of radiation delivered to normal tissue. Secondly, success is dependent on the accuracy of treatment delivery. Accuracy is determined by how closely the treatment actually delivered corresponds to the plan.

Image registration can be a factor in both of these aspects of radiation treatment. At the planning stage, images of different modalities may need to be registered in order to combine the best knowledge about the locations of tumor(s) and normal anatomy structures. In some cases pre-operative and post-operative images need to be registered so that radiation treatment can be planned for a region of tissue that surrounded a tumor before it was surgically removed.

At the treatment delivery stage, image registration is being used experimentally to help adapt treatment plans to a patient's condition at the time of treatment. The tomotherapy model for delivering external beam radiation can involve adapting a treatment plan to a patient [49]. This work involves adapting a brachytherapy treatment plan given an intra-operative patient image.

Deformations of the body between planning and treatment can be caused by mechanical shifting, stretching, or compression of tissue, which may be due to factors such as gravity, differences in filling of the bladder, or devices inserted to aid imaging. The lack of fast, reliable, and automated registration algorithms has in the past precluded the use of treatment techniques that would require computational image registration at the time of treatment. Therefore, the availability of such algorithms will open the door for new strategies to improve the accuracy of treatment delivery.

Brachytherapy is a form of radiation treatment for prostate cancer in which radioactive seeds are implanted in the prostate to continuously deliver radiation to the surrounding prostate tissue. A group of researchers at Memorial Sloan-Kettering Cancer Center (MSKCC) are investigating a method for improving brachytherapy outcomes by using magnetic resonance spectroscopy images (MRSI) in planning seed placement. Magnetic resonance spectroscopy can measure levels of choline and citrate within the prostate with a resolution of .24 cm^3. An elevated level of these chemicals in a region is believed to indicate the presence of a tumor deposit. Therefore the MRSI modality may allow prostate tumors to be localized better than has previously been possible. The MSKCC researchers intend to use MRSI information about tumor location to plan patterns of seed placement that will deliver especially high doses of radiation to areas of the prostate which are shown to be cancerous [84].

In order to effectively implement these plans, a precise method of placing the seeds must be available. Trans-rectal ultrasound is used to visualize the prostate, needle, and seeds intra-operatively and to guide seed placement. However, the positioning of the MRSI probe used in planning presses the prostate against the pubic bone and flattens it. Because the prostate appears in the planning MRSI shifted and deformed as compared to its appearance in the intra-operative ultrasound image, it would be difficult to use the original planning image to direct seed placement. To be most useful, the prostate in the planning image must be registered with the prostate in the ultrasound image. An accurately registered pair of these images would provide a powerful tool for guiding seed placement and following the pre-operative plan.

1.2 Image Registration with Physical Deformation Modeling

Generally, if image registration is required in a clinical setting, then rigid registration is used. Obviously rigid registration is insufficient for the brachytherapy application since it does not account for the significant deformation of the prostate. Active research in the image analysis community has led to the development of several categories of non-rigid registration algorithms. See [38], [39], or [15] for a detailed survey of non-rigid registration algorithms.

Many non-rigid registration algorithms do not use information about the material properties or geometry of an imaged object to constrain the computed deformation to the realm of physically possible deformations. Examples include landmark algorithms such as kriging [41] and related spline fitting methods such as thin plate splines [11] and elastic body splines [23], as well as optical flow algorithms such as Demons [73]. Methods like these lack physically based constraints and are most appropriate for registration problems for which the shape differences between images are not caused by mechanical forces. For example, registration of images from different patients or registration of atlas and patient images requires non-mechanical deformations. Also, if the material properties of the imaged objects or other parameters of physical deformation problem are unknown, the use of physically based constraints may be impossible. However when the physical parameters of a deformation can be reasonably approximated and the registration problem involves images of a single patient that evidence deformation caused by mechanical forces, a registration algorithm that explicitly models the physical deformation offers a distinct advantage. Such an algorithm can guarantee that the computed deformation is physically realistic, at least insofar as its model accurately represents the imaged objects. The finite element, finite volume, and finite difference algorithms all provide different methods for modeling deformable objects and computing mechanical deformations. These algorithms are discussed in more detail in chapter 2.

For the brachytherapy application the deformation evident in the images is known to be caused by a compressive force from the imaging probe. Therefore it is natural to choose a registration algorithm that models the mechanical deformation of the prostate and surrounding anatomy. The finite element method is well suited for this problem, but poses the following challenges:

1. A mesh of the problem domain is required. Mesh creation from an image requires image segmentation and element creation in each segmented region. These are very time-consuming tasks to perform manually, and automatic segmentation and mesh creation are areas of open research.

2. The finite element method is computationally intensive. It involves the formulation of a typically large system of linear equations. Formulating the system of equations and solving the system can require so much computational time that the process becomes impractical for applications with a time constraint.

3. The finite element method requires boundary conditions that define forces and/or displacements applied to portions of the mesh. Defining these man-

ually is a time-consuming and error prone task.

This dissertation describes methods, based on a geometric representation called m-reps, for solving all three problems.

1.3 Overview of Approach

This work addresses the challenges of applying the finite element method to the problem of generating a physically based non-rigid registration of a 3D source and target image pair. The basic steps of the approach are as follows.

1. M-rep models are fit to the objects of interest in the source image. In images with clear boundaries m-reps have been shown to perform well in automatic segmentations [55]. For images that lack clear object boundaries, manually drawn object contours can be used to produce binary images for which the m-rep segmentation algorithm performs well.

2. The source m-rep models produced in step 1 are fit to the objects of interest in the target image. This results in target m-rep models that share the same topology and object coordinate space as the source models but represent the differing geometry exhibited by the deformed objects.

3. A multi-scale mesh is automatically constructed from the source m-rep models using the algorithm described in chapter 4.

4. The shared object coordinate space of the source and target m-rep models implicitly defines a correspondence between source and target object surface points. These surface correspondences are taken as the initial displacement boundary conditions needed to compute a finite element based deformation.

5. Optionally, the initial boundary conditions may be refined. The refinement process consists of solving the finite element equations on the coarsest mesh and iteratively adjusting the surface correspondences between the source and target m-reps so that the energy of the computed deformation is minimized.

6. Given the boundary displacements, an efficient multiscale solution algorithm is used to solve the system of finite element equations on successively finer meshes until the desired precision is reached.

7. The finite element solution can be interpolated through the mesh to define the displacement at every point in the mesh. By interpolating the displacements at voxel locations in the source image, a deformed source image can be computed that is registered with the target image.

1.4 Thesis and Claims

M-rep based multiscale mesh generation, m-rep derived boundary conditions, and multiscale solution of a finite element system of equations are techniques that improve the automation and efficiency of finite element analysis as it is applied to medical imaging applications and to the prostate brachytherapy application in particular.

The claims of this dissertation are that

1. A novel method of automatically generating quality hexahedral meshes from 3D images has been developed. This m-rep based method is designed to mesh anatomical objects found in medical images and is capable of meshing objects that can be modeled with a tree of m-rep protrusion figures.

2. A mesh subdivision algorithm based on m-rep object coordinates has been developed. A subdivided mesh produced by this algorithm represents object shape more smoothly and accurately than the original mesh, thereby reducing the error in the finite element solution that is due to the discrete spatial representation of a continuous, curved geometric object. This is a benefit not provided by straightforward Euclidean mesh subdivision.

3. The m-rep based correspondence between source and target objects provides a good initial approximation for both the finite element boundary conditions and the solution. An approximation to the boundary conditions is necessary for the application of the finite element method. A good solution approximation can reduce the number of iterations required to solve the finite element system of equations.

4. The method presented in this dissertation produces good results for the problem of non-rigidly registering prostate images.

1.5 Applicability to Non-Brachytherapy Problems

Although the brachytherapy application provides the motivation for the work presented here, it is just one example of a medical application that could be impacted by the availability of a fast, physically based elastic registration algorithm.

Physically based non-rigid registration algorithms have been successfully applied to images of the prostate [9] [47], brain [26], and breast [4]. In addition to radiotherapy treatment planning, non-rigid registration can be useful in image guided surgery by enabling a pre-operative planning image to be adapted to an intra-operative situation.

Motion tracking is another medical imaging application that could benefit from non-rigid registration. Some cardiac studies require tracking heart wall motion through contractions. Segmenting the heart wall or other structure of interest from a series of images and then registering images from successive time steps produces the desired estimation of the motion [52].

Looking beyond image registration, much of the methodology described here could be readily applied to medical simulation problems. Finite element analysis is a natural tool to use in simulation tissue deformations, and published results include maxillofacial surgery simulation [17], liver surgery simulation [20], and childbirth simulation [37].

The hexahedral meshing part of this work has the potential to be applied even more broadly to non-image based and non-medical problems. The meshing algorithm could be applied to any finite element problem if a method was available for constructing problem-specific m-rep object models. Although current m-rep tools focus on producing models of imaged objects, it is certainly possible that the m-rep infrastructure could be enhanced to support non-image based modeling. For example, support could be added for conversion of boundary based models into m-rep format, allowing the m-rep meshing algorithm to be applied to CAD generated models [72].

1.6 Overview of Document

Chapter 2 presents background on mechanical modeling techniques and finite element analysis in particular. Chapter 3 provides a survey of existing meshing algorithms, while chapter 4 describes a new m-rep based meshing algorithm for collections of simple and complex objects. The assignment of boundary conditions is explained in chapter 5. The multi-grid solution algorithm is presented in chapter 6, and experimental results are given in chapter 7. Finally, a discussion of conclusions and future work can be found in chapter 8.

Chapter 2

Deformation Modeling Background

2.1 Introduction

Deformable models are used to represent real-world objects and predict those objects' responses to applied forces. To provide this predictive capability, a deformable model must incorporate

1. a description of an object's geometry and
2. information about an object's material properties.

Deformable models generally satisfy the first requirement by representing object geometry with a grid or mesh, a topic that is discussed in chapters 3 and 4. This chapter focuses on the second requirement, which is the mathematical modeling of a material's deformable characteristics. The most commonly used mathematical models are based on partial differential equations (PDEs) that establish the relationship between stress and strain for a given material. A review of these PDEs is presented here and begins with the definitions of stress and strain. The reader is directed to [77] or [51] for more thorough coverage of this topic.

2.2 Mathematical Definitions

2.2.1 Strain

Strain is a measure of deformation. If X is defined as $X = \begin{bmatrix} X_x & X_y & X_z \end{bmatrix}$ where $\begin{bmatrix} X_x & X_y & X_z \end{bmatrix}$ is a point in a deformable object, and $u(X)$ is a function that represents the displacement at point X, a change of distance between points in the neighborhood of X can be characterized by the following normal strains:

$$\varepsilon_{xx} = \frac{\partial u_x}{\partial x} \qquad \varepsilon_{yy} = \frac{\partial u_y}{\partial y} \qquad \varepsilon_{zz} = \frac{\partial u_z}{\partial z} \tag{2.1}$$

A deformation that results in a change of angle between intersecting lines in the neighborhood of X is characterized by the following shear strains.

$$\varepsilon_{xy} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \qquad \varepsilon_{xz} = \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \qquad \varepsilon_{yz} = \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y} \tag{2.2}$$

The linear strain tensor measures local deformation and is defined as

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & \varepsilon_{zz} \end{bmatrix} \tag{2.3}$$

2.2.2 Stress

Stress is a measure of the forces in a deformable object. Two types of forces can cause deformation: body forces and surface forces. Body forces are applied to an object's volume. Examples include gravity and inertial forces. No body forces are considered in this work, as surface forces are assumed to be the primary cause of the observed anatomic object deformation. This assumption is reasonable for the prostate deformation problem but might be revised for other specific problems. The stress caused by surface forces is measured in terms of force per unit area. If P is a force vector applied to a patch of surface A, the traction vector $\overrightarrow{t}$ can be defined as the limiting value of the force to area ratio as the area shrinks to 0.

$$\overrightarrow{t} = \frac{\delta P}{\delta A} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Traction vectors for patches perpendicular to the x, y, and z coordinate axes can be labelled as follows:

$$\text{If the patch surface normal } \overrightarrow{n} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad \text{then} \qquad \overrightarrow{t} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{xy} \\ \sigma_{xz} \end{bmatrix} \tag{2.4}$$

$$\text{If the patch surface normal } \overrightarrow{n} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad \text{then} \qquad \overrightarrow{t} = \begin{bmatrix} \sigma_{xy} \\ \sigma_{yy} \\ \sigma_{yz} \end{bmatrix} \tag{2.5}$$

$$\text{If the patch surface normal } \overrightarrow{n} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \text{then} \qquad \overrightarrow{t} = \begin{bmatrix} \sigma_{xz} \\ \sigma_{yz} \\ \sigma_{zz} \end{bmatrix} \tag{2.6}$$

The components of these traction vectors can be used to define the symmetric stress tensor, which succinctly represents the stress in the neighborhood of a point.

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{bmatrix} \tag{2.7}$$

The diagonal elements are normal stresses, and the off-diagonal elements are shear stresses. The stress tensor can be used to determine the traction vector for a surface patch of any orientation because $\overrightarrow{t} = \sigma \overrightarrow{n}$, where $\overrightarrow{n}$ is the surface patch normal.

2.3 Deformable Materials

The equations that describe a material's deformable behavior are known as constitutive equations. Fluid constitutive equations exist for viscous and non-viscous fluids, as well as compressible and incompressible fluids. Solid tissues are often represented using linearly elastic or viscoelastic constitutive equations. A search of the biomechanics literature reveals a number of more complicated and specialized models that have been developed for particular tissue types. A few of the more common constitutive models are reviewed below. A more detailed treatment of tissue biomechanics can be found in Fung's book on tissue biomechanics [29] and in Hirota's dissertation [?].

2.3.1 Linear Elastic Solid

Linear elasticity is the most idealized solid model. It assumes that deformations occur instantly and neglects the effects that strain rate and temperature have on the deformation process. It also assumes that an object immediately springs back to its original form when a deforming stress is removed. For linearly elastic materials, the stress tensor has a linear relationship with the strain tensor. The constitutive equation is

$$\sigma_{ij} = c_{ijkl}\varepsilon_{kl} \tag{2.8}$$

where the c_{ijkl} are constant coefficients that can be determined based on the elastic properties of a material.

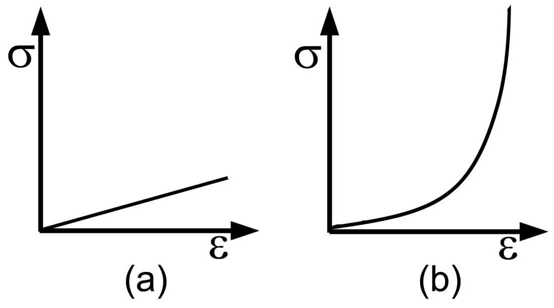


Figure 2.1: Stress/Strain graphs for (a) linear elastic solid, (b) hyperelastic solid

2.3.2 Hyperelastic Solid

The hyperelastic model retains the linear elastic assumption that the deformation process is perfectly reversible, but it removes the requirement of a linear relationship between stress and strain. With the hyperelastic model a material's stiffness is not a constant but rather a function of an object's current state of deformation. The constitutive equation for a hyperelastic model takes the general form

$$\sigma = f(\varepsilon) \tag{2.9}$$

where $f(\varepsilon)$ is a non-linear function. Fig. 2.1 shows how the stress/strain curve of a hyperelastic material could compare to that of a linear elastic material.

2.3.3 Viscoelastic Solid

Viscoelasticity introduces the concept of a rate dependent deformation process. It incorporates the complex behaviors of relaxation, creep, and hysteresis. Relaxation occurs when the stress in an object slowly diminishes as its strain is kept constant. Creep occurs when an object continues to slowly deform as constant stress is applied.

Hysteresis is involved if the deformation process that occurs when stress is applied to an object does not happen exactly in reverse when the stress is removed. These three effects are all time-dependent, and they replace the assumption of linear elasticity that deformations are instantaneous. Many soft tissues can be well represented with a viscoelastic model, and many different viscoelastic models exist.

Three of the simplest and most well-known of the viscoelastic models are the Maxwell, Voigt, and Kelvin models [13] [29]. The constitutive equations for these models are as follows:

$$\text{Maxwell:} \qquad G\sigma + \eta\frac{\partial\sigma}{\partial t} = G\eta\frac{\partial\varepsilon}{\partial t} \tag{2.10}$$

$$\text{Voigt:} \qquad \sigma = G\varepsilon + \eta\frac{\partial\varepsilon}{\partial t} \tag{2.11}$$

$$\text{Kelvin:} \qquad G_1\sigma + \eta\frac{\partial\sigma}{\partial t} = G_1G_2\varepsilon + (G_1 + G_2)\eta\frac{\partial\varepsilon}{\partial t} \tag{2.12}$$

The scalar constants G, G_1, and G_2 are dependent on the stiffness of the material, the scalar constant η is a coefficient of viscosity, and t represents time. The Kelvin model is the most general purpose of the three models, as it includes terms for the stress, strain, and the first derivatives of each with respect to time. The shape of the creep and relaxation functions of each of these models are shown in Fig. 2.2 and Fig. 2.3, respectively The exact curvature shown in these graphs will vary depending on the G, G_1, G_2, and η constants.

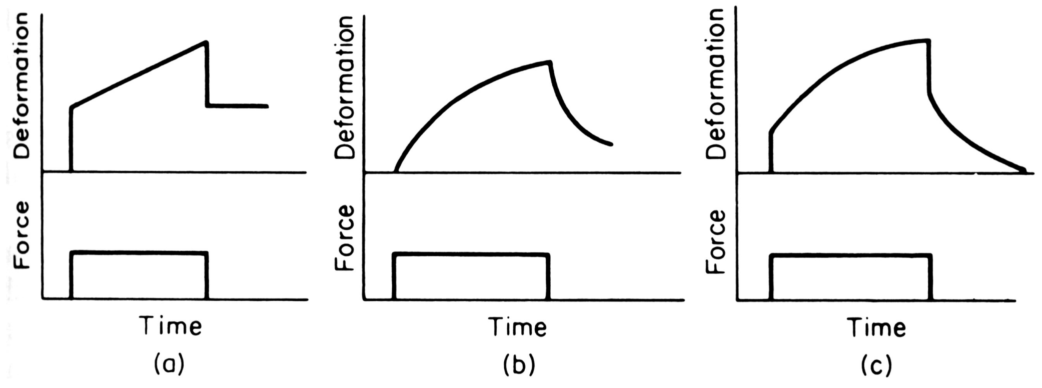


Figure 2.2: Creep function shapes for (a) Maxwell model (b) Voigt model (c) Kelvin model. Figure copied from [29].

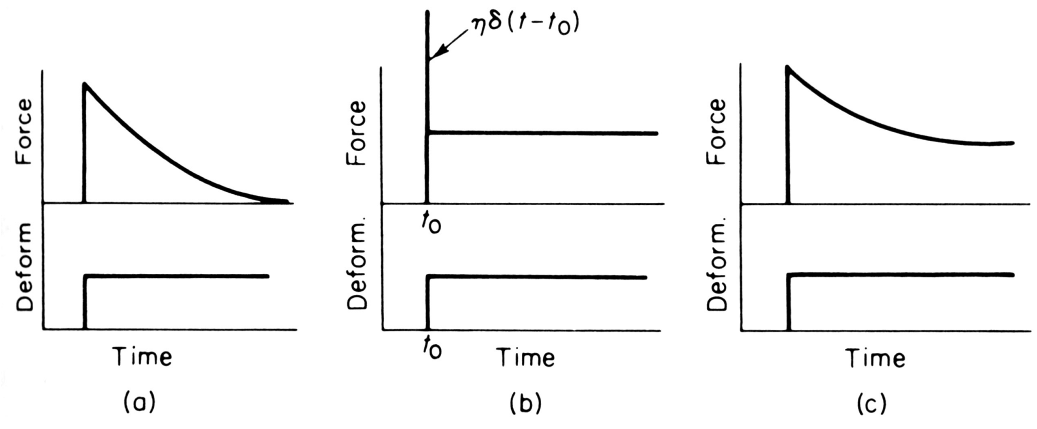


Figure 2.3: Relaxation function shapes for (a) Maxwell model (b) Voigt model (c) Kelvin model. Figure copied from [29].

2.4 Prostate Model

The prostate could probably be more accurately represented with a viscoelasatic model than with a linearly elastic model, but at this time empirical data is not available to determine what the parameters of such a viscoelastic model should be. While extensive knowledge is available about the elastic behavior of steel, plastics, wood, and other materials used in manufacturing, and the deformability of some soft biological materials including lung [40], muscle [54], and brain tissue [45] have been the subject of investigation, less information is available about the elastic properties of other soft tissue structures such as the prostate.

At least one study has indicated that the prostate can be reasonably modeled as a linearly elastic body [35]. In addition, other work that involved modeling the prostate also relied on a linear elastic prostate model, with apparent success [9]. In the absence of a more specialized constitutive model based on experiments with prostate tissue samples, isotropic linear elasticity is a reasonable model to apply.

Each element in a finite element mesh is assigned elastic constants appropriate for the material the mesh element represents. In the case of an isotropic linearly elastic material, the only two constants needed are Young's modulus and Poisson's ratio. For the prostate model in this work, the constants were chosen based on the results reported in [35]. The chosen value for Young's modulus is 60 kPa, and the chosen value for Poisson's ratio is .495.

2.5 Implementation of Constitutive Models

Although the linear elastic model is selected for the experiments detailed in this work, most of the techniques presented could be applied equally well to a finite element problem that employed a different constitutive model. For example, a simulation based on a rate dependent viscoelastic model is implemented by discretizing both space and time. The spatial discretization is achieved with a mesh, just as for a linear elastic problem. The temporal discretization is achieved by dividing the problem's time domain into slices and treating each time slice as a non-rate dependent problem. From an implementation point of view, this turns the task of solving a rate dependent problem into the task of solving a series of non-rate dependent problems.

Therefore when modeling a rate dependent, non-linear problem, the issues of mesh generation, boundary condition specification, and solution approach that are discussed in the following chapters must be addressed. Perhaps the most significant difference is that efficiency is an even greater concern with these types of problems because the time slicing approach increases the computational load. Consequently, a technique that offers a small savings in solution time for a linear elastic problem may provide an even greater savings for rate dependent problems.

2.6 Equations of Linear Elasticity

A succinct description of the mathematical basis for the finite element method is provided here. For a more thorough treatment of the mathematical detail and derivations the reader is referred to the references, especially [77]. See also [36], [51], [67], and [25].

2.6.1 Generalized Hooke's Law

The constitutive model for linear elasticity is given by the generalized Hooke's Law, which essentially states that the strain tensor is linearly related to the stress tensor. The equation (as previously shown in equation 2.8) is

$$\sigma_{ij} = c_{ijkl}\varepsilon_{kl}$$

where the c_{ijkl} are constant coefficients that can be determined based on the elastic properties of a material.

In the most general case there are 81 such coefficients, but for the special case of an isotropic material there are only two coefficients. The constitutive equation for isotropic linearly elastic materials is

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + \delta_{ij}\lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) \tag{2.13}$$

where δ_{ij} is Kronecker's delta function and μ and λ are the two coefficients.

To allow equation 2.13 to be written in matrix form, vectors containing the unique stress and strain coefficients are defined:

$$\underline{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{bmatrix} \qquad \underline{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{bmatrix} \tag{2.14}$$

Now by defining a coefficient matrix D, equation 2.13 can be written as

$$\underline{\sigma} = D\underline{\varepsilon} \tag{2.15}$$

$$\text{where} \quad D = \begin{bmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu \end{bmatrix} \tag{2.16}$$

μ and λ are known as Lamé's coefficients and can be determined from Young's modulus and Poisson's ratio. Young's modulus is a material's stress/strain ratio for a uni-axially applied force. Poisson's ratio is the amount of a material's contraction in the direction perpendicular to a uni-axially applied force divided by the amount of elongation in the direction of the force. The relationships between Young's modulus,

E, Poisson's ratio, ν and Lamés coefficients are as follows:

$$\begin{aligned} E &= \frac{\mu\left(3\lambda+2\mu\right)}{\lambda+\mu} && (2.17)\\ \nu &= \frac{\lambda}{2\left(\lambda+\mu\right)} && (2.18)\\ \lambda &= \frac{E\nu}{\left(1-2\nu\right)\left(1+\nu\right)} && (2.19)\\ \mu &= \frac{E}{2\nu+2} && (2.20) \end{aligned}$$

The coefficient matrix D from equation 2.16 can be restated in terms of Young's modulus and Poisson's ratio.

$$D = \frac{E}{\left(1-2\nu\right)\left(1+\nu\right)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}-\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}-\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}-\nu \end{bmatrix} \quad (2.21)$$

2.6.2 Equilibrium

In order for a body to be stationary, force equilibrium conditions must be met. These conditions are stated in the form of partial differential equations. The equations take into account both the stress tensor and any body force. A body force, $\overrightarrow{b}$, applies to volume elements as opposed to surface elements. Gravity is an example of a body force.

$$\overrightarrow{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \quad (2.22)$$

The force equilibrium equations are

$$\begin{aligned} \frac{\partial\sigma_{xx}}{\partial x}+\frac{\partial\sigma_{xy}}{\partial y}+\frac{\partial\sigma_{xz}}{\partial z}+b_x &= 0\\ \frac{\partial\sigma_{xy}}{\partial x}+\frac{\partial\sigma_{yy}}{\partial y}+\frac{\partial\sigma_{yz}}{\partial z}+b_y &= 0 \qquad (2.23)\\ \frac{\partial\sigma_{xz}}{\partial x}+\frac{\partial\sigma_{yz}}{\partial y}+\frac{\partial\sigma_{zz}}{\partial z}+b_y &= 0 \end{aligned}$$

2.6.3 Linear elastic PDE – Strong form

Substituting the constitutive equation 2.13 into the force equilibrium equations 2.23 yields the linear elastic partial differential equations:

$$\begin{aligned}
\frac{\partial}{\partial x}\left(2\mu\varepsilon_{xx}+\lambda\left(\varepsilon_{xx}+\varepsilon_{yy}+\varepsilon_{zz}\right)\right)+2\mu\frac{\partial\varepsilon_{xy}}{\partial y}+2\mu\frac{\partial\varepsilon_{xz}}{\partial z}+b_x &= 0\\
2\mu\frac{\partial\varepsilon_{xy}}{\partial x}+\frac{\partial}{\partial y}\left(2\mu\varepsilon_{yy}+\lambda\left(\varepsilon_{xx}+\varepsilon_{yy}+\varepsilon_{zz}\right)\right)+2\mu\frac{\partial\varepsilon_{xz}}{\partial z}+b_y &= 0\\
2\mu\frac{\partial\varepsilon_{xy}}{\partial x}+2\mu\frac{\partial\varepsilon_{xy}}{\partial y}+\frac{\partial}{\partial z}\left(2\mu\varepsilon_{zz}+\lambda\left(\varepsilon_{xx}+\varepsilon_{yy}+\varepsilon_{zz}\right)\right)+b_z &= 0
\end{aligned}\tag{2.24}$$

Substituting the definition of the strain terms (equations 2.1 and 2.2) produces the following equations:

$$\begin{aligned}
\frac{\partial}{\partial x}\left(2\mu\frac{\partial u_x}{\partial x}+\lambda\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)\right)+2\mu\frac{\partial\varepsilon}{\partial y}\left(\frac{1}{2}\left(\frac{\partial\mu_x}{\partial y}+\frac{\partial\mu_y}{\partial x}\right)\right)&\\
+2\mu\frac{\partial\varepsilon}{\partial z}\left(\frac{1}{2}\left(\frac{\partial\mu_x}{\partial z}+\frac{\partial\mu_z}{\partial x}\right)\right)+b_x &= 0\\
2\mu\frac{\partial\varepsilon}{\partial x}\left(\frac{1}{2}\left(\frac{\partial\mu_x}{\partial y}+\frac{\partial\mu_y}{\partial x}\right)\right)+\frac{\partial}{\partial y}\left(2\mu\varepsilon_{yy}+\lambda\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)\right)&\\
+2\mu\frac{\partial\varepsilon}{\partial z}\left(\frac{1}{2}\left(\frac{\partial\mu_y}{\partial z}+\frac{\partial\mu_z}{\partial y}\right)\right)+b_y &= 0\\
2\mu\frac{\partial\varepsilon}{\partial x}\left(\frac{1}{2}\left(\frac{\partial\mu_x}{\partial z}+\frac{\partial\mu_z}{\partial x}\right)\right)+2\mu\frac{\partial\varepsilon}{\partial y}\left(\frac{1}{2}\left(\frac{\partial\mu_y}{\partial z}+\frac{\partial\mu_z}{\partial y}\right)\right)&\\
+\frac{\partial}{\partial z}\left(2\mu\varepsilon_{zz}+\lambda\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)\right)+b_z &= 0
\end{aligned}\tag{2.25}$$

Collecting terms gives

$$\begin{aligned}
\mu\left(\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_x}{\partial y^2}+\frac{\partial^2 u_x}{\partial z^2}\right)+(\mu+\lambda)\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+b_x=0\\
\mu\left(\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_x}{\partial y^2}+\frac{\partial^2 u_x}{\partial z^2}\right)+(\mu+\lambda)\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+b_x=0\\
\mu\left(\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_x}{\partial y^2}+\frac{\partial^2 u_x}{\partial z^2}\right)+(\mu+\lambda)\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+b_x=0
\end{aligned}\tag{2.26}$$

These equations are known as Navier's equations and can be written more concisely as

$$\mu\nabla^2\overrightarrow{u}+(\mu+\lambda)\nabla\left(\nabla^T\overrightarrow{u}\right)+\overrightarrow{b}=0\tag{2.27}$$

2.7 Algorithms for Discretizing and Solving PDEs

Most real world deformation problems are complicated enough that it is not practical to solve them analytically. Instead, a PDE of interest is usually solved numerically using one of the finite difference, finite volume, or finite element methods. These numerical approaches seek to approximate the solution to a PDE at a finite number of points, called nodes, and interpolate the solution between those nodes. Each of these methods requires a discrete representation of the problem's spatial domain that includes a set of node points, but they vary both in their approaches to spatial discretization and in their formulations of the discretized equations.

2.7.1 Finite Difference

The finite difference method fills the region of interest with a regular Cartesian grid, and the points where grid lines intersect form the nodes. Of the three methods, finite difference has the most straightforward implementation. Interpolation of the solution inside the grid cells provides a continuous solution approximation across the region of interest, and numerical approximation of derivatives on a regular grid can be easily computed using a Taylor expansion around node points. Efficient solution algorithms that take advantage of the regular grid structure can be employed with the finite difference method.

The main disadvantage of the finite difference method is that unless the object of interest happens to be rectangular, the finite difference grid does not conform to the object's geometry. Therefore with the finite difference method the discrete representation of a curved object either has a blocky, jagged boundary or is padded to fit a rectangular grid. Either way, the representation contains geometric error that contributes to solution error. This limitation is addressed by the more flexible meshes of the finite element and finite volume methods.

2.7.2 Finite Element

The finite element method can represent objects with arbitrary geometry by using a mesh with isoparametric elements that conform to object boundaries. Meshes are discussed in greater detail in the following chapter; here it suffices to make three points about finite element meshes.

1. A mesh can be composed of different types of elements. In 2D, triangles and

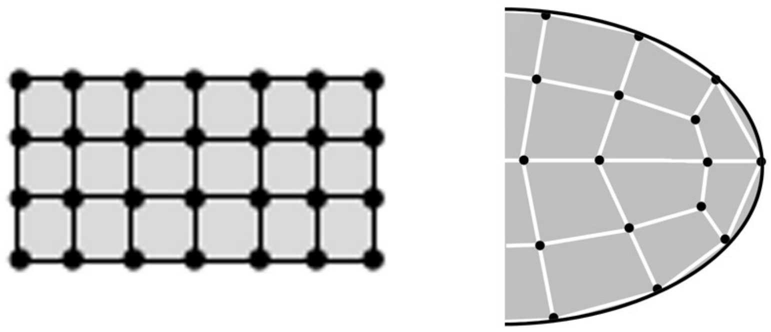


Figure 2.4: Left: Cartesian grid, like those used with the finite difference method. Right: Boundary fitted grid, like those used with finite elements.

quadrilaterals are common element types. In 3D, tetrahedra and hexahedra are common element types.

2. Isoparametric element edges do not have to be parallel with a coordinate axis, the way grid lines are in a finite difference problem. That means that the elements can be aligned with object boundaries and can model arbitrary geometries.

3. An element's interpolation functions can be of arbitrarily high order. Increasing the order of the polynomial interpolation functions is one method of improving the accuracy of the interpolated solution.

This geometric flexibility comes at the cost of greater complexity in the solution algorithm. However, the advantages offered by the more customizable meshing schemes are significant enough that the finite element method is perhaps the most popular choice for simulations involving deformable solids.

2.7.3 Finite Volume

The finite volume method can be used with either finite element type meshes or finite difference type grids. A unique characteristic of the finite volume method is that it makes use of both an initial mesh and the dual of that mesh. The dual is constructed so that the nodes in the initial mesh become element centers in the dual mesh.

The mesh dual is needed because the finite volume method uses an integral form of the relevant partial differential equations. With this method, the value computed for each node is a volume integral, where the volume is the corresponding element in the dual mesh. For the linear elastic equations, the finite volume method computes for each node in the initial mesh the average displacement of the points in the corresponding element in the dual mesh.

The construction of the dual mesh is an additional step not required by the finite element method. Another potential drawback to the finite volume method is that it is difficult to use higher than second order interpolation functions, unlike the finite element method [**?**].

The finite volume method conserves fluxes across element boundaries and has the advantage of greater stability for some problems involving fluid dynamics or large deformations. It is probably most commonly applied to fluid flow problems where this stability advantage can be critical.

2.8 Finite Element Formulation of Linear Elasticity

For this work, the finite element method is chosen for discretizing and solving the linear elastic PDE because it offers the advantage of flexible mesh geometry without the added complication of computing a dual mesh.

2.8.1 Linear elastic PDE – Weak form

In order to apply the finite element method, the weak form of the linear elastic PDE is needed. The general strategy for deriving the weak form of an equation is to multiply the equation by a weighting function and then integrate. These steps provide a more general formulation of the PDE that involves only first order derivatives, unlike the strong form which includes second order derivatives.

Define $v(\overrightarrow{x})$, an arbitrary weighting function defined over a problem's spatial domain:

$$v(\overrightarrow{x}) = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{2.28}$$

Multiply the force equilibrium equations 2.23 by the weighting function $v(\overrightarrow{x})$ and integrate over the volume V.

$$\begin{aligned}
&\int_V v_x \frac{\partial \sigma_{xx}}{\partial x}\,\mathrm{d}V + \int_V v_x \frac{\partial \sigma_{xy}}{\partial y}\,\mathrm{d}V + \int_V v_x \frac{\partial \sigma_{xz}}{\partial z}\,\mathrm{d}V + \int_V v_x b_x\,\mathrm{d}V = 0 \\
&\int_V v_y \frac{\partial \sigma_{xy}}{\partial x}\,\mathrm{d}V + \int_V v_y \frac{\partial \sigma_{yy}}{\partial y}\,\mathrm{d}V + \int_V v_y \frac{\partial \sigma_{yz}}{\partial z}\,\mathrm{d}V + \int_V v_y b_y\,\mathrm{d}V = 0 \\
&\int_V v_z \frac{\partial \sigma_{xz}}{\partial x}\,\mathrm{d}V + \int_V v_z \frac{\partial \sigma_{yz}}{\partial y}\,\mathrm{d}V + \int_V v_z \frac{\partial \sigma_{zz}}{\partial z}\,\mathrm{d}V + \int_V v_z b_z\,\mathrm{d}V = 0
\end{aligned} \tag{2.29}$$

The Green-Gauss theorem states

$$\int_V \phi \operatorname{div} q\,\mathrm{d}V = \int_S \phi q^T n\,\mathrm{d}S - \int_V (\nabla\phi)^T q\,\mathrm{d}V \tag{2.30}$$

where ϕ and q are functions defined over the volume V, S is the surface of V, and n is the outward facing unit-length surface normal. The Green-Gauss theorem is restated below for three particular choices of the function q.

$$\text{If } \overrightarrow{q} = \begin{bmatrix} \psi \\ 0 \\ 0 \end{bmatrix}, \qquad \text{then} \qquad \int_V \phi \frac{\partial \psi}{\partial x}\mathrm{d}V = \int_S \phi\psi n_x \mathrm{d}S - \int_V \frac{\partial \phi}{\partial x}\psi \mathrm{d}V \tag{2.31}$$

$$\text{If } \overrightarrow{q} = \begin{bmatrix} 0 \\ \psi \\ 0 \end{bmatrix}, \qquad \text{then} \qquad \int_V \phi \frac{\partial \psi}{\partial y}\mathrm{d}V = \int_S \phi\psi n_y \mathrm{d}S - \int_V \frac{\partial \phi}{\partial y}\psi \mathrm{d}V \tag{2.32}$$

$$\text{If } \overrightarrow{q} = \begin{bmatrix} 0 \\ 0 \\ \psi \end{bmatrix}, \qquad \text{then} \qquad \int_V \phi \frac{\partial \psi}{\partial z}\mathrm{d}V = \int_S \phi\psi n_z \mathrm{d}S - \int_V \frac{\partial \phi}{\partial z}\psi \mathrm{d}V \tag{2.33}$$

The Green-Gauss theorem as stated in equations 2.31, 2.32, and 2.33 is used to integrate equation 2.29.

$$\begin{aligned}
&\int_S v_x\sigma_{xx}n_x\,\mathrm{d}S - \int_V \frac{\partial v_x}{\partial x}\sigma_{xx}\,\mathrm{d}V + \int_S v_x\sigma_{xy}n_y\,\mathrm{d}S - \int_V \frac{\partial v_x}{\partial y}\sigma_{xy}\,\mathrm{d}V + \\
&\qquad\qquad \int_S v_x\sigma_{xz}n_z\,\mathrm{d}S - \int_V \frac{\partial v_x}{\partial z}\sigma_{xz}\,\mathrm{d}V + \int_V v_x b_x\,\mathrm{d}V = 0 \\
&\int_S v_y\sigma_{xy}n_x\,\mathrm{d}S - \int_V \frac{\partial v_y}{\partial x}\sigma_{xy}\,\mathrm{d}V + \int_S v_y\sigma_{yy}n_y\,\mathrm{d}S - \int_V \frac{\partial v_y}{\partial y}\sigma_{yy}\,\mathrm{d}V + \\
&\qquad\qquad \int_S v_y\sigma_{yz}n_z\,\mathrm{d}S - \int_V \frac{\partial v_y}{\partial z}\sigma_{yz}\,\mathrm{d}V + \int_V v_y b_y\,\mathrm{d}V = 0 \\
&\int_S v_z\sigma_{xz}n_x\,\mathrm{d}S - \int_V \frac{\partial v_z}{\partial x}\sigma_{xz}\,\mathrm{d}V + \int_S v_z\sigma_{yz}n_y\,\mathrm{d}S - \int_V \frac{\partial v_z}{\partial y}\sigma_{yz}\,\mathrm{d}V + \\
&\qquad\qquad \int_S v_z\sigma_{zz}n_z\,\mathrm{d}S - \int_V \frac{\partial v_z}{\partial z}\sigma_{zz}\,\mathrm{d}V + \int_V v_z b_z\,\mathrm{d}V = 0
\end{aligned} \tag{2.34}$$

Substituting the definition of the traction vector (equation 2.4) gives the following equations:

$$\begin{aligned}
\int_V v_x t_x\,\mathrm{d}S - \int_V \left(\frac{\partial v_x}{\partial x}\sigma_{xx} + \frac{\partial v_x}{\partial y}\sigma_{xy} + \frac{\partial v_x}{\partial z}\sigma_{xz}\right)\mathrm{d}V + \int_V v_x b_x\,\mathrm{d}V = 0 \\
\int_V v_y t_y\,\mathrm{d}S - \int_V \left(\frac{\partial v_y}{\partial x}\sigma_{xy} + \frac{\partial v_y}{\partial y}\sigma_{yy} + \frac{\partial v_y}{\partial z}\sigma_{yz}\right)\mathrm{d}V + \int_V v_y b_y\,\mathrm{d}V = 0 \\
\int_V v_z t_z\,\mathrm{d}S - \int_V \left(\frac{\partial v_z}{\partial x}\sigma_{xz} + \frac{\partial v_z}{\partial y}\sigma_{yz} + \frac{\partial v_z}{\partial z}\sigma_{zz}\right)\mathrm{d}V + \int_V v_z b_z\,\mathrm{d}V = 0
\end{aligned} \tag{2.35}$$

Defining the operator $\tilde{\nabla}$ allows the weak form of the equation to be written more concisely.

$$\tilde{\nabla} = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} + \frac{\partial}{\partial x} \\ \frac{\partial}{\partial z} + \frac{\partial}{\partial x} \\ \frac{\partial}{\partial z} + \frac{\partial}{\partial y} \end{bmatrix} \qquad \tilde{\nabla} v(\overrightarrow{x}) = \begin{bmatrix} \frac{\partial v_x}{\partial x} \\ \frac{\partial v_y}{\partial y} \\ \frac{\partial v_z}{\partial z} \\ \frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \\ \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \\ \frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \end{bmatrix} \tag{2.36}$$

$$\int_V \left(\tilde{\nabla} v(\overrightarrow{x})\right)^T \underline{\sigma}\,\mathrm{d}V = \int_S v^T \overrightarrow{t}\,\mathrm{d}S + \int_V v^T \overrightarrow{b}\,\mathrm{d}V \tag{2.37}$$

where $\underline{\sigma}$ is the vector defined by equation 2.14.

2.8.2 Discrete approximation of the solution

The finite element method employs the following solution strategy:

1. Define a mesh across the space for which a solution is desired.
2. Compute an approximate solution for the nodes of the mesh.
3. Interpolate the solution between the nodes to get a continuous solution for the space.

As the size of the mesh elements tends to zero, this approximate solution converges to the exact answer that would be obtained analytically if an analytical solution were feasible.

In this work linear mesh elements are used, and the displacement field is interpolated linearly between nodes. Each node has an associated shape function that controls the interpolation of the solution. A shape function for node X equals 1 at node X, varies linearly between 1 and 0 across the elements that touch node X, and equals 0 at all points outside the elements that touch node X. The displacement solution at any point in an element can be computed as a weighted sum of the displacements at the nodes of the element, where the weights are determined by the shape functions of the nodes.

If $\overrightarrow{x}$ is a position vector and $N_1(\overrightarrow{x}) ... N_n(\overrightarrow{x})$ are the shape functions for a three-dimensional mesh with n nodes, the shape function matrix can be defined as follows:

$$N(\overrightarrow{x}) = \begin{bmatrix} N_1(\overrightarrow{x}) & 0 & 0 & N_2(\overrightarrow{x}) & 0 & 0 & ... & N_n(\overrightarrow{x}) & 0 & 0 \\ 0 & N_1(\overrightarrow{x}) & 0 & 0 & N_2(\overrightarrow{x}) & 0 & ... & 0 & N_n(\overrightarrow{x}) & 0 \\ 0 & 0 & N_1(\overrightarrow{x}) & 0 & 0 & N_2(\overrightarrow{x}) & ... & 0 & 0 & N_n(\overrightarrow{x}) \end{bmatrix} \tag{2.38}$$

If $u(\overrightarrow{x})$ is a function that returns the displacement vector for any position $\overrightarrow{x}$, and $\begin{bmatrix} u_{ix} \\ u_{iy} \\ u_{iz} \end{bmatrix}$ denotes the displacement of the i^{th} node, the vector $\underline{a}$ of node displacements can be defined as follows:

$$\underline{a} = \begin{bmatrix} u_{1x} \\ u_{1y} \\ u_{1z} \\ \vdots \\ u_{nx} \\ u_{ny} \\ u_{nz} \end{bmatrix} \tag{2.39}$$

The continuous displacement function can then be written as the matrix - vector product of $N\left(\overrightarrow{x}\right)$ and $\underline{a}$.

$$u\left(\overrightarrow{x}\right) = N\left(\overrightarrow{x}\right)\underline{a} \tag{2.40}$$

Now define the matrix B as

$$B = \tilde{\nabla} N = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & 0 & \frac{\partial N_2}{\partial x} & 0 & 0 & \dots & \frac{\partial N_n}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & 0 & \frac{\partial N_2}{\partial y} & 0 & \dots & 0 & \frac{\partial N_n}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial z} & 0 & 0 & \frac{\partial N_2}{\partial z} & \dots & 0 & 0 & \frac{\partial N_n}{\partial z} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & 0 & \dots & \frac{\partial N_n}{\partial y} & \frac{\partial N_n}{\partial x} & 0 \\ \frac{\partial N_1}{\partial z} & 0 & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial z} & 0 & \frac{\partial N_2}{\partial x} & \dots & \frac{\partial N_n}{\partial z} & 0 & \frac{\partial N_n}{\partial x} \\ 0 & \frac{\partial N_1}{\partial z} & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial z} & \frac{\partial N_2}{\partial y} & \dots & 0 & \frac{\partial N_n}{\partial z} & \frac{\partial N_n}{\partial y} \end{bmatrix} \tag{2.41}$$

Recalling the definition of strain from equations 2.1 and 2.2, the unique components of the strain tensor can be rewritten using equation 2.40 and equation 2.41.

$$\underline{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \frac{\partial u_z}{\partial z} \\ \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \\ \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \\ \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y} \end{bmatrix} = \tilde{\nabla} u = \tilde{\nabla}\left(N\left(\overrightarrow{x}\right)\underline{a}\right) = B\underline{a} \tag{2.42}$$

Substituting equation 2.42 into the constitutive equation 2.15 gives

$$\underline{\sigma} = D\underline{\varepsilon} = DB\underline{a} \tag{2.43}$$

Recalling that the weight function v from equation 2.28 is arbitrary, here let it be defined as

$$v(\overrightarrow{x}) = NC \tag{2.44}$$

where C is an arbitrary matrix. Applying the operator $\tilde{\nabla}$ to equation 2.44 produces the following equation.

$$\tilde{\nabla} v(\overrightarrow{x}) = \tilde{\nabla}(NC) = BC \tag{2.45}$$

Substituting equation 2.45 into the weak form of the linear elastic PDE (equation 2.37) gives

$$C^T \left(\int_V B^T \underline{\sigma} \,\mathrm{d}V - \int_S N^T \overrightarrow{t} \,\mathrm{d}S - \int_V N^T \overrightarrow{b} \,\mathrm{d}V \right) = 0 \tag{2.46}$$

Since C is arbitrary, it can be dropped from equation 2.46.

$$\int_V B^T \underline{\sigma} \,\mathrm{d}V - \int_S N^T \overrightarrow{t} \,\mathrm{d}S - \int_V N^T \overrightarrow{b} \,\mathrm{d}V = 0 \tag{2.47}$$

Substituting equation 2.43 into equation 2.47 gives

$$\left(\int_V B^T D B \,\mathrm{d}V \right) \underline{a} = \int_S N^T \overrightarrow{t} \,\mathrm{d}S + \int_V N^T \overrightarrow{b} \,\mathrm{d}V \tag{2.48}$$

Rewriting the equation with matrix notation gives

$$K\underline{a} = \underline{f} \qquad \text{where} \qquad \begin{array}{l} K = \int_V B^T D B \,\mathrm{d}V \\ \underline{f} = \int_S N^T \overrightarrow{t} \,\mathrm{d}S + \int_V N^T \overrightarrow{b} \,\mathrm{d}V \\ \underline{a} \quad \text{contains the displacements at the nodes} \end{array} \tag{2.49}$$

2.8.3 Boundary conditions and Solution

If there are n nodes in a three dimensional mesh, equation 2.49 describes a $3n \times 3n$ system. The size of the system can be reduced by the number of essential boundary conditions that are specified, and the resulting reduced system can be solved using any of the conventional methods of solving linear system. The solution produces a displacement vector for every node in the mesh, and by interpolating the displacements between the nodes a continuous deformation field can be generated.

The handling of boundary conditions is described in detail in chapter 5, and the particular solution method used in this work is explained in chapter 6.

Chapter 3

Meshing Background

3.1 Introduction

The finite element method requires that a deformable object's geometry be represented by a mesh. A mesh provides a discrete model of an object using a set of node points and a set of elements. The elements define node connectivity and determine how a finite element solution is interpolated between the nodes. A wide variety of mesh types exist, differing in element shape and size, interpolation order, and other factors.

This chapter first reviews the basic design options that are available in mesh construction and then surveys the main categories of mesh generation algorithms. Finally, work in medical imaging that has included finite element analysis and meshing is discussed. The intent is to present sufficient background information for the reader to understand the novel meshing algorithm presented in chapter 4 in the context of existing mesh generation algorithms.

3.2 Mesh Design Options

Meshes can be characterized by the type of elements that compose them and by the way mesh elements fit together. Each of the following five sections discusses an aspect of mesh construction by examining individual element attributes or by looking at possible ways elements can be joined to form a mesh.

3.2.1 Element Shape

One of the defining characteristics of an element is its shape. In two dimensions, triangles and quadrilaterals are common element shapes. In three dimensions, tetrahedral and hexahedral elements are predominant, while wedge and pyramid elements are used less commonly. Tetrahedra, pyramid, and hexahedra element topology is illustrated in Fig. 3.1. In the context of meshing, terms such as hexahedra and pyramid refer to element topology, not geometry. For example, a hexahedral element has the same topology as a cube but is not constrained to have planar faces or perpendicular edges. Although many meshes consist entirely of elements with the same shape, mixed meshes containing a variety of element shapes are equally valid.

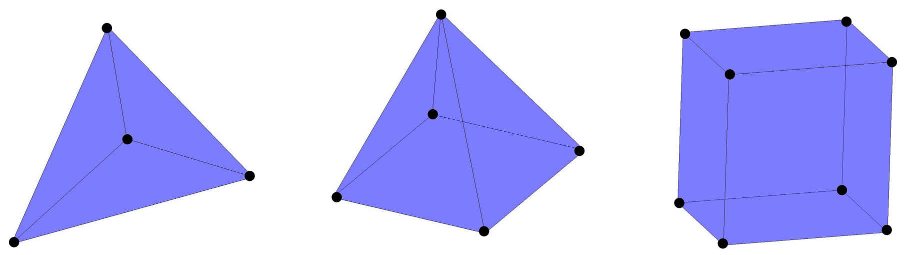


Figure 3.1: Linear three-dimensional elements. Left: Tetrahedral element Center: Pyramid element Right: Hexahedral element

For many finite element applications meshes constructed with hexahedral elements are preferable to meshes built from tetrahedral elements due to their superior convergence and accuracy characteristics. Research has shown that for both linear elastic problems and non-linear elasto-plastic problems the error in a finite element solution is less for a mesh of linear hexahedral elements than for a mesh of similarly sized linear tetrahedral elements [8]. However, currently available automatic meshing algorithms are more successful at constructing quality tetrahedral meshes than quality hexahedral meshes. The development of automatic hexahedral meshing algorithms is a challenging problem that motivates current research efforts in the meshing community [62].

3.2.2 Isoparametric Elements

One of the advantages of finite element meshes is their ability to conform to the boundaries of objects that have arbitrary geometry. This capability is made possible through the use of isoparametric elements. Isoparametric elements exist in two coordinate spaces. First, an isoparametric element exists in its parameter space, where it maintains its ideal shape. Secondly, it exists in world space, where its shape may be a rotated, stretched, and skewed version of its ideal shape. This arrangement provides the freedom to position elements in world space so that their edges and faces are aligned with object boundaries rather than with the coordinate axes. The world space and parameter space of a hexahedral element is illustrated in Fig. 3.2.

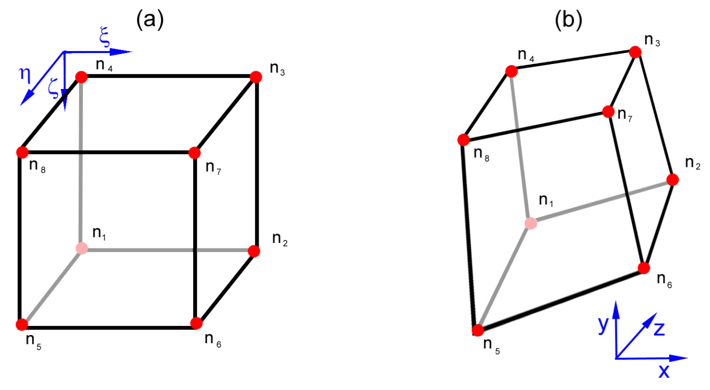


Figure 3.2: (a) Hexahedral element in its (ξ, η, ζ) parameter space (b) Hexahedral element mapped into world space

Each of an isoparametric element's nodes is assigned both parameter space and world space coordinates, and a mapping function is defined to map points from the parameter space into world space. The mapping function is simply a node based interpolation function. Applied to the nodes' world space coordinates, the mapping function generates the world space coordinates that correspond to any parametric point in an element. For a linear hexahedral element, the mapping function is as follows.

$$N(\xi,\, \eta,\, \zeta) \;=\; \sum_{i=1}^{8} N_i(\xi,\, \eta,\, \zeta) \cdot [x_i\; y_i\; z_i] \tag{3.1}$$

where $[x_i\; y_i\; z_i]$ are the coordinates of node i and

N_i is defined as

$$\begin{array}{ll} N_1 = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta) & N_2 = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta) \\ N_3 = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta) & N_4 = \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta) \\ N_5 = \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta) & N_6 = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta) \\ N_7 = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta) & N_8 = \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta) \end{array} \tag{3.2}$$

The same mapping function is used to interpolate a finite element solution across an element's interior space, except that $[x_i\ y_i\ z_i]$ node coordinates are replaced by the computed solution at each node. For the three-dimensional deformation problem, the desired result is a continuous vector field that represents the deformation an object undergoes as it changes shape. The computed solution comes in the form of a displacement vector for each node, and the mapping function is used to interpolate the displacement vectors across the interior of an element to provide the continuous vector field.

Although the flexibility afforded by isoparametric elements is beneficial because it allows finite element meshes to more accurately model object geometry, it is important that elements not be too distorted. If an element's shape in world space deviates too dramatically from its ideal shape, some of the assumptions made by the finite element method become invalid. Therefore, mesh generation algorithms must be concerned with measuring and controlling the quality of element shape. There are a number of metrics for element shape quality. For hexahedral elements one of the common element shape metrics is the determinant of the Jacobian of the mapping function. Intuitively, the determinant of the Jacobian is a measure of element volume. When an element is inverted, the volume becomes negative and the mapping between the parameter space and world space folds. As a hexahedral element's shape approaches that of a cube, the scaled value of the determinant of the Jacobian tends to 1 since the det(J) value approaches the element volume as computed by the product of three edge lengths.

In order for a valid finite element solution to exist on a mesh, the determinant of the Jacobian of the shape function $N(\xi, \eta, \zeta)$ must be non-negative everywhere [7]. When the determinant of the Jacobian is divided by the product of three edge lengths, it is scaled to the interval [-1, 1]. Although a solution can be computed as long as det(J) is non-negative, desirable elements have a scaled Jacobian value of at least .5. Smaller scaled det(J) values are associated with strongly distorted elements that can negatively impact the condition number of the finite element stiffness matrix and reduce the accuracy of the computed solution. It is sufficient to check the det(J) values at the Gauss integration points in each element [34]. The computation of the finite element stiffness matrix involves the numerical evaluation of an integral over the volume of the entire mesh, and the Gauss integration points are the locations inside each element where this integral is evaluated.

3.2.3 Element Size and Order

So far only linear elements have been discussed, but in fact quadratic, cubic, and higher order elements can be used in the construction of a finite element mesh. These higher order elements are so named because their mapping functions N_i are polynomials of the specified degree. Typically, higher order elements have nodes not only at their corners but also along their edges. For example, in two dimensions a quadratic quadrilateral element (Fig. 3.3(a),(b)) has 8 nodes, and each side of the element is a quadratic curve that interpolates the three nodes that lie along that side. The interpolation function for the quadratic quadrilateral element, given below, is a second order polynomial in both x and y.

$$N(\xi, \eta) = \sum_{i=1}^{8} N_i(\xi, \eta) \cdot [x_i \; y_i] \tag{3.3}$$

$$\begin{array}{ll} N_1 = -\frac{1}{4}(1-\xi)(1-\eta)(1+\xi+\eta) & N_2 = -\frac{1}{4}(1+\xi)(1-\eta)(1-\xi+\eta) \\ N_3 = -\frac{1}{4}(1+\xi)(1+\eta)(1-\xi-\eta) & N_4 = -\frac{1}{4}(1-\xi)(1+\eta)(1+\xi-\eta) \\ N_5 = \frac{1}{2}(1-\xi^2)(1-\eta) & N_6 = -\frac{1}{2}(1+\xi)(1-\eta^2) \\ N_7 = \frac{1}{2}(1-\xi^2)(1+\eta) & N_8 = \frac{1}{2}(1-\xi)(1-\eta^2) \end{array} \tag{3.4}$$

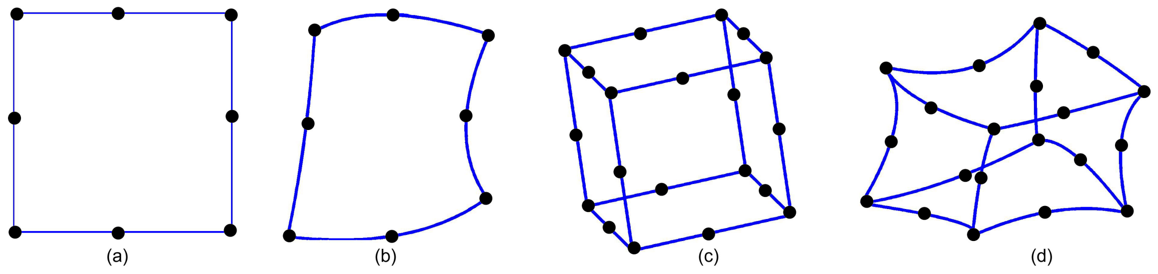


Figure 3.3: (a) 8-node quadratic quadrilateral element as it appears in parametric space (b) 8-node quadratic quadrilateral element as it could appear in world space (c) 20-node quadratic hexahedral element as it appears in parametric space(d) 20-node quadratic hexahedral element as it could appear in world space

The quadratic hexahedral element has 20 nodes and the following mapping function (3.3(c),(d)) :

$$N(\xi,\, \eta,\, \zeta) = \sum_{i=1}^{20} N_i(\xi,\, \eta,\, \zeta) \cdot [x_i\ y_i\ z_i] \tag{3.5}$$

$$\begin{array}{ll}
N_1 = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta)(-\xi-\eta-\zeta-2) & N_2 = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta)(\xi-\eta-\zeta-2) \\
N_3 = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta)(\xi+\eta-\zeta-2) & N_4 = \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta)(-\xi+\eta-\zeta-2) \\
N_5 = \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta)(-\xi-\eta+\zeta-2) & N_6 = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta)(\xi-\eta+\zeta-2) \\
N_7 = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)(\xi+\eta+\zeta-2) & N_8 = \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta)(-\xi+\eta+\zeta-2) \\
N_9 = \frac{1}{4}(1-\xi^2)(1-\eta)(1-\zeta) & N_{10} = \frac{1}{4}(1+\xi)(1-\eta^2)(1-\zeta) \\
N_{11} = \frac{1}{4}(1-\xi^2)(1+\eta)(1-\zeta) & N_{12} = \frac{1}{4}(1-\xi)(1-\eta^2)(1-\zeta) \\
N_{13} = \frac{1}{4}(1-\xi^2)(1-\eta)(1+\zeta) & N_{14} = \frac{1}{4}(1+\xi)(1-\eta^2)(1+\zeta) \\
N_{15} = \frac{1}{4}(1-\xi^2)(1+\eta)(1+\zeta) & N_{16} = \frac{1}{4}(1-\xi)(1-\eta^2)(1+\zeta) \\
N_{17} = \frac{1}{4}(1-\xi)(1-\eta)(1-\zeta^2) & N_{18} = \frac{1}{4}(1+\xi)(1-\eta)(1-\zeta^2) \\
N_{19} = \frac{1}{4}(1+\xi)(1+\eta)(1-\zeta^2) & N_{20} = \frac{1}{4}(1-\xi)(1+\eta)(1-\zeta^2)
\end{array} \tag{3.6}$$

The accuracy of a finite element solution can be controlled by adjusting either the element size or the element order. The finite element solution will converge to the exact solution of the PDE as the element size tends to zero or as the order of approximation tends to infinity. This makes sense intuitively because by creating either smaller elements or higher order elements the number and density of nodes is increased. A greater number of nodes means the solution is calculated at more points. It should be noted that an increase in the number of nodes also leads to increased computational cost, so the solution accuracy is always limited by the available computation time.

With all meshing algorithms a design choice must be made between constructing a mesh with a greater number of linear elements or a smaller number of higher order elements. In adaptive meshing algorithms the number of nodes is increased until a desired level of accuracy is reached. If as the node count is increased smaller linear elements are created, the method is referred to as an h-adaptive method. If instead the approximation order of the elements is increased with the addition of new nodes, the method is known as p-adaptive.

3.2.4 Consistency and Continuity

In order for a finite element solution to be well defined, it must be uniquely determined at each point in the mesh and vary smoothly across element boundaries as well as inside elements. This means that all of the element mapping functions must stitch together to form a composite mapping function that has no holes, no folds, and is C^0 continuous over the entire meshed volume. To ensure that these conditions are met, care must be taken with the way elements are joined to their neighbors.

First, there is the obvious requirement that neighboring element volumes have no overlaps and no gaps between them. Second, if a face or edge is shared between neighboring elements, the mapping function applied to the face or edge by each of the elements must be identical. Otherwise, the inconsistency in the mapping functions would cause the solution to be discontinuous across the element boundary.

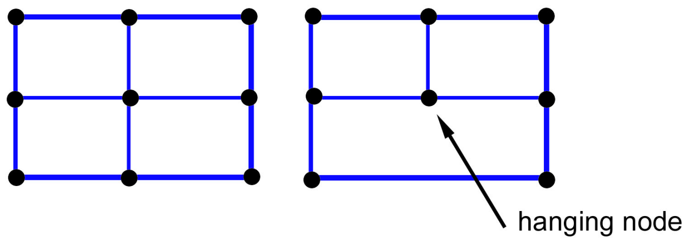


Figure 3.4: Left: A mesh segment with no hanging nodes Right: The center node is a hanging node. Each of the top two elements share half of the bottom element's upper edge, but not the whole edge.

Finite element meshing schemes typically disallow hanging nodes (see Fig. 3.4) in order to guarantee mapping function consistency across element boundaries. A hanging node occurs if two elements share part of a face or edge, but not the whole face or edge. With hanging nodes along the shared interface the mapping functions applied by the neighboring elements are different since the elements are performing an interpolation between different sets of nodes.

3.2.5 Continuous vs. Overset Meshes

The only situation in which it is acceptable for mesh elements to overlap is if an overset meshing strategy, also known as a chimera mesh, is employed. An overset meshing scheme allows separate meshes to be constructed for neighboring objects or for different sections of the same object. Each of the separate meshes must overlap with one or more of the other meshes in the neighborhood of its interior edges so that no unmeshed holes exist in the area of interest. Fig. 3.5 shows an example of a two dimensional overset mesh.

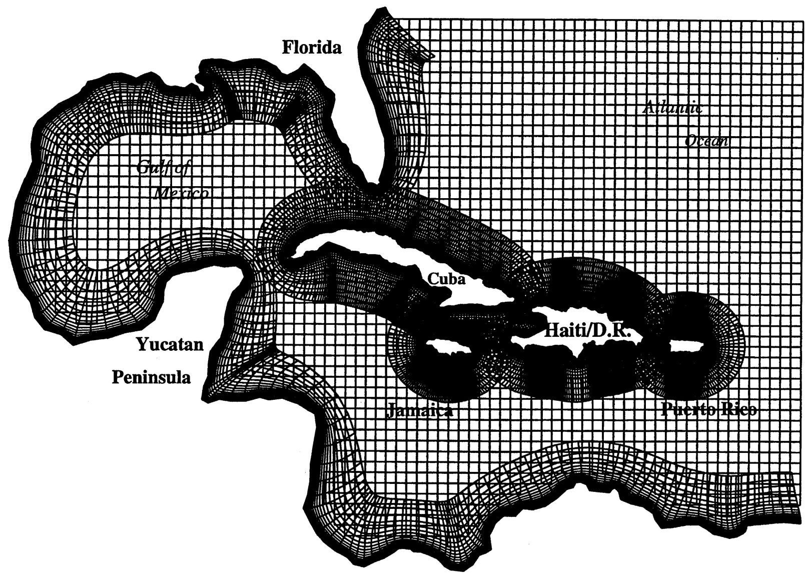


Figure 3.5: An example of an overset grid of the Gulf of Mexico. Notice that the sections of boundary fitted grid around the coastline overlap with the background Cartesian grid and overlap with each other in spots. Figure copied from [18].

If an overset mesh is employed, in the solution phase an iterative method is used with each of the separate mesh sections. A special interpolation procedure is used to propagate the solution between neighboring mesh sections at their overlapped boundaries after each solution iteration. This special interpolation step maintains the consistency of the solution between each of the meshes, but it also involves a significant amount of computational overhead in the solution phase. It has the advantage of allowing very small mesh elements to be used in some regions and large elements used in adjacent regions without dealing with the difficulty of patching together meshes with disparate element sizes [42].

3.3 Meshing Algorithms

The two main categories of mesh generation algorithms are structured methods and unstructured methods. The term structured is applied to meshes that have the same topology as a Cartesian grid. Because the topology of a structured mesh is fixed, it can be efficiently stored as an $i \times j$ array of nodes in two dimensions and as an $i \times j \times k$ array of nodes in three dimensions. The node connectivity of a structured mesh is implicit in the data structure. See Fig. 3.6 for an illustration.

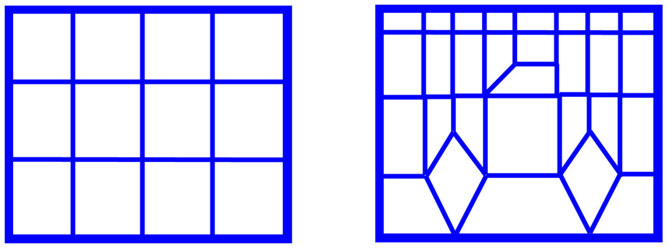


Figure 3.6: 3D meshes constructed with linear quadrilateral elements. Left: A structured example Right: An unstructured example

Structured meshing algorithms are typically based on a curvilinear coordinate system that is imposed on the object being meshed. Structured meshing algorithms differ in the way they create such a coordinate system within a non-rectangular object; several of these methods will be discussed in the following section. Quadrilateral and hexahedral elements can be easily used in a structured mesh.

An unstructured mesh allows a wider variety of node connectivity patterns and explicitly stores the node list for every element. Unstructured meshing algorithms have the advantage of being able to work with complicated object geometry within which it may be difficult or impossible to generate a Cartesian coordinate system. Unstructured meshes most often contain triangle or tetrahedral elements, although there are some unstructured algorithms that generate predominantly quadrilateral or hexahedral elements [62].

Not all meshing methods can be clearly categorized as either structured or unstructured. In between the structured and unstructured categories are a number of block structured and hybrid methods. A block structured method divides a complicated object into pieces called blocks, defines a grid pattern for each of the grid interface surfaces, and then generates a structured mesh for each block that matches the specified interface patterns where neighboring blocks abut. This strategy allows structured meshing techniques to be applied to more complicated and realistic objects than would otherwise be possible.

A hybrid method employs both structured and unstructured techniques to form a single mesh. In some instances this takes the form of creating a structured mesh of separate objects or regions and then creating an unstructured mesh between the structured pieces in order to patch all the regional meshes together into one continuous mesh [63].

In order to provide a context for the m-rep based meshing algorithm presented later in this chapter, several well-established structured and unstructured meshing algorithms are briefly described and compared in the following sections. For a more thorough treatment of each of these algorithms, the Handbook of Grid Generation is an excellent resource [3].

3.3.1 Structured Meshing Algorithms

3.3.1.1 Algebraic Mesh Generation

There are parallels between algebraic mesh generation and the development of isoparametric elements. In both cases an interpolation function is used to establish a correspondence between points in a parametric domain and points in a physical domain. With isoparametric elements a parametric domain represents a single element, while an algebraic mesh generation algorithm represents an entire meshed volume with a single parametric domain.

The first step in algebraic mesh generation is the formation of a Cartesian grid in the parametric domain that has the desired number of rows and columns of elements. In two dimensions this could look like a square filled with rows and columns of quadrilateral elements, and in three dimensions this could look like a cube filled with rows, columns, and layers of hexahedral elements. In the parametric domain these elements have their ideal shape; the quadrilaterals are squares and the hexahedra are cubes.

The second step is specifying a correspondence between the boundary of the parameter space grid and the boundary of the physical domain that needs to be meshed. The physical domain does not need to be a square or cube, but corners and edges must be defined on the surface of the physical domain so that it can be represented with the same logical structure as the boundary of the parameter space grid. An example of boundary correspondence between a square domain and a curved domain is shown in Fig. 3.7. Depending on the mapping function chosen in the next step, boundary tangent vectors or intermediate edge points may also need to be specified.

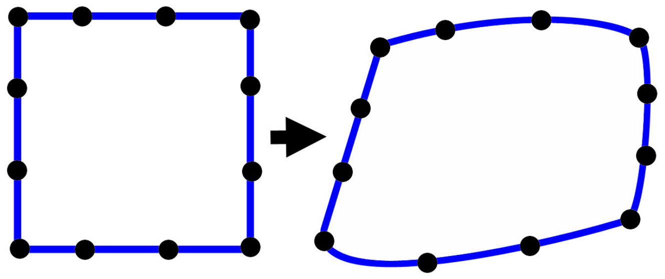


Figure 3.7: Algebraic and elliptic mesh generation algorithms require boundary correspondences to be established between a parametric domain (left) and a physical domain (right).

Third, a mapping function must be formulated that establishes a one to one correspondence between each parametric point in the parameter space grid and a point in the physical domain. Typically the mapping function is simply a polynomial function that interpolates the coordinates of grid's corner points across the parametric grid. The order of the mapping function does not have to be the same for each of the spatial dimensions. For example, the x coordinates could be interpolated cubically and the y coordinates linearly. Lagrangian or Hermite interpolation formulas may also be used if sufficient boundary derivatives or intermediate points are specified in the second step.

Finally, the Cartesian grid is transformed from the parametric domain to the physical domain by computing the physical coordinates of each of the nodes in the parametric domain via the mapping function. The mesh maintains the same structured topology as the Cartesian grid has in the parametric domain. This is illustrated in Fig. 3.8.

The advantage of algebraic mesh generation is that it offers a computationally efficient and straightforward method of generating a structured grid of quadrilateral or hexahedral elements. For some objects it can offer a quality mesh, but for objects with significant curvature, concavity, or holes, the resulting mesh may not be usable. Algebraically generated meshes can contain excessively skewed elements, and it is possible for them to fold and extend outside the boundaries of an object. Therefore,

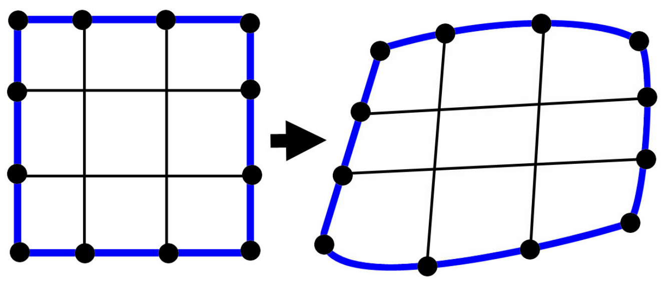


Figure 3.8: A quadrilateral mesh linearly interpolated from parametric domain to physical domain

algebraically generated meshes are frequently used to provide an initial mesh that is iteratively improved by the application of an elliptic mesh generation algorithm. More information on algebraic mesh generation can be found in [66].

3.3.1.2 Elliptic Mesh Generation

Elliptic generation algorithms create structured meshes that typically have greater smoothness and mesh line orthogonality than algebraically generated meshes; yet the initial steps in elliptic mesh generation are similar to those for algebraic mesh generation. For both meshing algorithms the first step is to create a Cartesian grid in a parametric space and establish a correspondence between the boundary of the parameter domain and the physical domain to be meshed. The elliptic mesh generation algorithm, however, differs from the algebraic algorithm in the method used to transform coordinates from the parameter space to the physical space.

Instead of using polynomial functions to interpolate the physical coordinates of mesh nodes, a system of elliptic partial differential equations is used to define a transformation between the parametric domain and the physical domain. The simplest example of such a system is the Laplace equation. The Laplace equation is notable for generating smooth meshes; it is the elliptic equation that will generate the smoothest mesh possible for a given domain [75]. In the Laplace equation given below, ξ^1, ξ^2, ξ^3 are the coordinate directions in the three dimensional parametric space and x^1, x^2, x^3 are the world space coordinate directions.

$$\nabla^2 \xi^i = 0 \text{ where } i = 1, 2, 3 \tag{3.7}$$

Expanded, the equation is as follows:

$$\begin{aligned}
\left(\frac{\partial^2 \xi_1}{\partial {x_1}^2} + \frac{\partial^2 \xi_1}{\partial {x_2}^2} + \frac{\partial^2 \xi_1}{\partial {x_3}^2}\right) &= 0 \\
\left(\frac{\partial^2 \xi_2}{\partial {x_1}^2} + \frac{\partial^2 \xi_2}{\partial {x_2}^2} + \frac{\partial^2 \xi_2}{\partial {x_3}^2}\right) &= 0 \\
\left(\frac{\partial^2 \xi_3}{\partial {x_1}^2} + \frac{\partial^2 \xi_3}{\partial {x_2}^2} + \frac{\partial^2 \xi_3}{\partial {x_3}^2}\right) &= 0
\end{aligned} \tag{3.8}$$

These equations establish a relationship between the parametric and physical coordinates. In order for the system to be solvable, boundary conditions must be specified. This requirement is met by the correspondence between the parametric and physical domain boundaries that is one of the inputs to the method. An example of a mesh generated with the Laplace equation is shown in Fig. 3.9.

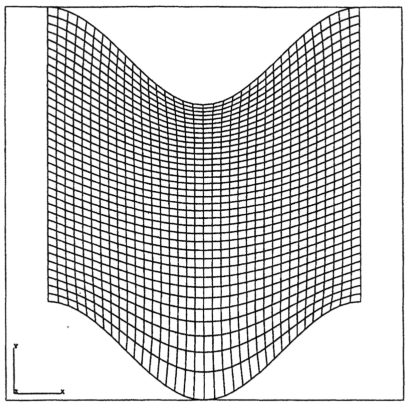


Figure 3.9: An elliptically generated grid using the Laplace equation. Figure is from [68].

Because it is more convenient to compute the solution to the system in the parameter space than in the physical space, the equations are transformed so that the spatial coordinates x^i are the dependent variables and the parametric coordinates ξ^i are the independent variables. The resulting system of equations can be solved on a regular grid in the parameter space using a standard PDE solver. The solution gives physical space coordinates for each of the nodes of the parameter space mesh, effectively creating a new mesh fit to the arbitrarily shaped physical domain. These PDE solvers arrive at a solution by iteratively improving an initial guess. For elliptic mesh generation, the usual strategy is to take an algebraically generated mesh as the initial guess for the solution and then iteratively improve the spatial coordinates of the mesh nodes until a convergence criteria is met.

The usefulness of the Laplace equation in mesh generation is limited by the fact that with this equation there is no way to control the spacing or slope of the mesh lines. More control is provided by the widely used Poisson's equation for mesh generation, given below.

$$\nabla^2 \xi^i = P^i \tag{3.9}$$

The control functions P^i may be defined in a number of different ways and are designed to achieve certain desired properties in the resulting grid such as orthogonality at domain boundaries or variable spacing of mesh lines. For some methods the control functions are determined based solely on initial boundary conditions, while for other methods the control functions are iteratively adjusted to achieve the desired mesh properties.

Much research has been devoted to developing variations and improvements to the basic elliptic mesh generation algorithm. A good overview of elliptic algorithms can be found in [68]. A briefer treatment can be found in [27]. A considerable amount of effort has been invested in developing elliptic methods because the meshes generated can be of very good quality. With an elliptic method it is straightforward to create a mesh with quadrilateral or hexahedral elements. The meshes tend to be quite smooth, boundary orthogonality of mesh lines can be achieved, and element size and shape can be controlled, all of which are attractive qualities for a meshing algorithm. A main disadvantage is that the method can be quite time consuming, especially in three dimensions, due to the necessity of iteratively solving the partial differential mesh generation equations. Another limiting factor is the need to establish a correspondence between the boundary of the physical domain and the boundary of the parametric domain as an input to the meshing algorithm. Depending on the geometry of the object to be meshed, this requirement can be difficult to meet.

3.3.1.3 Hyperbolic Mesh Generation

The input to a hyperbolic mesh generation algorithm is an initial mesh boundary with predefined node points. In two dimensions the mesh boundary is a curve, and in three dimensions it takes the form of a surface. A hyperbolic mesh algorithm is used to generate a mesh outward from the boundary, unlike algebraic and elliptic algorithms that generate a mesh for the interior of a closed region. The hyperbolic algorithm operates by beginning at the initial mesh boundary and marching outward in the direction normal to the boundary to create a new layer of nodes. The mesh is formed by creating and connecting successive layers of nodes until the area of interest is filled. Because these type of meshes fill an exterior space, they are especially well suited for problems that involve modeling fluid flow around an object. An example of such a hyperbolic mesh is shown in Fig. 3.10.

The node positions on each new layer are found by solving the governing hyperbolic equations. These equations place constraints on the orthogonality of mesh lines and require that element volumes match those specified by a user defined volume distribution function. The following hyperbolic field generation equations can be used to generate a mesh in three dimensions.

$$\begin{aligned}
\frac{\partial x_1}{\partial \xi_1}\frac{\partial x_1}{\partial \xi_3} + \frac{\partial x_2}{\partial \xi_1}\frac{\partial x_2}{\partial \xi_3} + \frac{\partial x_3}{\partial \xi_1}\frac{\partial x_3}{\partial \xi_3} &= 0 \qquad (3.10)\\
\frac{\partial x_1}{\partial \xi_2}\frac{\partial x_1}{\partial \xi_3} + \frac{\partial x_2}{\partial \xi_2}\frac{\partial x_2}{\partial \xi_3} + \frac{\partial x_3}{\partial \xi_2}\frac{\partial x_3}{\partial \xi_3} &= 0\\
\frac{\partial x_1}{\partial \xi_1}\frac{\partial x_2}{\partial \xi_2}\frac{\partial x_3}{\partial \xi_3} + \frac{\partial x_1}{\partial \xi_3}\frac{\partial x_2}{\partial \xi_1}\frac{\partial x_3}{\partial \xi_2} + \frac{\partial x_1}{\partial \xi_2}\frac{\partial x_2}{\partial \xi_3}\frac{\partial x_3}{\partial \xi_1} & \qquad (3.11)\\
-\frac{\partial x_1}{\partial \xi_1}\frac{\partial x_2}{\partial \xi_3}\frac{\partial x_3}{\partial \xi_2} - \frac{\partial x_1}{\partial \xi_2}\frac{\partial x_2}{\partial \xi_1}\frac{\partial x_3}{\partial \xi_3} - \frac{\partial x_1}{\partial \xi_3}\frac{\partial x_2}{\partial \xi_2}\frac{\partial x_3}{\partial \xi_1} &= \triangle V
\end{aligned}$$

where $\triangle V$ is the element volume

This system of equations can be solved with a non-iterative method, resulting in an algorithm one or two orders of magnitude faster than the elliptic mesh generation algorithms that require an iterative solver. A finite difference solution technique that can be employed to solve the hyperbolic equations is described in [18].

Hyperbolic mesh generation offers the advantages of mesh orthogonality (in some dimensions), computational efficiency, and user controllable element sizes. A potential disadvantage is that the generated mesh may not be smooth for certain combinations of initial boundaries and element size distribution functions. In certain situations, especially when meshing concavities, it is possible for very elongated elements to be formed. This tends to occur when the mesh lines in the direction normal to the surface are forced to lie close together because they proceed out from a concave surface, and the mesh lines in the perpendicular direction must have much wider spacing in order to satisfy the element volume requirement.

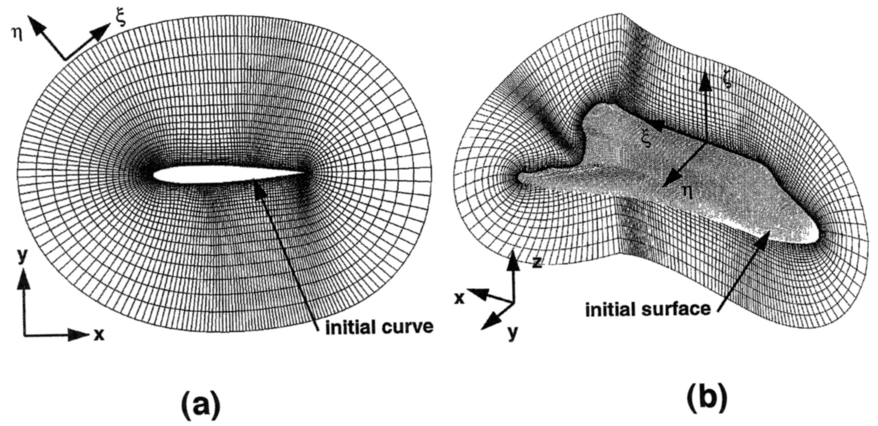


Figure 3.10: Hyperbolically generated meshes in (a) two dimensions and (b) three dimensions. Figure is from [18].

3.3.2 Unstructured Meshing Algorithms

3.3.2.1 Delaunay Based Meshing

Delaunay methods produce a mesh composed of triangle elements in two dimensions or tetrahedral elements in three dimensions. These methods require as input a set of points distributed throughout an object to be meshed. The input points may lie on the boundary and in the interior of the object being meshed, and their spacing determines the resolution of the mesh. A variety of general and problem specific algorithms have been developed to generate the required point distribution, including regular and random sampling of the problem domain.

The Delaunay mesh is constructed by connecting the input points to form elements. The Delaunay tesselation can be explained as the dual of the Voronoi graph of the input points. Briefly, the edges of the Voronoi graph form convex polygons around each of the input points such that each input point's polygon encloses the space that is closer to that point than any of the other input points. Generically, in two dimensions each edge in the graph is equidistant from two input points, and each vertex is equidistant from three input points. Likewise, in three dimensions each face is equidistant from two input points, each edge is equidistant from three input points, and each vertex is equidistant from four input points. In constructing the dual of a Voronoi graph, each vertex of the graph becomes the center of an element. Each of the input points equidistant from a Voronoi vertex is connected to form an element. In two dimensions the three equidistant points become the nodes of a triangle element, and in three dimensions the four equidistant points become the nodes of a tetrahedral element. Algorithms exist that construct the Delaunay mesh in $O(n \log n)$ time. Details of these algorithms can be found in [50].

Frequently the initial Delaunay mesh must be adjusted before it is suitable for use in finite element analysis. There are two main problems that may occur. First, the edges of the input object may not be contained as edges of the mesh. If this happens, the mesh representation distorts the object's shape. See Fig. 3.11 for an example. The second potential problem is the existence of elements with poor shape, especially sliver like elements with very small volume. Delaunay based meshing algorithms attempt to remedy these problems by adjusting the mesh in one or more of the following three ways:

1. Extra input points can be added, either on the boundary or in the interior of the object. The addition of an extra point will change the mesh in the neighborhood of the new point. Extra points are often added to force the inclusion of a boundary edge in the mesh.

2. If the exterior boundary of a group of elements sharing an edge or face is convex, the shared edge or face can be deleted and the resultant convex polygon can be subdivided into elements in a different way. This technique can be used to remove poorly shaped elements.

3. A smoothing process can be employed to improve element shape. In some implementations a Laplacian smoothing routine is used. In other cases node positions are adjusted in an attempt to make all the edges incident on a vertex close to the same length.

Delaunay based meshing algorithms have the advantage of being more robust and more fully automatic than many of the other meshing algorithms. Delaunay methods can also be computationally efficient, but they are limited to the production of triangular and tetrahedral elements. Other difficulties lie in getting uniform element size and controlling the shape and orientation of the elements. Post-processing adjustment of the mesh can help with these problems, but guarantees cannot usually be made regarding final element quality. A more thorough review of Delaunay-Voronoi meshing algorithms is provided in [6].

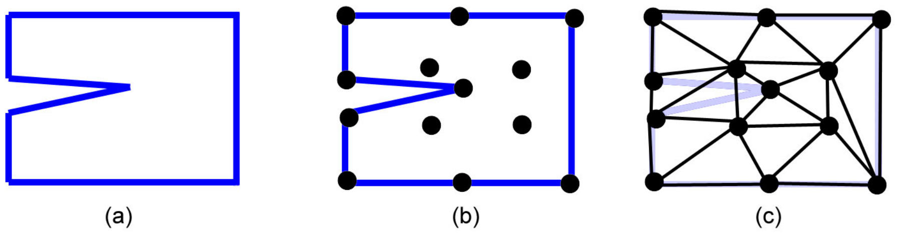


Figure 3.11: (a) The boundary of a 2D object to be meshed (b) A distribution of points in the object and on its boundary (c) The Delaunay triangulation of the points. Note that the edges of the object's concave wedge are not included as edges of the triangulation, so the mesh does not accurately represent the object's geometry.

3.3.2.2 Advancing Front, Paving, and Plastering

Advancing front, paving, and plastering algorithms share the same basic approach to mesh generation, but they differ in the element types they create. Advancing front algorithms typically create triangle or tetrahedral element meshes, while paving algorithms create quadrilateral element meshes and plastering algorithms create hexahedral element meshes. An overview of the advancing front methodology is presented first, followed by comments about the differences in paving and plastering algorithms.

Unlike Delaunay methods that first begin with a set of nodes and then generate the elements, advancing front meshing algorithms generate nodes and elements simultaneously. The input to an advancing front algorithm is a discretized representation of the object boundary. In two dimensions this means a boundary defined by line segments and in three dimensions by a triangulated surface. Each boundary element must be accompanied by a normal, to differentiate the interior side from the exterior side.

The key data structure in advancing front algorithms is the active front, which contains the collection of oriented edges (2D) or faces (3D) that bound an open, unmeshed portion of the object interior and are available to form a side of a new element. Initially, the active front is the discretized boundary. After each new element is created, the active front data structure is updated by removing the base face of the newly formed element and possibly some other faces and adding one or more of the faces of the new element to the active front. The algorithm terminates when the active front is empty, signalling a completed mesh.

The element creation process is as follows:

- an edge(2D) or face(3D) belonging to the active front is selected
- either a node near the selected edge/face is selected, or if no suitable node exists a new node is created and selected
- edges are created to connect the selected node with the selected edge/face so that a new element is formed
- the active front is updated to include newly created edges/faces and any newly inactivated faces are removed

The most complicated part of this process is the decision of which node to select, or whether to create a new node to form the element. The first step in making this decision is calculating the ideal location for the node that will be connected to the selected edge/face.

The ideal node location is a function of the desired element size and shape as well as the size and orientation of the selected edge/face. In the general case, the desired element size can be a directional quantity that varies across the domain, leading to the creation of anisotropic elements with graduated sizes.

Computing the ideal node location is simpler if a unit size element is desired with equilateral angles. A transformation T can be defined that is a function of the element size parameters and will transform the desired element into a normalized space where it has unit size and equilateral angles. By applying the transformation T to the currently selected face, the ideal node location for the formation of the next element can be computed in the normalized space. Applying the inverse of T to the ideal node location in the normalized space yields the physical coordinates of the ideal node location. In the normalized space the ideal node location lies along the surface normal that extends from the center of the face, and a point along that line is chosen such that edges of unit length or a tetrahedra of unit volume is created.

After computing the ideal node location, a list of nodes that lie within a certain radius of the ideal coordinates is created and sorted by distance from the ideal coordinates. The first node on this list that could be used to form an element whose faces do not intersect any existing element faces is selected. If no elements on this list meet the criteria, a new element is created. Fig. 3.12 illustrates an advancing front algorithm. More detail on advancing front algorithms can be found in [53].

Like the Delaunay meshes, tetrahedral advancing front meshes can be improved by post-processing. Both the edge/face swapping and the smoothing routines described in the Delaunay meshing section can be applied to advancing front meshes.

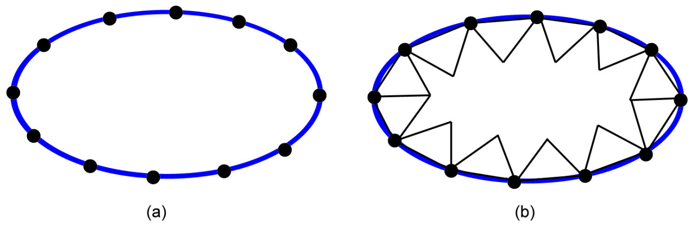


Figure 3.12: (a) An object boundary with boundary node points (b) The active front of the mesh, after the first layer of elements has been created.

Paving and plastering algorithms tackle the more difficult problems of meshing arbitrary regions with quadrilateral or hexahedral elements, respectively. The difficulty with using quad and hex elements in an advancing front algorithm lies in finding a way to join opposing fronts when they collide. With triangles or tetrahedra this is simple, since a valid element is formed by connecting a base face from one front with a single node from the opposing front. To form a quad element an edge from each of the opposing fronts must be connected, and to form a hex element a quadrilateral face from each of the opposing fronts must be connected. Matching edges and faces turns out to be a more difficult problem than matching a face with a node.

The paving algorithm [81] has been made robust by checking for front collision after the creation of every element with proximity and edge intersection tests. If collision or impending collision is detected, one of three operations is performed to join the colliding parts of the front.

- *Seaming* joins two faces from the same region of the advancing front that collide due to the front's high curvature in the region.
- *Splitting* results in one continuous advancing front being split into two separate fronts, due to the collision of two opposing parts of the front. The colliding parts are joined, and the two new fronts lie on either side of the joined region.
- *Connecting* occurs when two separate fronts collide and are joined into one front.

Plastering [43] by hexahedral elements alone has failed to develop into a robust algorithm because its active front can reach a condition where the remaining unmeshed space is impossible to fill with hexahedra. The plastering concept is used in the Hex-Tet algorithm contained in the CUBIT meshing package, which adds hexahedral elements in an advancing front approach as far as possible and then fills the remaining space with tetrahedra [43] [1].

3.3.2.3 Superposition and Grid-Based Methods

Grid based methods superimpose a standard background mesh, commonly a Cartesian grid, on the space occupied by an object model. All the elements that lie outside the object boundary are removed, along with all the elements that are intersected by the object boundary or lie too close to the boundary. Typically, elements inside the object that are closer than half the length of an element edge are also removed. The result is an initial mesh that fits the inside of the object but has open space between the exterior of the mesh and the object boundary.

The next step in this meshing process is the most difficult. The mesh must be completed by adding a layer of elements to the outside of the initial mesh that conform to the shape of the object boundary.

A typical strategy involves creating a corresponding node on the object boundary for each node on the surface of the interior mesh. Node positions on the boundary may be chosen by projecting the interior nodes outward from the object center, by creating new nodes at the surface points closest to interior mesh nodes, or by some other algorithmic process. After nodes are generated, edges are formed to connect pairs of nodes, and edges are created between the new boundary nodes so that their connectivity mirrors that of the nodes on the initial mesh surface. These new nodes and edges form a layer of hexahedral elements that conform to the object boundary. The difficult part of this process is choosing good locations for the new nodes on the boundary surface. A very robust method must handle a variety of special case geometries in order to avoid mesh folding and degenerate elements.

Without post-processing to improve the shape of the outer layer elements, their quality is likely to be poor. The advantage of this method is that it gives good quality elements in the interior of the object, and it can generate meshes very quickly. Superposition methods are reviewed more thoroughly in [62]

3.3.2.4 Quadtree / Octree Methods

Spatial tree based meshing algorithms work in two stages. The first stage involves building the tree data structure from the input object model, while the second stage builds the mesh from the tree data structure. The first stage operates in a fairly standard manner for most tree based meshing algorithms, but there is great variety in the way the element construction in second stage is implemented. Therefore, a description of the first stage is provided, and a few examples of how different implementations handle the second stage are given.

Typically, quadtrees are used for two dimensional meshing problems and octrees are used for three dimensional problems. The quadtree and octree methods operate similarly, but since the focus of this chapter is on three dimesional meshing the octree algorithm will be described.

The octree building stage begins by forming a bounding cube around the object model. This bounding cube forms the root of the octree. If the root fails the subdivision termination criteria, the bounding cube is subdivided into eight equal size octants and the test/subdivide procedure is applied recursively to each of the octants. In the tree structure each octant is represented as a child of the octant one level larger that encompasses it.

The subdivision termination criteria could include several different tests. Typical reasons to terminate subdivision include the following:

- The object boundary does not intersect the octant.
- The object boundary intersects the octant and the geometric complexity of the portion of the boundary inside the octant is less than some limit. A measure of geometric complexity can be a function of the curvature of the portion of the boundary that lies inside an octant.
- A specified maximum tree level has been reached.

Some implementations require a maximum difference of one subdivision level between neighboring octants, and a few require all octants to be at the same subdivision level. These addition of constraints would require further octant subdivision before finalization of the tree structure.

As mentioned, the task of forming elements from tree octants can be approached in several different ways. The simplest approach is to make one hexahedral element out of each cell. If a tree contains terminal octants at different subdivision levels, the faces of the hexahedral elements will not be compatible with each of their neighbors' faces. This difficulty can be resolved by including additional constraint equations in the formulation of the finite element equations in order to account for the hanging nodes. A different solution avoids the problem altogether by requiring the entire domain be subdivided to the same level, but it pays the price of creating possibly many additional nodes and elements. A third solution allows a maximum one level difference between neighbor cells, and at each subdivision step it divides edges into thirds and creates 27 octants. Templates are available that provide a compatible hex mesh between one level different neighbors.

The fact that the cells do not conform to object boundaries also poses a potential problem for this method. Some implementations accept the jagged boundary that results, while others solve the problem using the same strategy described in the superposition section of adding a layer of boundary nodes and connecting them to the interior mesh nodes. As before, the problem with this approach is that the shape of the boundary elements can become very skewed.

Other element creation algorithms fill each tree octant with tetrahedra. Cells that lie completely in the interior of the object can be meshed using tetrahedral templates. Tetrahedral meshing patterns can be cataloged for each of the possible cell sizes and cell neighbor configurations. This approach is most practical if a maximum of one level difference between neighboring octants is permitted.

Another alternative is to mesh both interior cells and cells that are intersected by the model boundary using the Delaunay method. The cell vertices and the points where the cell and model intersect and be used as the input points to the Delaunay method. This provides a way to control the distribution of input points across the model so that the Delaunay method yields a mesh with tetrahedra of the desired size is each part of the model.

Other methods of creating elements from the tree structure exist; a broader survey of them can be found in [**?**].

As mentioned, some octree methods offer the advantage of hexahedral element creation, but like the superposition methods they can suffer from poor quality boundary elements. These octree meshes can often be improved by post-processing. As usual, smoothing can be applied to improve element shape. If tetrahedral elements are used, the edge/face swapping described in the Delaunay section can also be applied.

3.3.2.5 Medial Methods

Price and Armstrong [60] have developed a meshing algorithm based on an object's medial axis. By using the global shape information provided by the medial axis, this method can generate hexahedral meshes that conform to object boundaries and do not face the same difficulties with boundary elements as the superposition and octree methods. This medial method generates quadrilateral meshes in two dimensions and hexahedral meshes in three dimensions, but the discussion here will be limited to three dimensions. The quadrilateral algorithm is similar but simpler.

The algorithm relies on the medial axis, referred to as the medial surface in three dimensions, to subdivide a geometrically complex solid object into convex polyhedra for which hexahedral meshing templates have been defined. The medial surface can be described as the locus of the center of an inscribed sphere of maximal diameter as it rolls around the interior of an object. The medial surface together with the radius function defined on the medial surface is sufficient to exactly reconstruct the boundary of an object [10] [48]. The medial surface and radius function taken together are known as the medial axis transform (MAT). For this algorithm the medial axis is constructed using a constrained Delaunay tesselation of points distributed on the object model's boundary. The centers of some of the Delaunay tetrahedra will lie on the medial surface and provide a starting point for mapping the surface.

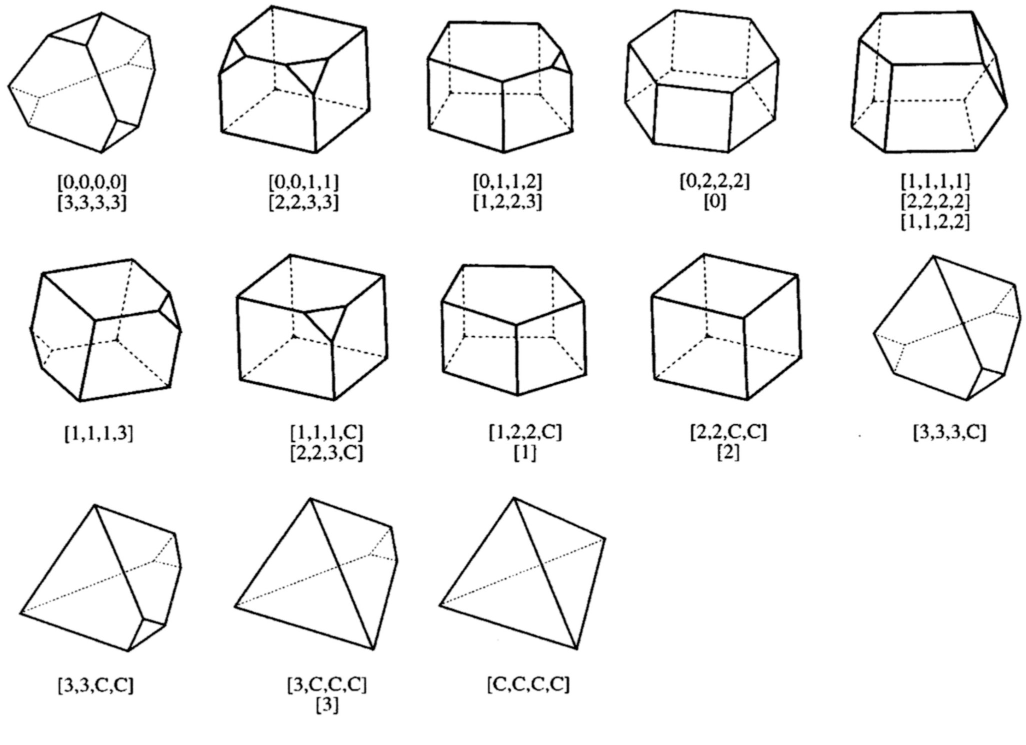


Figure 3.13: The 13 meshable solid primitives used in Price and Armstrong's medial meshing algorithm. Figure from [60].

The subdivision strategy is based on an analysis of the medial surface. First, the edges and vertices of the medial surface are classified based on their connectivity and the object's surface geometry. In this classification scheme there are four types of medial edges and 19 types of medial vertices. Meshable subregions are formed first around the edges and then around the vertices of the medial surface. Then, the faces of the medial surface are subdivided according to rules about the type of edges and vertices that bound a medial face. Finally, each subregion of the object can be identified as matching one of the 13 solid primitives shown in Fig. 3.13. Each of the 13 primitives is a convex polyhedron with at most 8 faces and can be further subdivided into hexahedral elements. Linear programming is used to match hexahedral element edges that lie on the faces of subregion boundaries.

Other than being quite complicated, a difficulty with this method is that the medial axis transform is not stable because slight changes in an object's boundary can lead to branching and significant changes in the medial surface. Therefore, small boundary changes can significantly affect the global meshing pattern. Nevertheless, this method is notable for being one of the first successful automatic hexahedral meshing algorithms for complicated geometry.

3.3.2.6 Spatial Twist Continuum and Whisker Weaving

The spatial twist continuum (STC) [28] [62] bears some similarity to the medial axis. The difference is that given a quadrilateral or hexahedral mesh, the STC runs through the middle of rows or columns of elements rather than through the middle of an object. In two dimensions, the STC is composed of chords, which are curves that run through the center of a quadrilateral element and intersect two of its opposing faces. At the center of each element is a chord intersection. Either two chords intersect, or a single chord intersects itself. A quadrilateral mesh can be constructed as the dual of an arrangement of chords, but the dual of every chord arrangement does not result in a valid mesh. Figs. 3.14, 3.15, and 3.16 illustrate the STC for two dimensional, three dimensional, and surface meshes, respectively.

In three dimensions the STC is composed of sheets, each of which runs through the center of a layer of hexahedral elements. The chord curves are formed by the intersection of two such sheets, and three sheets intersect at the center of a hexahedral element. As in two dimensions, a hexahedral mesh can be constructed as the dual of an arrangement of chords, but the dual of every chord arrangement does not result in a valid mesh.

The whisker weaving algorithm uses the STC to construct hexahedral meshes that

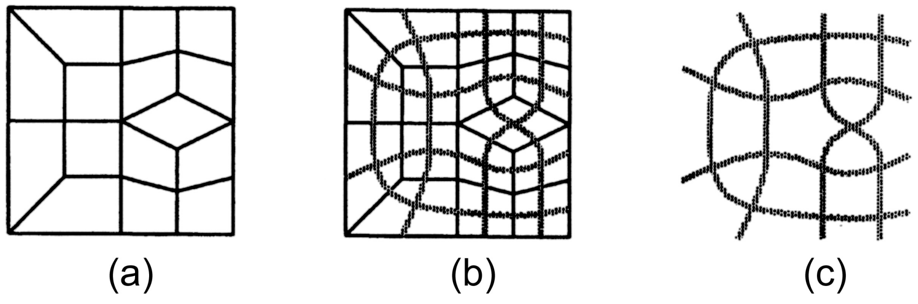


Figure 3.14: Quadrilateral mesh and the corresponding two-dimensional spatial twist continuum. Figure from [62].

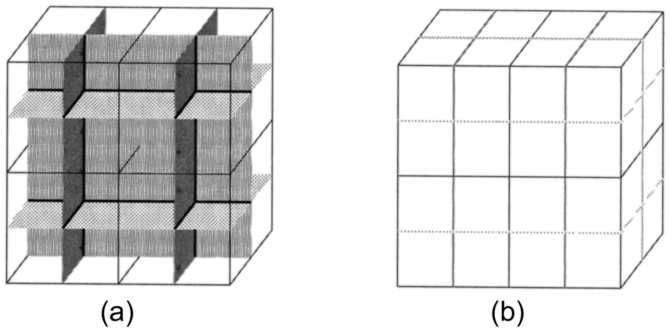


Figure 3.15: Spatial twist continuum for a mesh of four hexahedra and corresponding surface STC. Figure from [62].

are compatible with an arbitrary quadrilateral surface mesh of the object boundary. The intersection of a sheet of the STC with the object boundary is a closed three dimensional curve, or loop. The whisker weaving algorithm begins by constructing loops through the quadrilaterals of an object's surface mesh. Each loop can be contracted to a point and in the process sweep out a sheet of the STC. The whisker weaving algorithm contracts the surface loops to a point one at a time, until no loops remain. The end result is a set of STC sheets. This set of sheets is then checked and possibly modified to ensure that the dual of the STC defines a valid hexahedral mesh. If degeneracies exist, they are removed by either moving existing STC sheets or inserting a new sheet. The final step is to compute the dual of the STC and thereby create the hexahedral elements.

The whisker weaving algorithm has been shown to successfully create hexahedral meshes for a variety of geometries. Unfortunately, it frequently creates some elements

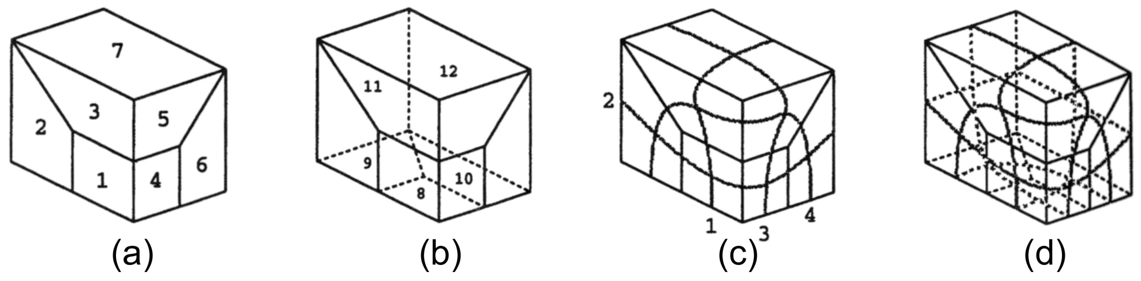


Figure 3.16: Spatial twist continuum of a surface mesh. Figure from [62].

with very poor shape. Large meshes created with the whisker weaving algorithm reportedly have 2% - 5% of the elements with a negative Jacobian. However, a literature search found that the only algorithms attempting to construct a hexahedral mesh compatible with a prescribed surface mesh for objects with arbitrary geometry were whisker weaving and plastering. As discussed previously, the plastering algorithm has only succeeded in creating hex-dominant meshes, so the whisker weaving algorithm is unique in its ability to match a surface mesh with an all-hex volume mesh. Even though element quality is not uniformly good, this algorithm represents a significant advancement in meshing research and remains an area of active investigation.

3.3.3 Summary of Meshing Algorithms

This section summarizes the salient features of each of the previously discussed meshing methods and briefly addresses the question of which type of meshing task is best suited for each of the meshing algorithm categories. The type of input a meshing algorithm requires can be an important factor in selecting an appropriate method for a problem.

- Mapped meshing (algebraic and elliptic methods) works for closed regions that have a boundary that can be mapped to a square or cube. Elliptic algorithms tend to give high quality meshes but can be computationally intensive.

- Hyperbolic methods can mesh a region that begins at a meshed surface but does not have a closed boundary. Hyperbolic meshes are frequently used when an overset meshing scheme is applied. Most applications for overset meshing are in the field of computational fluid dynamics (CFD). The mesh quality can be very good.

- Delaunay meshing is widely used due to the well known and robust algorithms for computing a Delaunay tesselation. It must be initialized with a distribution of input points. Some tetrahedra with poor shape are usually produced, and the mesh quality can be improved with post-processing operations.

- Advancing front mesh generation is also a widely used method. It must be initialized with a meshed boundary surface. Robust and efficient algorithms exist to produce tetrahedra, and post-processing may be used to improve mesh quality.

- Octree methods are initialized with a closed surface representation of an object and subdivide space in a hierarchic fashion. This spatial subdivision provides a structure for creating elements with different sizes. Typically octrees will have smaller subdivisions along the object boundary, which leads to the creation of smaller elements near the boundary. The method of element creation varies across octree methods, and mesh quality is highly dependent on the way elements are created.

- Grid based methods are automatic and fast, but the straightforward methods produce poor element shape at the boundary. Large, complicated grid based codes exist with heuristics that improve the shape of boundary elements. This method can be applied most easily if a relatively uniform element size is needed. Hexahedral meshes are produced, and a predefined surface representation of the object is required.

- Armstrong and Price's medial meshing algorithm uses global knowledge about an object's shape to decompose it into smaller more easily meshed chunks. It is a complicated algorithm, and current versions may not handle all possible geometries. However, it automatically produces hexahedral meshes in a principled way and is an area of ongoing research.

- Whisker weaving attempts one of the most difficult meshing problems: the automatic creation of a hexahedral mesh that conforms to an arbitrary quadrilateral surface mesh. It also is a complicated algorithm and currently does not handle objects with holes. Some of the elements it produces have very poor shape, but it is a promising area of active research.

3.4 Medical Image Based Meshing

This dissertation was motivated by the problem of building quality finite element meshes from three-dimensional medical images. Therefore an examination of the choices made by other researchers facing the task of three-dimensional image based meshing is instructive.

In all cases the objects of interest must be segmented from the image prior to meshing. Segmentation is a challenging problem separate from mesh generation, and the format of the segmentation results affects the choice of meshing algorithm. At a minimum, a segmentation yields a boundary description of the objects. A segmented object may be represented with a binary image, a stack of contours, a triangulated surface, a smooth spline or NURBS surface, a medial model, or another type of three-dimensional solid model.

Of particular interest is a study by Viceconti et al. [79] that compared the performance of four meshing algorithms applied to a CT of a femur. One tetrahedral meshing algorithm was studied along with three hexahedral meshing algorithms. A summary of the results is as follows.

- The tetrahedral meshing program required a polygonal boundary model as input and was able to generate a good quality mesh quickly.

- One of the hexahedral methods was a mapped meshing algorithm that required a significant amount of human interaction to define a set of meshable blocks that filled the femur volume. This algorithm required the most user time but also produced a mesh that yielded the smallest error.

- Another of the hexahedral algorithms was a voxel based method that simply turned each segmented voxel from the CT into a hexahedral element. A binary image rather than a geometric model of the femur was required for this method. This was a simple and fast algorithm that generated elements with good shape. However, the surface of the model was jagged, and it had to generate a significantly greater number of nodes in order to get error rates comparable to the other method. The increase in node count means that the solution stage for the voxel based mesh was computationally more expensive.

- Finally, an unstructured hexahedral meshing algorithm called HEXAR was tested. HEXAR required a polygonal model of the femur boundary. It used a structured interior grid and added unstructured elements to the surface of the interior mesh in order to capture the smooth object boundary. HEXAR produced a good result for the femur, but it required a significant amount of computational time on a Cray C90.

Of the meshing algorithms developed specifically for use with three-dimensional images, a literature search revealed examples only of tetrahedral meshing algorithms. Yao and Taylor [82] developed a tetrahedral algorithm that is capable of constructing a mesh directly from a stack of contours drawn on a CT image. Cebral and Lohner [16] developed an algorithm that begins with a three-dimensional binary image, triangulates the surface of the black/white boundary, and then uses an advancing front method to build a tetrahedral mesh.

As mentioned in section 3.2.1, hexahedral meshes offer some important computational advantages. However, tetrahedral meshing remains the most practical option for image based meshing because existing hexahedral meshing algorithms are either not sufficiently automatic, robust, capable of handling general geometries, fast, or widely available.

The m-rep based meshing algorithm presented in the following chapter is an approach that uses medial models to tackle both the problems of segmentation and automatic hexahedral mesh generation. This provides the benefits of a hexahedral mesh as well as the convenience of an integrated segmentation and meshing process. Since all image based meshing problems must address the issue of segmentation as well as mesh construction, this is a notable advantage.

Chapter 4

M-Rep Based Meshing Algorithm

The meshing algorithm presented in this chapter relies on m-rep object models for both global and local object shape information. An overview of m-rep models is provided, followed by an explanation of the meshing algorithm.

4.1 M-Reps

The discussion of m-reps begins with an introduction to the basic idea of medial modeling and then progresses to a detailed description of the structure of m-reps. Both single figure and more complicated multi-figure m-rep models are considered. Also included in this section is a review of the currently available methods of m-rep construction. An explanation of two alternative object based coordinate systems that are derived from m-rep object coordinates concludes this section.

4.1.1 Medial Model Concept

The medially based object representation introduced by Blum [10] consists of a medial surface and a radius function defined on the medial surface. The intuitive picture is of spheres with the largest possible radius fit inside an object so that they touch the object's boundaries on both sides of the object. The center point of each such sphere is a point on the medial surface, and the radius of each sphere defines the radius function at that point on the medial surface. The medial surface branches to represent object protuberances. See Fig. 4.1 for an illustration.

Since their introduction by Blum, medial models have proven useful for a variety of applications requiring shape modeling or shape decomposition. For example, medial

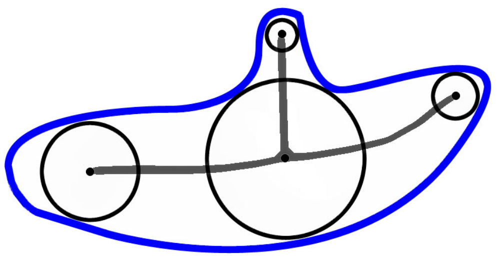


Figure 4.1: A 2D figure with a branched medial axis.

methods have been successfully applied to object recognition [19], image segmentation [56], and modeling of an object population that incorporates shape variability [69].

One of the main advantages of medial representations is that the branching structure of the medial surface provides global topology information about an object, as well as local boundary location information. However, Blum's original medial axis transform has been criticized because small protuberances and slight changes in object boundary shape can lead to changes in the branching structure of the medial axis. M-rep models provide an approach for managing this difficulty.

4.1.2 Anatomy of an M-Rep Figure

M-reps are medially based solid models that encapsulate information about interior object shape more directly than traditional boundary based models. They also possess a number of characteristics that make them particularly well suited for modeling anatomic objects. For the deformation modeling application, the object based coordinate system provided by m-reps facilitates both the construction of the finite element mesh and the efficient solution of the finite element system of equations.

In the general case, an m-rep consists of a hierarchical tree of figures, where each figure represents a main component of an object or a protrusion or indentation of one of the main components. The simplest m-rep consists of a single figure which represents a slab-like region with a non-branching medial locus. Single figure models will be described in this section, and multi-figure models will be addressed later in the chapter. Readers are referred to [55] for a more thorough explanation of m-reps.

A figure is composed of a medial sheet and several functions defined on the medial sheet. These functions include a scalar function that represents the object radius, a

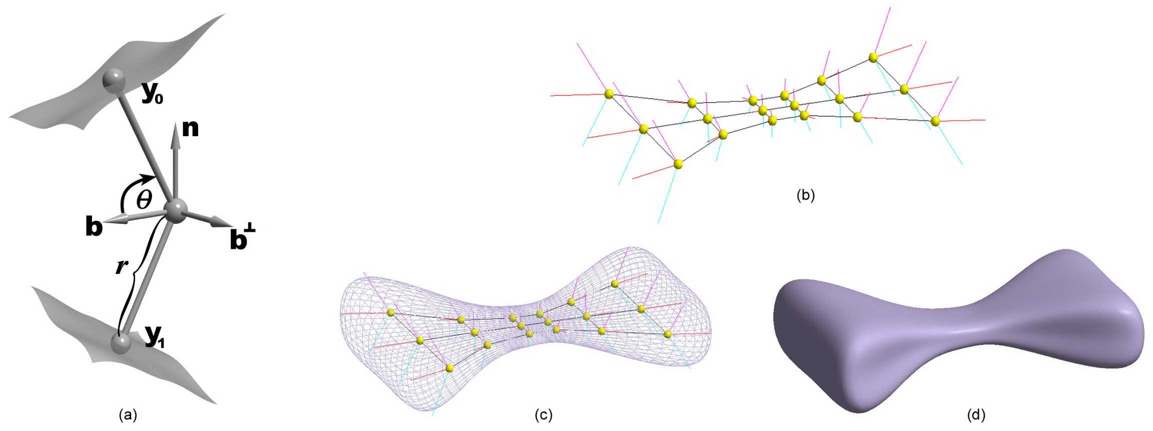


Figure 4.2: (a) Diagram of a medial atom (b) A single figure m-rep model composed of a lattice of medial atoms (c) M-rep with wireframe object surface rendering (d) M-rep with solid object surface rendering

function that represents the object boundary's vector displacement from the medial sheet along the medially implied normal, and frame and angle functions that provide orientation information. For implementation purposes the medial sheet and the functions defined on it are represented by discrete samples stored in a lattice data structure. Each of the discrete samples is referred to as a medial atom. The structure of a single atom is illustrated in Fig. 4.2(a). Medial atoms are the smallest building blocks of an m-rep, and each stores the following information.

- x, position coordinates of the medial sheet at a sample point
- r, the radius, which is defined as the distance from x to the object boundary
- two vectors which originate at x and point to the implied boundary locations that an inscribed sphere at x with radius r would touch.
- θ, the angle between $\vec{b}$ and a boundary vector
- $F = (\vec{n}, \vec{b}, \vec{b}^{\perp})$, a frame that defines the tangent plane of the medial sheet at x and the direction $\vec{b}$ on the tangent plane that is in the ∇r direction

Atoms along the outer edges of the lattice have an additional variable, the elongation e. The elongation term allows control of the curvature around the crest of the object.

The lattice arrangement of medial atoms helps define an object based coordinate system for m-reps. Any point in an object can be referenced by its m-rep defined (u, v, t, τ) coordinates. The u and v directions coincide with the rows and columns of medial atoms. u ranges from 1 to the number of rows of medial atoms in the lattice. v ranges from 1 to the number of columns of medial atoms in the lattice. Integer values of u and v coincide with the medial atom locations. τ ranges between 0 at the medial surface and ± 1 at the object surface, while t measures the angle between $+\theta$ and $-\theta$ in the crest region of the object. This object based coordinate system provides spatial and orientational correspondence between deformed versions of the same object. For example, in a prostate model a given set of (u, v, t, τ) coordinates will reference approximately the same part of the prostate using both compressed and uncompressed prostate m-rep models.

M-reps provide a smooth, continuous object boundary and medial surface by interpolating the medial atom data across the medial atom lattice. A subdivision surface algorithm presented in [71] is used to interpolate object surfaces in the Pablo m-rep modeling program. The approach taken by the m-rep meshing software is to acquire a sufficiently dense sample of surface points from this Pablo subdivision surface code and then simply apply a bi-cubic interpolation procedure to those sample points. Bi-cubic patches parameterized by (u, v) provide a simple interface for quickly evaluating the (x, y, z) coordinates of any (u, v, t, τ) point on an object's surface. This is important for optimization functions that are applied to object surface coordinates, such as the element shape and boundary condition optimization functions described later in this document.

The medial surface is also represented with bi-cubic patches that interpolate the medial atom locations. The transformation of (u, v, t, τ) to (x, y, z) for interior object points is accomplished by computing the corresponding (u, v) point on the medial sheet and the corresponding $(u, v, t, \pm 1)$ point on the object surface, and then linearly interpolating between the two points based on the value of τ. The function $object_{xyz}$, defined below in equation 4.1, maps (u, v, t, τ) m-rep based object coordinates to

(x, y, z) world coordinates.

$$\begin{aligned} object_{xyz}(u, v, t, \tau) &= medialSurf_{xyz}(u, v) * (1 - |\tau|) + objectSurf_{xyz}(u, v, t, sign(\tau)) * |\tau| \\ \text{where} \quad & medialSurf_{xyz} \text{ is a function that returns the world} \\ & \text{coordinates of a point on the medial sheet.} \\ & objectSurf_{xyz} \text{ is a function that returns the world} \\ & \text{coordinates of a point on the object surface.} \end{aligned} \tag{4.1}$$

The shape complexity of an m-rep is, of course, limited by its sampling rate. A dense lattice of medial atoms is capable of capturing more shape detail than a coarse lattice. The dimensions of an m-reps lattice of atoms is determined by the model builder and is based on an object's shape complexity as well as the level of detail required for a particular application.

M-reps deal with the difficulty of instability of the medial sheet's branching structure by defining a displacement function on an object's surface to account for small surface perturbations. This allows a simple medial sheet with few branches to represent gross object topology, and it allows slight indentations and protrusions to be represented by the surface displacement function. The function $objectSurf_{xyz}$ mentioned in equation 4.1 depends on this displacement function in the following way:

$$\begin{aligned} objectSurf_{xyz} &= boundaryVector(u, v, t, \tau) * radius(u, v) + boundaryDisplacement(u, v, t, \tau) \\ \text{where} \quad & boundaryVector() \text{ is a function that returns a unit vector} \\ & \text{that sits on the medial sheet and points in the direction of the} \\ & \text{object boundary} \\ & radius(u, v) \text{ is a function that returns a scalar value indicating} \\ & \text{the object radius} \\ & boundaryDisplacement() \text{ is a function that returns a vector} \\ & \text{of small magnitude that represents boundary surface details} \end{aligned} \tag{4.2}$$

Katz [33] discusses a method for determining which boundary protrusions should be modeled using a surface displacement function and which should be modeled as actual medial sheet branches. This strategy is advantageous because it both simplifies the data structure and allows an m-rep with a fixed topology to model an entire class of objects. For many object segmentation and object population modeling applications a single m-rep with variable atom data and variable surface displacements can account for whole class of objects.

One advantage of the object based coordinate system is that a mesh defined using an m-rep's object based coordinates is automatically individualized to fit any deformed version of the m-rep model. Since a deformed m-rep model maintains the same (u, v, t, τ) coordinate space as the original model, a mesh defined in that space can be transferred to a deformed model and mapped into world space using the deformed model's $(u, v, t, \tau) \rightarrow (x, y, z)$ mapping function. Such a mapping will transfer a mesh structure from one geometry to another but does not guarantee physical correspondence. The desire for physical correspondence motivates the application of the finite element method.

Another benefit of the object based coordinate system is the ability to express distances as a fraction of object width. This is convenient for mesh generation as it provides a natural way to size elements according to the proportions of an object.

In addition to the advantages offered by object based coordinates, m-reps are particularly attractive as part of an automatic meshing algorithm because they can be used to automatically segment objects from images [55]. In this work m-reps form a critical link between the image and the mesh generation process. Through the automatic segmentation procedure m-rep models are fit to image data, and then the geometry information stored in the m-rep is used to automatically generate a high quality finite element mesh.

4.1.3 Alternative Object Based Coordinate Systems

While an object based coordinate system offers distinct advantages for mesh construction, there are instances in which using four (u, v, t, τ) coordinates to reference a point in three dimensional space causes difficulties. Most notably, optimization algorithms are inefficient when forced to search a three dimensional space using four coordinates. The problem is further exacerbated when the domain of an optimization is the surface of a figure, since a surface is inherently two dimensional.

The solution to this problem involves two coordinate systems that are derived from the m-rep (u, v, t, τ) coordinates. The (A, B, C) coordinate system that spans the volume of an m-rep figure and the (a, b) coordinate system that spans the boundary surface of an m-rep figure will be referenced several places in this document. The structure of the coordinate systems is explained below, and the mapping functions that transform (u, v, t, τ) coordinates into (A, B, C) or (a, b) coordinates are defined

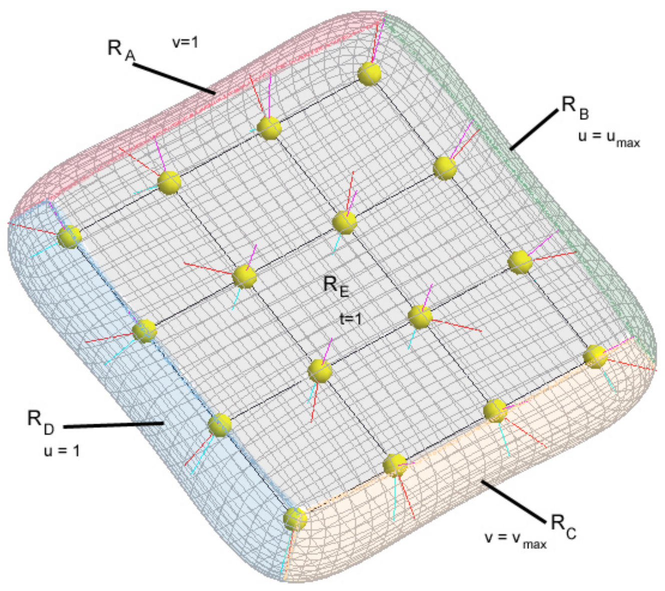


Figure 4.3: Five regions of an m-rep figure

later in this section.

An m-rep figure can be divided into five regions: a center region and four rim regions, labelled R_A, R_B, R_C, R_D and R_E in Fig. 4.3. The $(u, v, t, \tau) \rightarrow (A, B, C)$ coordinate mapping is based on the observation that for any one of the regions only three of the four medial coordinates are needed to uniquely identify a point. For the entire center region, R_E, all points can be referenced by $(u, v, 1, \tau)$; t does not need to vary. In each rim region, either u or v maintains a constant value. Therefore, the (u, v, t, τ) parameter space can be viewed as five three-dimensional parameter spaces rather than as one four-dimensional space. The (A, B, C) parameter space simply maps these five three-dimensional segments into one continuous three-dimensional space. For the center region, the mapping is direct.

$$A = u \qquad B = v \qquad C = \tau \tag{4.3}$$

For the rim regions, a more complicated mapping is necessary because the t and τ medial object coordinates for the rims are essentially polar coordinates, while the (A, B, C) coordinates form an object based Cartesian type coordinate system. See Fig. 4.4 for an illustration. To roughly fit this crest region with simply computed coordinates, the (A, B, C) coordinates for each rim span a wedge shaped region, as

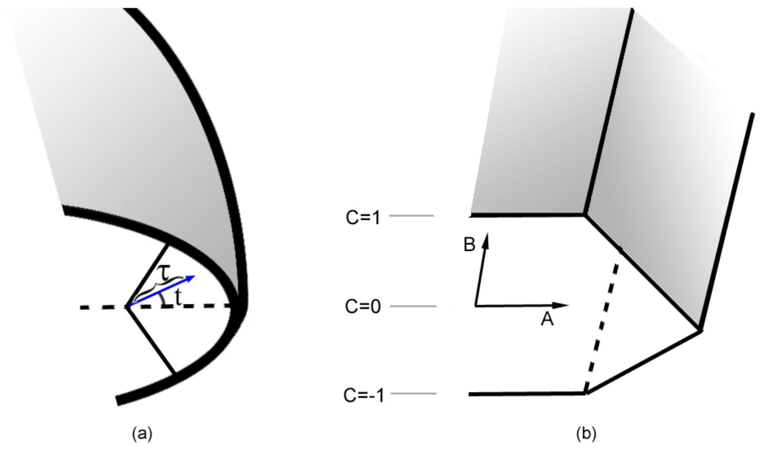


Figure 4.4: A sliced view of a rim region, with the polar type medial object coordinates shown on the left and the corresponding (A, B, C) parameter space on the right.

shown in Fig. 4.5. The equations of the $(u, v, t, \tau) \to (A, B, C)$ mapping for each rim region are as follows:

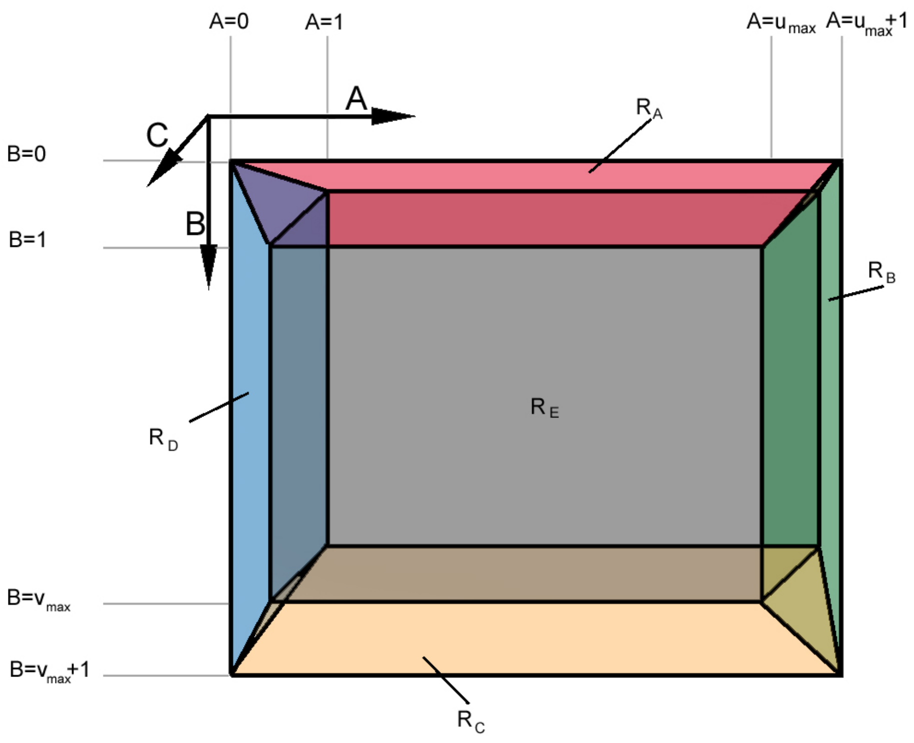


Figure 4.5: Diagram of the (A, B, C) parameter space with the five m-rep regions labelled

$R_A :$

$$\begin{aligned} A_{tmp} &= 1 - \frac{|\tau|}{\tan\left(\frac{t\pi}{2}\right) + 1} \\ A &= = v\left(A_{tmp}\right) + (1 - v)\left(\frac{A_{tmp} - 1}{u_{max} - 1}\right)(u_{max} + 1) \\ B &= v \\ C &= \tau\left(1 - \frac{1}{\tan\left(\frac{t\pi}{2}\right) + 1}\right) \end{aligned}$$

$R_B :$

$$\begin{aligned} A &= u \\ B_{tmp} &= v_{max} + \frac{|\tau|}{\tan\left(\frac{t\pi}{2}\right) + 1} \\ B &= (u_{max} + 1 - u)\left(B_{tmp}\right) + (u - u_{max})\left(\frac{B_{tmp} - 1}{v_{max} - 1}\right)(v_{max} + 1) \\ C &= \tau\left(1 - \frac{1}{\tan\left(\frac{t\pi}{2}\right) + 1}\right) \end{aligned}$$

$R_C :$

$$\begin{aligned} A_{tmp} &= u_{max} + \frac{|\tau|}{\tan\left(\frac{t\pi}{2}\right) + 1} \\ A &= (v_{max} + 1 - v)\left(A_{tmp}\right) + (v - v_{max})\left(\frac{A_{tmp} - 1}{u_{max} - 1}\right)(m + 1) \\ B &= v \\ C &= \tau\left(1 - \frac{1}{\tan\left(\frac{t\pi}{2}\right) + 1}\right) \end{aligned}$$

$R_D :$

$$\begin{aligned} A &= u \\ B_{tmp} &= 1 - \frac{|\tau|}{\tan\left(\frac{t\pi}{2}\right) + 1} \\ B &= u\left(B_{tmp}\right) + (1 - u)\left(\frac{B_{tmp} - 1}{v_{max} - 1}\right)(v_{max} + 1) \\ C &= \tau\left(1 - \frac{1}{\tan\left(\frac{t\pi}{2}\right) + 1}\right) \end{aligned}$$

(a, b) medial object surface coordinates are defined using the (A, B, C) coordinates described above. The (a, b) parameter space is basically the surface of the (A, B, C)

parameter space unfolded along the $(u_{max}+1)$ edge.

$$\begin{aligned} a &= \begin{cases} A & \text{if } C \geq 0 \\ 2\,(u_{max}+1) - A & \text{if } C < 0 \end{cases} \\ b &= B \end{aligned} \tag{4.4}$$

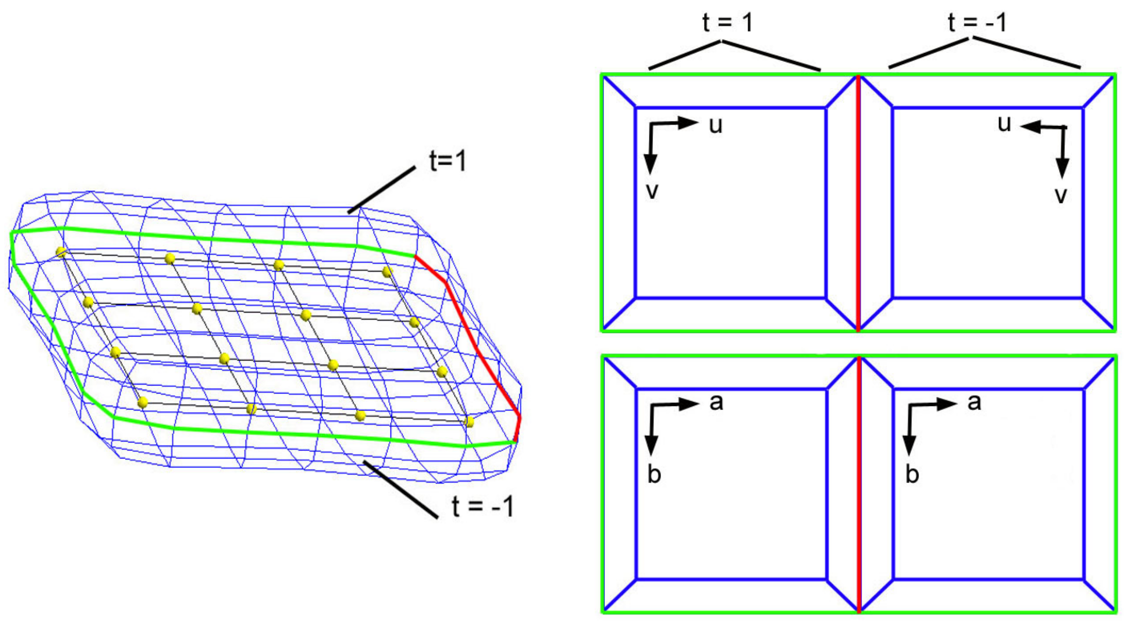


Figure 4.6: A figure is sliced along its crest on three sides (outlined in green) and remains hinged along its crest on one side (outlined in red). The flattened view of the figure's surface is shown on the right. In the diagram the (u, v) axis is mirrored about the hinge crest, but the (a, b) axis remains consistent.
The mapping from the curved surface in three dimensions to the two dimensional diagram necessarily stretches some parts of the surface. In particular, the crest region must either be stretched or cut when mapped to a two dimensional diagram. In this figure the crest region is stretched, so straight lines drawn through the crest region on the diagram do not necessarily translate into straight lines or geodesic lines on three dimensional surface.

4.1.4 Multi-Figure M-Reps

The (u, v) parameterization of the medial sheet described so far only covers a single, unbranched medial sheet. However, many objects of interest have branching medial sheets. M-reps represent branching through the use of sub-figures. Each continuous,

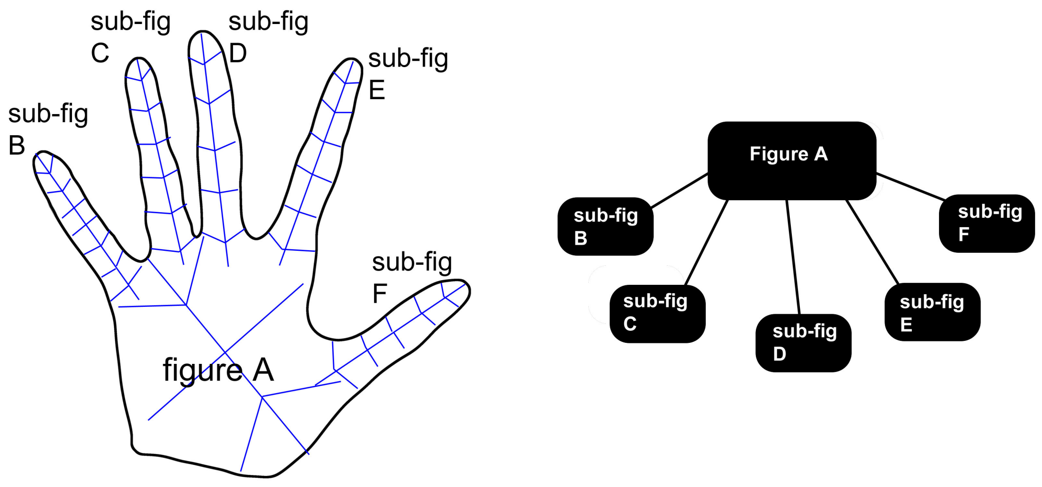


Figure 4.7: Left: 2D multi-figure m-rep of a hand Right: Hierarchical figure structure for hand m-rep

unbranching portion of a medial sheet is represented by a single figure lattice of medial atoms. Objects with a branching medial sheet have multiple figures, the union of which represent the entire object volume.

M-rep object topology is captured by a tree data structure that organizes the figures. For example, an m-rep hand model could consist of a single figure to represent the palm and five sub-figures to represent the fingers. Fig. 4.7 illustrates a possible arrangement of figures in two dimensions, along with the corresponding hierarchical figure data structure.

In a multi-figure model each figure has a separate (u, v, t, τ) coordinate system, and the coordinate systems overlap in the attachment regions. An attached sub-figure stores information about the location and orientation of its object based coordinate system in terms of its host figure's object coordinates. The result is that a multi-figure object is covered by a patchwork of object based figural coordinate systems.

In the m-rep model construction process decisions must be made about which object features should be represented with a sub-figure and how the figure hierarchy should be arranged. Katz and Pizer [33] have addressed this issue and have developed a method for deriving a stable m-rep figure hierarchy for complicated object shapes. This method analyzes an object's medial axis transform (MAT) and determines which parts of the medial axis primarily account for an object's substance and which parts primarily represent connections between different object regions. Only those sections of the medial axis that account for a significant amount of object substance need to be modeled as an m-rep figure or sub-figure.

To increase the versatility of sub-figures, blending parameters are available to govern the smoothness of the object surface in the neighborhood of a sub-figure attachment. For each sub-figure two blending parameters are defined. One parameter controls how far the blend region extends onto the surface of the subfigure, and the other controls how far the blend region extends onto the surface of the host figure.

4.1.5 Current M-Rep Software

There are three ways m-rep models can be generated:

- Build a new m-rep from scratch, using a modeling tool.
- Modify an existing m-rep to fit an object in a new image.
- Automatically generate an m-rep from volumetric data.

An m-rep model building tool named Pablo has been created that allows m-rep figures and sub-figures to be manually created and edited. Pablo allows figures to be created and deleted and provides editing access to the position, radius, and angle θ of individual medial atoms. This allows a user to construct a model for any object shape.

The Pablo program also has the capability to automatically fit an m-rep to image data. It uses a conjugate gradient algorithm to adjust the medial atom properties to maximize the boundary match between the boundary implied by the m-rep and the visible boundary of an imaged object.

Styner [69] presented a method for automatically building m-rep models from object surface data using pruned Voronoi skeletons. The primary focus of his work is the use of object shape statistics to build an m-rep model that is representative of not just one object but a population of objects. Such a model can be optimized to fit a particular instance of the object population.

4.2 Meshing M-Rep Objects

Some of the most promising methods of unstructured hexahedral mesh generation have made use of global information about an object's shape. For example, the medially based meshing algorithm presented by Armstrong [60] uses global shape information contained in the medial axis transform (MAT) to decompose an arbitrary object into meshable polyhedral subregions. Though this algorithm is quite complicated and computing the MAT is a challenging problem for some objects, this approach has enjoyed some success.

The whisker weaving algorithm builds hexahedral meshes for arbitrary objects using global shape information represented by the spatial twist continuum (STC). While it produces some elements with poor shape, it is a promising area of research. Examining the STC is enlightening because it highlights the reason that a hexahedral mesh design should be based on global shape information. Every hexahedral mesh has a STC dual, and valid modifications to a STC can only be performed by adding, deleting, or moving entire sheets of the STC. A sheet of the STC corresponds to a whole layer of hexahedral elements, and partial sheets cannot be permitted because they would represent an incompatible mesh with hanging nodes. Therefore, any modification to a hexahedral mesh is a global operation.

The m-rep based meshing algorithm described in the following section is simpler than whisker weaving and Armstrong's meshing method because it does not derive global shape information from a surface mesh or polygonal boundary model. Instead, it relies on the global and local shape information encapsulated in an m-rep model. M-reps are particularly well suited for medical image segmentation and registration applications [57] [55], and they carry the additional advantage of simplifying hexahedral mesh generation.

4.2.1 Single Figure Meshes

An explanation of the m-rep meshing algorithm begins naturally with the simplest m-rep model, a single figure. The m-rep based meshing algorithm uses a standard meshing pattern for each figure of a model and assigns object based coordinates to each node. The mapping from object based coordinates to world space coordinates determines the nodes' placement in world space.

The first step in meshing an m-rep figure is the construction of a sampling grid on the (u, v) parameter plane of the medial sheet. The vertices of the sampling grid are placed at regular intervals in (u, v) coordinates, and their world space coordinates are calculated by interpolating a position on the medial sheet. The spacing in the u and v directions is determined by the ratio of the average thickness of the object to the average world space distance between medial atoms. S_u, the number of medial

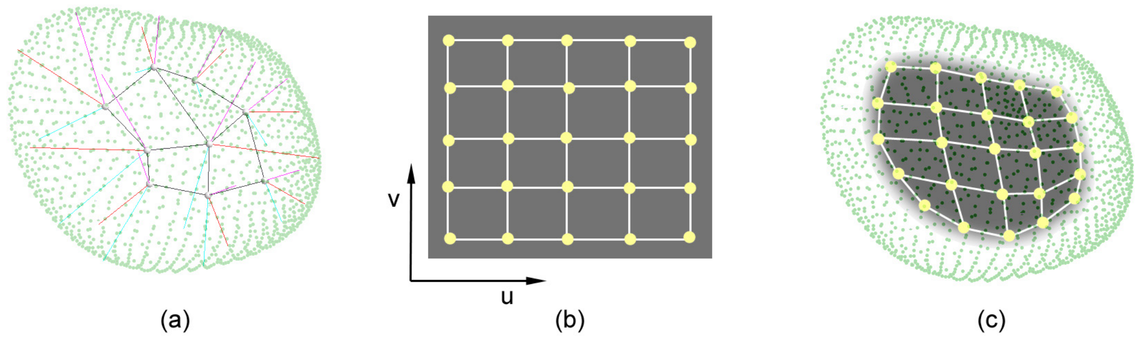


Figure 4.8: (a) M-rep constructed from 3x3 lattice of medial atoms. (b) The (u,v) parameter plane of the medial sheet with a 5x5 grid of sample points indicated. (c) Object with sample point interpolated and drawn on the medial sheet.

sheet samples in the u direction and S_v, the number of medial sheet samples in the v direction are defined as follows.

$$S_u = \frac{u_{max}}{(u_{max}-1)*v_{max}} * \sum_{i=1}^{u_{max}-1}\sum_{j=1}^{v_{max}} \frac{\|object_{xyz}(u,v,.5,1) - object_{xyz}(u+1,v,.5,1)\|}{\|object_{xyz}(u+.5,v,0,1) - object_{xyz}(u+.5,v,1,1)\|} \tag{4.5}$$

$$S_v = \frac{v_{max}}{u_{max}*(v_{max}-1)} * \sum_{i=1}^{u_{max}}\sum_{j=1}^{v_{max}-1} \frac{\|object_{xyz}(u,v,.5,1) - object_{xyz}(u,v+1,.5,1)\|}{\|object_{xyz}(u,v+.5,0,1) - object_{xyz}(u,v+.5,1,1)\|}$$

where $object_{xyz}$ is defined by equation 4.1.

Using the computed sample spacing, the average hexahedral element will have roughly equal dimensions. This process is illustrated in Fig. 4.8.

From the sampling grid on the medial sheet, the coordinates of the other layers of nodes can be derived. For every (u, v) sample point except those around the outer rim of the medial lattice, five nodes are created at $\tau = -1, -.5, 0, .5, 1$. For sample points around the rim, a slightly different set of six nodes is created, with the sixth node sitting out on the object crest. The object coordinates of the nodes are given in Table 4.1, and the node and element patterns are illustrated in Fig. 4.9.

The basic four layer figure meshing pattern shown in Figs. 4.9, 4.10, 4.11, and 4.12 provides a good coarse mesh of a figure and has the advantage of having no more

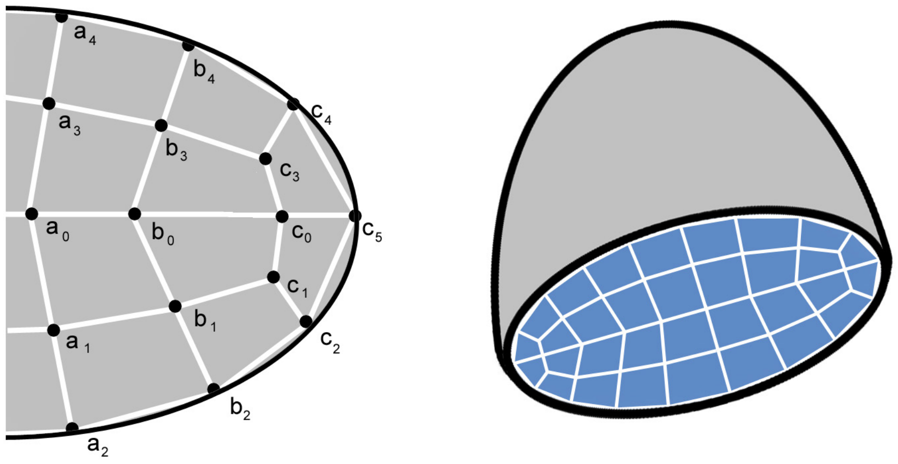


Figure 4.9: Left: Three groups of nodes constructed from three corresponding medial sheet samples are labelled. a_0 and b_0 are samples on the center portion of an object's medial sheet that give rise to nodes a_0 - a_4 and b_0 - b_4, respectively. c_0 is a sample on the outer rim of the medial sheet, from which nodes c_0 - c_5 are constructed.

Right: This is an interior view of the structure of the single figure meshing pattern. For a meshed single figure m-rep, any slice that is perpendicular to either the $\overrightarrow{u}$ or $\overrightarrow{v}$ directions on the medial sheet will reveal the same element structure as shown above.

The vector $\overrightarrow{(c_5 - c_0)}$ from each group of crest nodes (see group c on the left) is roughly perpendicular to the boundary of the medial sheet. This is true for the crest nodes at the corners of the medial sheet's (u, v) parameter space as well as for the crest nodes that lie along an edge of the parameter space.

than one face of any element lie on the surface of the object. This is important for the subdivision process described later. It also works well for objects with both high and low curvature crests.

Although the sample spacing is regular in medial coordinates, when the mesh is mapped into world space (x, y, z) coordinates the elements in narrower regions of the object tend to be smaller than the elements in wider areas. Typically, this is a desirable property, as a mesh usually needs to have smaller elements in narrower parts of an object in order to sufficiently model the detail. This behavior flows naturally from the use of the m-rep object coordinate system.

Because the mesh construction is guided entirely by information contained in the m-rep model, the meshing process requires no user interaction.

Table 4.1: Object coordinates of the mesh nodes shown in Fig. 4.9, where (u_a, v_a) and (u_c, v_c) are the coordinates of the medial sheet samples that generate the a_i and c_i nodes.

node label	u	v	t	τ
a_0	u_a	v_a	1	0
a_1	u_a	v_a	1	-.5
a_2	u_a	v_a	1	-1
a_3	u_a	v_a	1	.5
a_4	u_a	v_a	1	1
c_0	$u_c + .3$	v_c	1	0
c_1	u_c	v_c	1	-.5
c_2	u_c	v_c	1	-1
c_3	u_c	v_c	1	.5
c_4	u_c	v_c	1	1
c_5	u_c	v_c	0	1

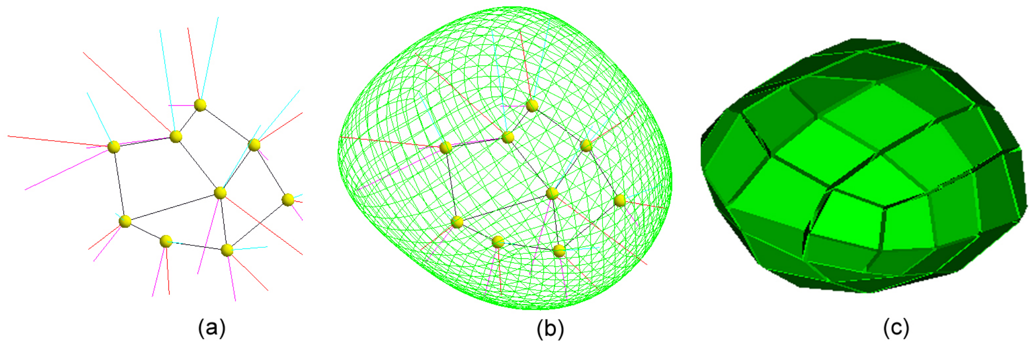


Figure 4.10: (a) M-rep model of a prostate (b) Prostate m-rep with implied surface (c) Base level prostate mesh

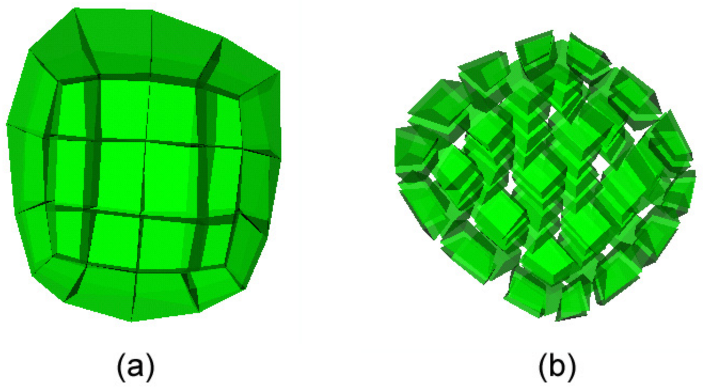


Figure 4.11: Two transparent views of the prostate mesh, intended to show the three dimensional structure of the mesh. (a) A view perpendicular to the prostate's medial sheet. (b) A view through the stacks of slightly shrunk elements.

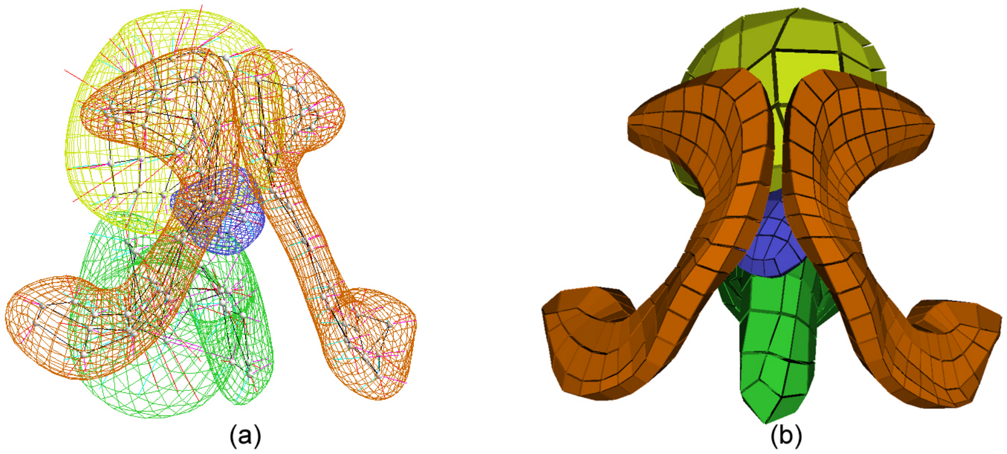


Figure 4.12: (a) M-rep model of male pelvis, including pubic bones, rectum, bladder, and prostate (b) Mesh of male pelvis objects

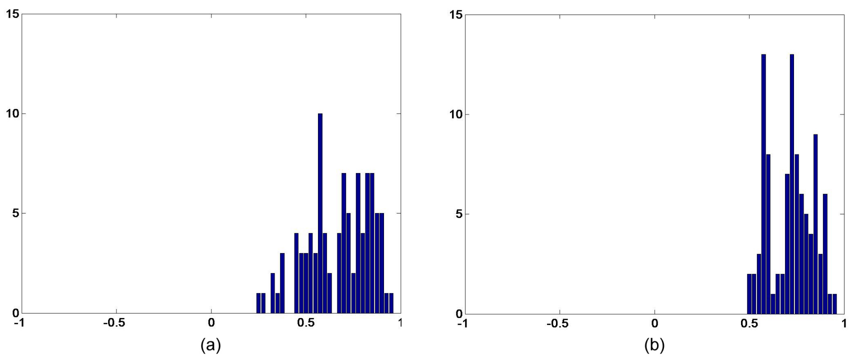


Figure 4.13: Histograms of det(J) for elements of the prostate mesh (a) before element quality optimization and (b) after element quality optimization

4.2.2 Mesh Quality Optimization

A three dimensional isoparametric element is defined in a parameter space (Fig. 3.2a) and is mapped into world space (Fig. 3.2b) via the element shape function $(x, y, z) = N(\xi, \eta, \zeta)$. The mapping function for an 8-node isoparametric linear hexahedral element is given in equation 3.2.

As explained in section 3.2.2, an important measure of hexahedral element quality is the determinant of the Jacobian of the mapping function. As gauged by the determinant of the Jacobian, the element quality of the majority of hexahedral elements generated by the m-rep based meshing algorithm is good, but some elements generated near the corners of the parameterized medial sheet or in areas of higher curvature can have poorer shape. Therefore, after mesh construction an optimization is performed to improve the shape of the less desirable elements.

The mesh quality improvement procedure first assigns to each element a score that is the minimum value of the determinant of the Jacobian of the element shape function evaluated at each of the Gauss integration points in the element. The node positions of all elements with a score less than .5 are optimized so that the det(J) values of the element and its neighbors are maximized.

Because the mesh is constructed with medial object (u, v, t, τ) coordinates but node positions do not actually have four degrees of freedom, coordinate transforms are applied in the optimization process. For nodes in the interior of a figure the parameters optimized are (A, B, C) coordinates, whose derivation from (u, v, t, τ) coordinates is explained in section 4.1.3. (A, B, C) coordinates are better for the

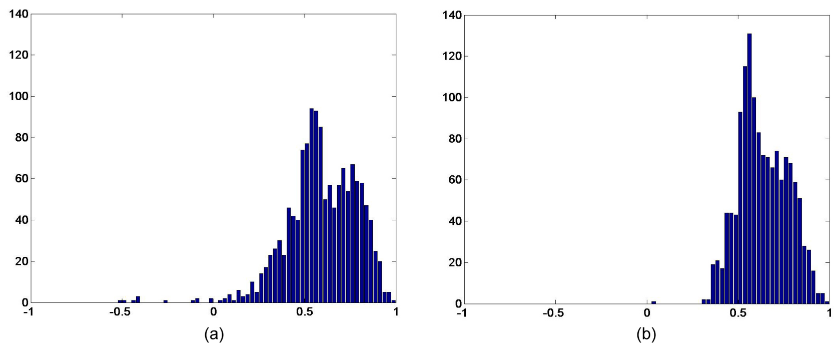


Figure 4.14: Histograms of det(J) for elements of the five-object male pelvis mesh (a) before element quality optimization and (b) after element quality optimization

optimization purpose than (x, y, z) world coordinates because they are closely based on the medial object coordinates and the mapping between (u, v, t, τ) and (A, B, C) is easily invertible, unlike the mapping between (u, v, t, τ) and (x, y, z). For nodes on the surface of a figure the parameters optimized are (a, b). The derivation of the (a, b) to (u, v, t, τ) coordinate mapping is also given in the appendix. Optimization in this two-dimensional parameter space implicitly constrains surface nodes to remain on the surface. The benefit of using these coordinate transformations in the optimization process is that the number of coordinate variables needed to specify node positions is reduced, thereby making the optimization process more robust and efficient.

This optimization process has successfully produced a mesh in which determinant of the Jacobian of all hexahedral elements is greater than .5 for the prostate model. This mesh is shown in Fig. 4.10, and histograms of element quality before and after optimization are in Fig. 4.13. For the five-object male pelvis model that includes the pubic bones, rectum, and bladder in addition to the prostate, all elements had a positive det(J) after optimization, and only a few elements had scores less than .5. The five-object mesh is shown in Fig. 4.12 and element quality histograms for the pelvis model are in Fig. 4.14.

4.2.3 Mesh Subdivision

If greater accuracy than provided by the initial coarse mesh is desired, the initial mesh can be subdivided. The result is a mesh with smaller elements that provides a finer representation of the solution. The subdivision algorithm involves adding a node to the midpoint of each existing edge in the mesh, adding a node to the center of each existing quadrilateral face, and adding a node to the center of each existing hexahedral element. Fig. 4.15 shows the subdivision pattern for the three element types.

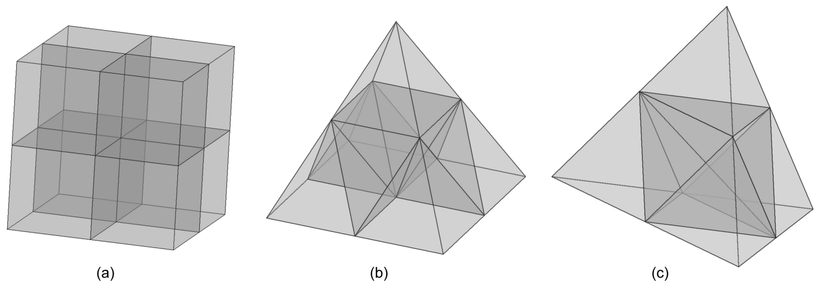


Figure 4.15: Element subdivision patterns for (a) hexahedra (b) pyramid (c) tetrahedra

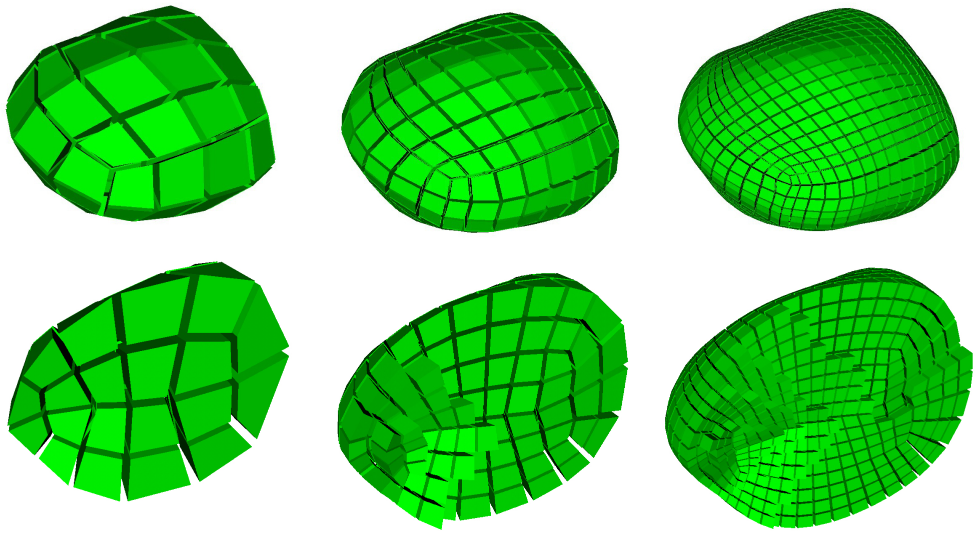


Figure 4.16: A prostate mesh at three scale levels.

The hexahedral elements that represent the m-rep modeled objects have nodes with both world space (x, y, z) coordinates and medial (u, v, t, τ) coordinates. By subdividing these elements using their medial node coordinates, an improved, smoother approximation to the object geometry is achieved with subdivision. The smoother boundary achieved with subdivision of the prostate mesh can be seen in Fig. 4.16. In contrast, straightforward subdivision with world space coordinates would also provide improved resolution for representing the solution but would not reduce the geometric error or blockiness of the mesh. The medial coordinate based subdivision process allows for increased precision in both the geometry and the solution.

If adjacent faces of an element lay on the object surface, the subdivision process described would lead to increasingly distorted and flattened elements since any surface patch is flat at a sufficiently small scale. The meshing pattern presented here has no elements with more than one face lying on the object surface, thus allowing good element shape to be maintained through an arbitrary number of mesh subdivisions.

4.2.4 Multi-Figure Meshes

Meshing multi-figure m-reps is significantly more complicated than meshing single figure m-reps due to the requirement that host figure and and subfigure meshes share a common interface at their intersection boundary. Satisfying this requirement results in continuous meshes for multi-figure objects, allowing the deformation of complicated objects to be simulated. The meshing algorithm previously described for single figure models must be modified to ensure mesh compatibility at host/subfigure intersection boundaries. Without taking steps to ensure compatibility in the attachment region, a separate, unconnected mesh would be generated for each figure, and deforming forces would not be transmitted across a sub-figure attachment boundary.

One of the issues that must be dealt with in the multi-figure algorithm is the varying size of elements between figures. It is possible for a subfigure to be significantly smaller than its host figure. Generated separately from its host figure, such a subfigure's mesh would naturally have smaller elements than the host figure's mesh. Connecting small elements to large elements clearly poses a difficulty. A possible solution would be to make the elements throughout a model have a uniform size. However, that solution typically does not provide an efficient allocation of computational resources. Element size is related to node density, and the number of nodes impacts the amount of memory required as well as the solution time required for a finite element problem. Therefore putting small elements in areas of a mesh that do not require small elements for accurate modeling wastes resources. Meshes that make efficient use of resources typically have elements whose size reflects an object's scale and level of geometric detail in the elements' neighborhoods.

An alternative strategy is to mesh the main body of a subfigure with elements

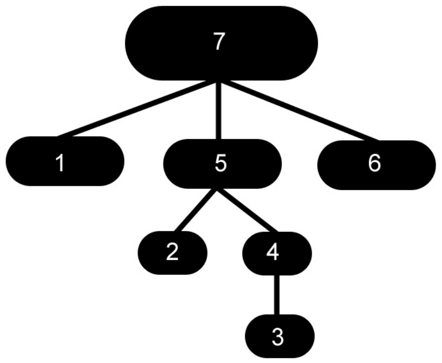


Figure 4.17: The tree nodes represent figures in an m-rep tree data structure. The numbers show the order in which the figures would be meshed.

that are appropriate for its size and shape, mesh the main body of a host figure with elements that are appropriate for its size and shape, and create a transitional mesh in the region where the two figures are joined. The transitional mesh elements can gradually change size and provide a smooth, blended connection between the main bodies of the host and subfigure meshes. This is the approach taken by the multi-figure meshing algorithm. Therefore the goal in the design of the multi-figure meshing algorithm was to create an algorithm derived from the single figure meshing algorithm that made only the minimum adjustments necessary to the single figure meshing pattern to guarantee host/subfigure mesh compatibility. This goal was based on the premise that the single figure meshing pattern is ideally fitted to a figure and that the size of the elements in a single figure mesh are appropriate for the scale of the figure. Therefore mesh quality will be best preserved by making minimal alterations to the single figure meshing pattern.

The multi-figure meshing algorithm described in this section manages the meshing of a sub-figure's attachment region. This algorithm is recursive, and it traverses an m-rep figure tree in depth-first fashion. Before a mesh is generated for a host figure, meshes are created for all of its subfigures. The order of meshing for a sample figure tree is shown in Fig. 4.17.

This approach has similarities to the block structured meshing methods mentioned in chapter 3. Block structured methods typically require users to manually decompose an object into meshable chunks and define a logical correspondence between each chunk and the unit cube. Then a Cartesian-type structured mesh is generated for each chunk and some heuristic is applied to guarantee mesh compatibility between chunks. With the multi-figure m-rep meshing algorithm, the decomposition into chunks is implicitly defined by the m-rep figure tree structure. The figure meshing pattern is not a Cartesian-type mesh but rather the meshing pattern described in section 4.2.1. To satisfy the compatibility requirement, the multi-figure meshing algorithm creates the attachment transition mesh that is described in detail in section 4.2.4.2.

4.2.4.1 Structure of multi-figure meshing algorithm

The structure of the recursive multi-figure meshing routine is as follows:

```
meshFigure(currentFig) {
    for i = 1 : currentFig.numSubfigures
        meshFigure(currentFig.subfigure(i))

    if currentFig.numSubfigures == 0 and currentFig.host == NULL
        meshSingleFigure(currentFig)
    else if currentFig.numSubfigures == 0
        meshLeafFigure(currentFig)
    else if currentFig.host == NULL
        meshRootFigure(currentFig)
    else
        meshGeneralFigure(currentFig)
}
```

The `meshSingleFigure` routine refers to the single figure meshing algorithm previously described in section 4.2.1. `meshLeafFigure` handles the case of a figure that is attached to a host figure but has no subfigures of its own. Transition region meshing is described in depth in the explanation of the `meshLeafFigure` algorithm in section 4.2.4.2. `meshRootFigure` handles the case of a figure that is not attached to a host figure but does have subfigures attached to it. The mechanism for getting the surface mesh of a host figure to be compatible with the footprint of a subfigure is explained along with the `meshRootFigure` algorithm in section 4.2.4.3. `meshGeneralFigure` handles the case of a figure that is both attached to a host figure and has one or more subfigures attached to it. This algorithm contains many of the steps from the `meshLeafFigure` and `meshRootFigure` routines and is covered in section 4.2.4.4. The way the multi-figure meshing algorithm treats a smooth blend region is discussed in section 4.2.4.5.

4.2.4.2 Leaf figure meshing

This section examines each of the steps in the `meshLeafFigure` routine. Meshing a leaf figure includes the task of creating a transition mesh that connects the main body of the leaf figure mesh with the surface of the host figure mesh. In the interest of clarity in the following discussion, some terms relating to the transition mesh are defined below.

- hinge atom row - the row of medial atoms along one edge of a subfigure's medial atom lattice that attaches to a host figure
- top of transition mesh - a slice through a subfigure that lies between the transition portion of the subfigure mesh and the main body of the subfigure (see Fig. 4.18(b))
- bottom of transition mesh - the portion of a host figure's surface mesh that corresponds to the footprint of a subfigure on the host figure's surface (see Fig. 4.18(a))
- sides of transition mesh - the surface of the transition mesh region, not including the portions already labelled as top and bottom

The constraints imposed on the transition mesh are as follows:

1. It must be composed of hexahedral elements in order to meet the goal of constructing an all-hex mesh.
2. Its top surface must exactly match the meshing pattern of a cross-section of the leaf figure's main body mesh.
3. Its bottom surface must exactly match the meshing of the subfigure's footprint on the host figure's surface.

In general, constructing a hexahedral mesh that conforms to a given quadrilateral surface mesh is a difficult task. In fact it has been proven impossible to create a hexahedral mesh for certain quadrilateral surface mesh patterns [46]. The creation of the host/subfigure transition mesh here is simplified by the fact that a surface mesh is specified only for the top and bottom surfaces of the transition region rather than for

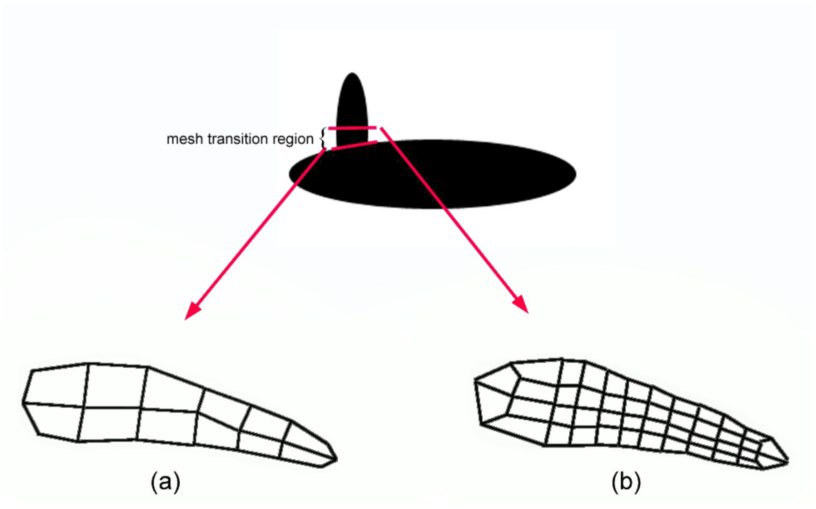


Figure 4.18: An example of transition region meshing patterns for (a) the Cartesian pattern found on the bottom of a transition mesh that corresponds to the portion of a host figure's surface mesh occupied by a subfigure's footprint (b) the m-rep generated pattern that that exists on the top of a transition mesh and corresponds to a cross-section of a subfigure's mesh.

its entire surface. There are no constraints applied to the surface mesh pattern of the sides of the transition region. The pattern on the top surface of the transition mesh will always be the m-rep generated kind of pattern shown in 4.18(b). The pattern on the bottom surface of the transition mesh will always be a Cartesian mesh of the kind shown in 4.18(a). The number of rows and columns in the patterns can vary, but the topology of the patterns will be as shown in Fig. 4.18. Because the topology of the transition region's top and bottom surface patterns is known, a set of templates can be generated that provide transitions between two types of patterns. Although the diagram shows a subfigure whose intersection with its host is roughly perpendicular, the template based transition algorithm works equally well if a subfigure meets its host in a smooth, blended curve. In such a case the intersection region of the subfigure would be wider than its main body, but the meshing pattern would not be different. The template based transition meshing process is described in the following leaf figure meshing steps.

Leaf step 1: Locate the host/subfigure intersection surface in both host figure and subfigure coordinates.

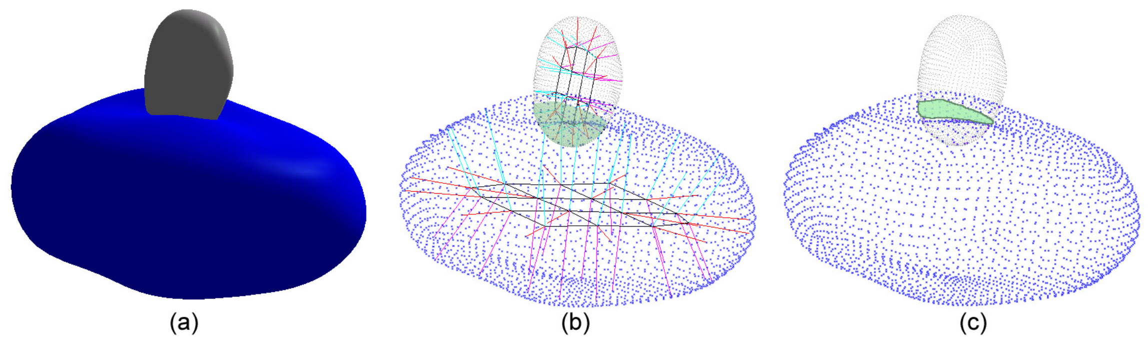


Figure 4.19: (a) A blue host figure has an attached gray subfigure. (b) A view of the two figure m-rep model shows the area of overlap between the host and subfigures in green. (c) The intersection patch that cuts through the subfigure and lies on the boundary of the host figure is shown in green.

Both the host and subfigure (u, v, t, τ) coordinate systems are defined in the area of overlap shown in Fig. 4.19. The perimeter of the intersection surface needs to be known in terms of host and subfigure coordinates; this is accomplished through a spatial search process. Points on the perimeter of the intersection surface satisfy the following equation:

$$\begin{aligned} \|subFigure_{xyz}(u_c, v_c, t_c, \tau_c) \quad &- \quad hostFigure_{xyz}(u_p, v_p, t_p, \tau_p)\| = 0 \qquad (4.6) \\ \text{where} \quad & (u_c, v_c, t_c, \tau_c) \text{ are subfigure coordinates} \\ & (u_p, v_p, t_p, \tau_p) \text{ are host figure coordinates} \\ & \tau_c = \pm 1 \\ & \tau_p = \pm 1 \\ & subFigure_{xyz}() \text{ and } hostFigure_{xyz}() \text{ are functions} \\ & \quad \text{that return } (x, y, z) \text{ world coordinates corresponding} \\ & \quad \text{corresponding to } (u, v, t, \tau) \text{ figure coordinates,} \\ & \quad \text{as defined in equation 4.1.} \end{aligned}$$

If a subfigure is attached along the $u = 0$ or $u = u_{max}$ edge of its coordinate system, a set of points on the perimeter of the intersection surface can be computed by finding the (u_p, v_p, t_p, τ_p) and (u_c, v_c, t_c, τ_c) points for which equation 4.6 is true. A dense sampling of these points is computed by choosing a set of fixed values for v_c and minimizing the left hand side of equation 4.6, allowing u_c, t_c, u_p, v_p, t_p to vary.

If a subfigure is attached along its $v = 0$ or $v = v_{max}$ side, a set of fixed values for u_c may be chosen and the left hand side of equation 4.6 minimized by allowing v_c, t_c, u_p, v_p, t_p to vary.

Starting coordinates for the minimization are provided by the m-rep hinge atoms. Hinge atoms are the row or column of medial atoms in a subfigure that are connected to the host figure. The m-rep data structure stores the positions of the hinge atoms in terms of host figure coordinates, so the hinge atom positions are known in both the host and subfigure coordinate systems. Because hinge atoms lie in the host/subfigure attachment region, their coordinates provide a good starting place in the search for points on the perimeter of the intersection surface.

Leaf step 2: For the intersection surface, compute (1) the surface mesh topology dimensions that best fit the ideal element size and meshing pattern for the host figure and (2) the surface mesh topology and dimensions that best fit the ideal element size and meshing pattern for the subfigure.

A host figure's intersection surface mesh pattern is shown in Fig. 4.18(a). As described in the later explanation of the root figure meshing algorithm, the host figure's mesh is constructed to ensure that the portions of its surface that coincide with the footprint of a subfigure have the topology of a Cartesian grid; the number of rows and columns in the grid will vary for different figure geometries. In the context of attachment region meshing, a row of elements runs parallel to the subfigure's medial sheet, and the extent of a row is referred to as the length of the intersection surface. A column of elements runs perpendicular to the subfigure's medial sheet, and the extent of a column is referred to as the width of the intersection surface. The length and width of the intersection surface and the desired element size for the host figure determines the number of rows and columns in the pattern.

$$\text{numRows} = \frac{\text{intersection surface width}}{\text{ideal element edge length}} \tag{4.7}$$

$$\text{numCols} = \frac{\text{intersection surface length}}{\text{ideal element edge length}} \tag{4.8}$$

While the attachment area mesh connects to the surface of the host figure, it attaches to a cross-section of the subfigure. This is reflected in the subfigure intersection mesh pattern shown in Fig. 4.18(b). The number of columns in a subfigure's mesh pattern is determined by the overall dimensions of the subfigure, as explained in section 4.2.1.

The attachment region mesh must somehow connect the differing host and subfigure intersection mesh patterns in order to form a continuous mesh. This is the task of the next step in the algorithm.

Leaf step 3: Create an attachment transition mesh that connects the host figure mesh and the subfigure mesh.

The construction of an attachment mesh is template based. It begins by taking the host mesh pattern as the base of the attachment transition mesh. Templates are then chosen to build up from the base to produce a mesh that matches the subfigure's mesh pattern at the top of the attachment transition mesh.

The attachment transition mesh is constructed in three phases. The first phase adjusts the number of rows in the pattern. The second phase adjusts the topology of the pattern, switching from a Cartesian mesh topology (Fig. 4.18(a)) to the topology consistent with a cross-section of a subfigure's mesh (Fig. 4.18(b)). The third and final phase adjusts the number of columns in the pattern. Examples of transition mesh cross-section patterns for each phase are shown in Fig. 4.20.

In the first phase templates are applied to produce a pattern with two rows of elements. The templates applied for this purpose are shown in Fig. 4.21. The first template shown converts a pattern with one element row to a pattern with two element rows. The remaining templates convert patterns with more than one row to a pattern with just two rows. These two dimensional templates are applied to the base pattern column by column so that in three dimensions the elements formed are extruded versions of the two dimensional patterns. This works properly because the base pattern is a Cartesian type grid, as shown in the upper left corner of Fig. 4.20.

The second phase of attachment mesh pattern construction starts with a pattern like the one shown in the upper right corner of Fig. 4.20. In this phase, a three dimensional template is applied to the ends of the mesh pattern to convert from the Cartesian grid topology of the host's pattern to the subfigure's mesh pattern (see Fig. 4.18). The template shown in Fig. 4.22(a) is applied to the elements in the first and last columns of the pattern, while the template shown in Fig. 4.22 is applied to the remaining elements column by column. The result is a mesh pattern that matches the subfigure's attachment pattern, except that the number of columns in the transition mesh may differ from the number of columns in the subfigure's attachment pattern.

The third phase adjusts the number of columns in the attachment transition mesh

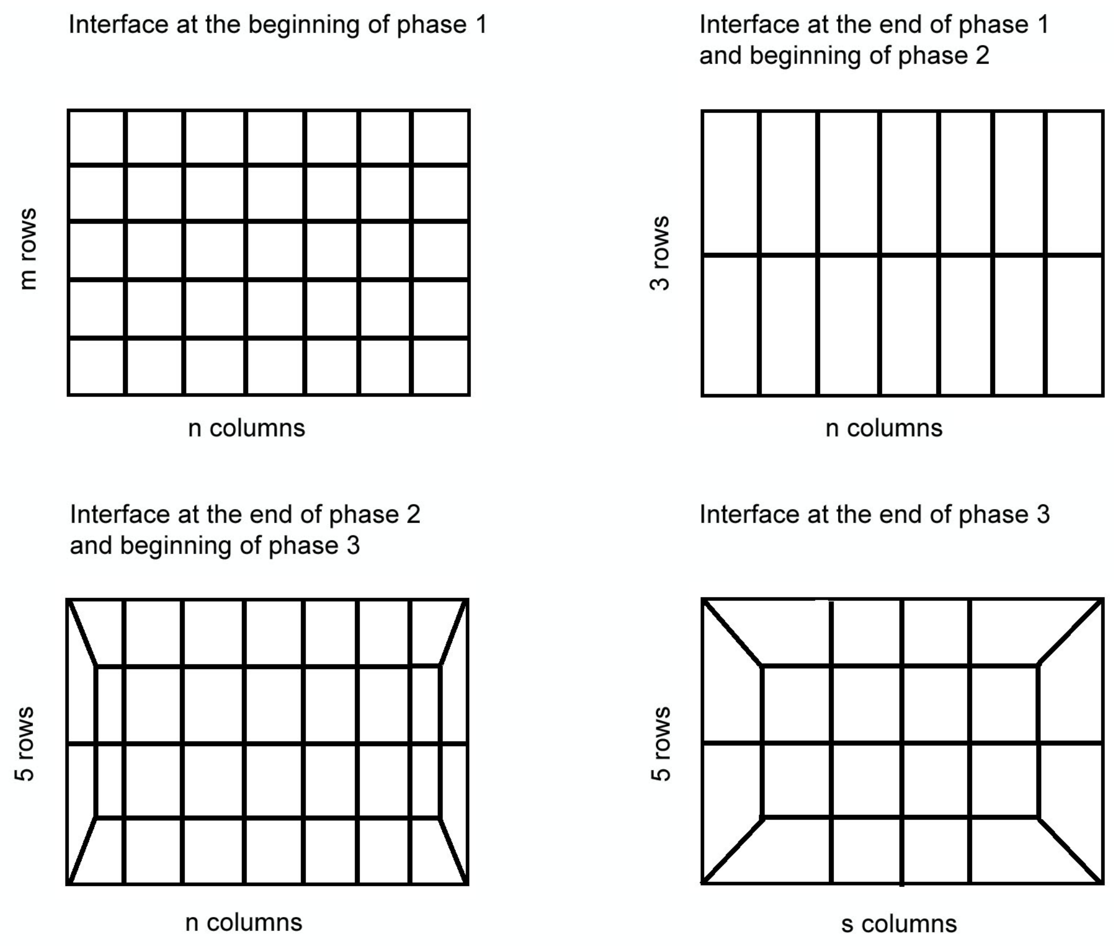

Interface at the beginning of phase 1
m rows
n columns
Interface at the end of phase 1 and beginning of phase 2
3 rows
n columns
Interface at the end of phase 2 and beginning of phase 3
5 rows
n columns
Interface at the end of phase 3
5 rows
s columns

Figure 4.20: Transition mesh interface patterns

The Cartesian type pattern on the surface of the host figure begins with m rows of mesh lines and n columns of mesh lines, as shown in the upper left corner. In phase one of the transition mesh the number of mesh lines is adjusted to 3, resulting in a $3 \times n$ Cartesian type pattern at the end of phase one. In phase two, three dimensional templates are applied to change the topology of the pattern. At the end of the second phase, the cross-section transition mesh pattern is similar to the one shown in the bottom left corner. In the third and final phase, the number of columns is adjusted to match the number of columns, s, in the subfigure mesh. This results in the kind of pattern shown in the bottom right corner.

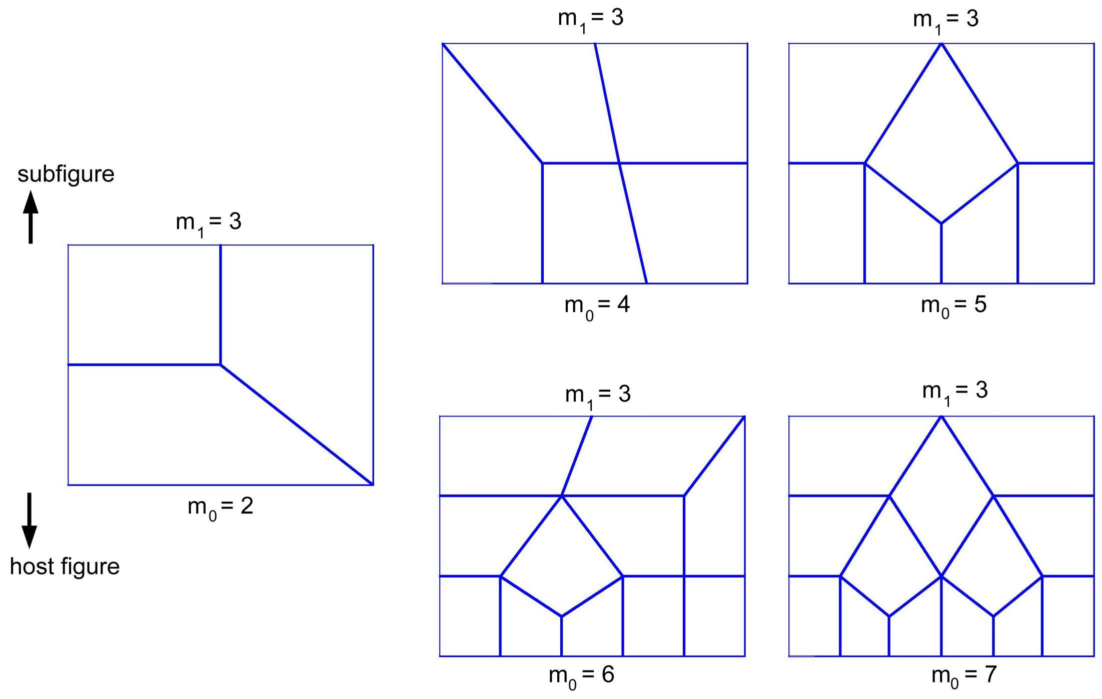


Figure 4.21: Templates for converting a mesh pattern with up to seven rows of mesh lines to a pattern with three rows of mesh lines.

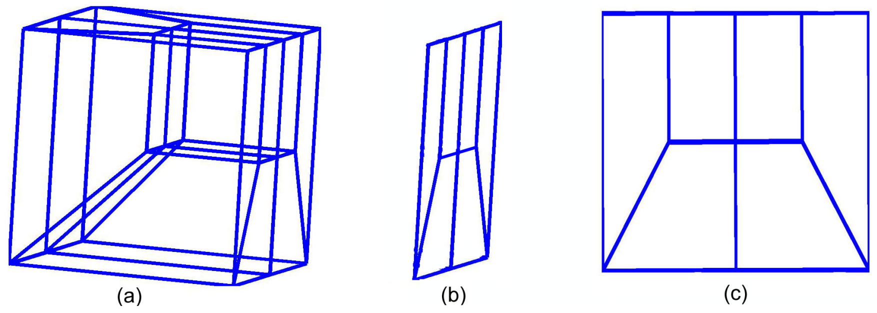


Figure 4.22: 3D templates for converting between a host figure's Cartesian grid type attachment pattern and the pattern associated with a cross-section of a subfigure's mesh. (a) The template applied to the columns at either end of a pattern in the second phase of a transition mesh (b) The template applied to the center columns of a pattern in the second phase of a transition mesh, shown at an angle so that its compatibility with pattern (*a*) is obvious. (c) The same pattern as shown in (*b*), drawn in the plane of the page.

so that it exactly matches the subfigure's attachment pattern. The number of columns may need to be either increased or decreased, and templates for both options are shown in Fig. 4.23 and and Fig. 4.24. A column adjustment template is applied across all four of the element rows. Different column templates may be applied to different columns, to achieve the desired final column count. An example of two different column templates used together to achieve a 10 $\rightarrow$ 4 column transition is shown in Fig. 4.25.

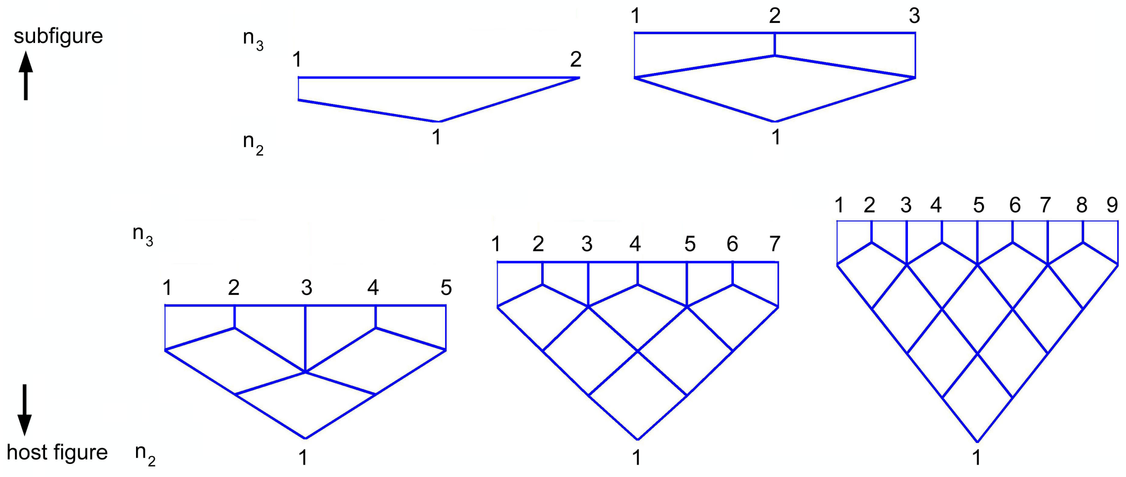


Figure 4.23: Templates that may be applied to increase the number of columns in an attachment transition mesh pattern.

The final result is a collection of templates put together in such a way that a continuous transition is provided between the host attachment pattern and the subfigure attachment pattern. An example of such a result is shown in Fig. 4.26, and a more complicated transition result is shown in Fig. 4.27.

The attachment transition pattern topology is computed in a cubical (i, j, k) parameter space. Afterwards the coordinates are scaled to fit into the subfigure's (A, B, C) coordinate space. The (A, B, C) coordinate system is explained in detail in Appendix A. Finally, the (A, B, C) coordinates are mapped to the subfigure (u, v, t, τ) figure coordinates and (x, y, z) world coordinates. A view of an attachment transition mesh in world space is provided in Fig. 4.28.

Leaf step 4: Mesh the remaining volume of the subfigure using the single figure meshing algorithm.

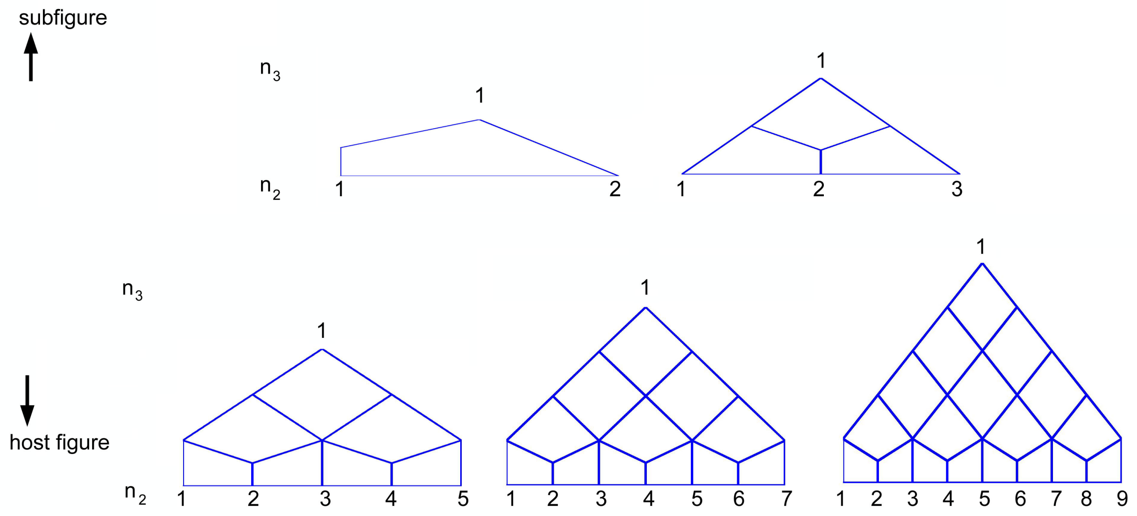


Figure 4.24: Templates that may be applied to decrease the number of columns in an attachment transition mesh pattern.

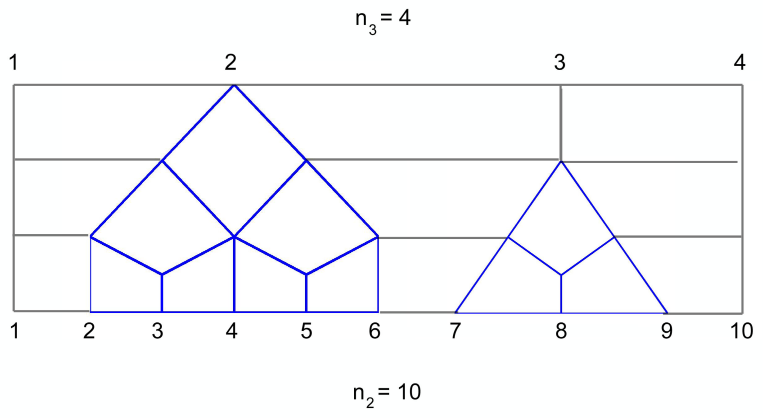


Figure 4.25: Sample column transition pattern to reduce the number of mesh line columns from 10 to 4.

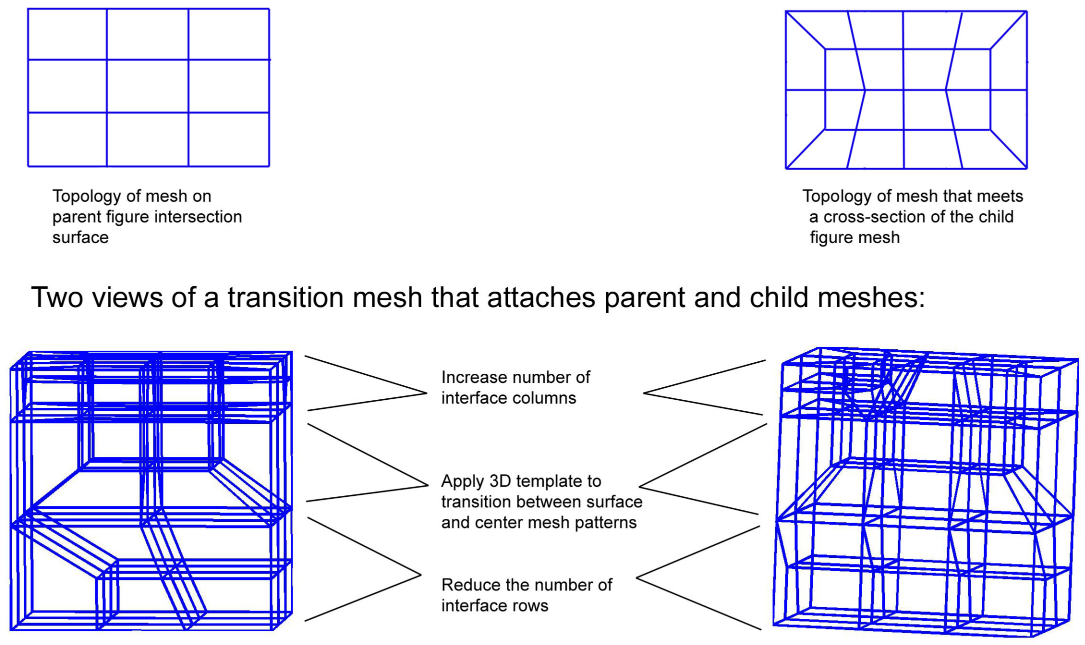

Topology of mesh on parent figure intersection surface
Topology of mesh that meets a cross-section of the child figure mesh
Two views of a transition mesh that attaches parent and child meshes:
Increase number of interface columns
Apply 3D template to transition between surface and center mesh patterns
Reduce the number of interface rows

Figure 4.26: An example of an attachment transition mesh. The lower pair of images shows the transition mesh from two different viewpoints.

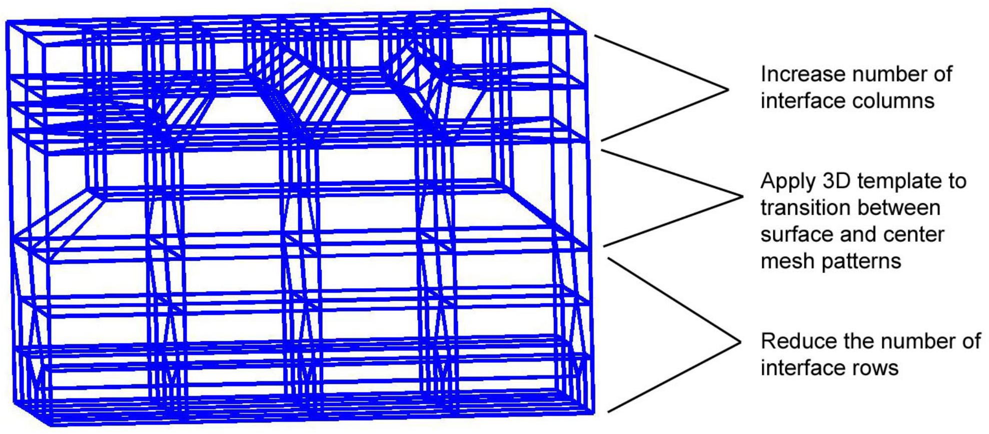

Increase number of interface columns
Apply 3D template to transition between surface and center mesh patterns
Reduce the number of interface rows

Figure 4.27: Another example of an attachment transition mesh.

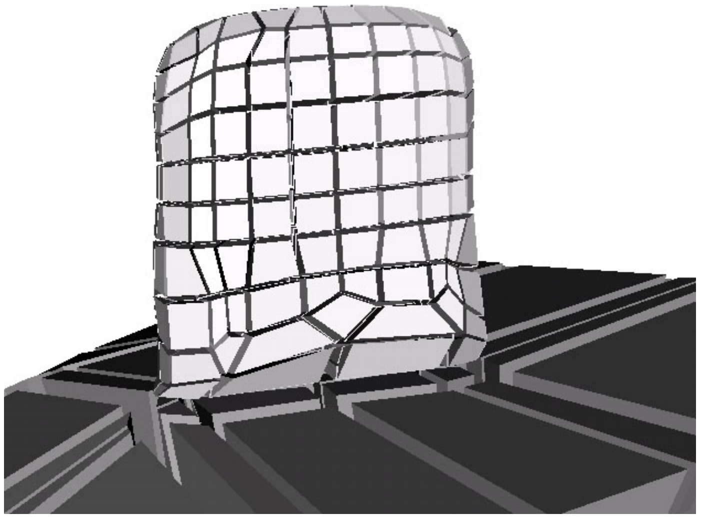


Figure 4.28: A view of a completed subfigure mesh.

The portion of the subfigure that lies above the top of the attachment transition is meshed using the single figure meshing algorithm described in section **??**. Connecting this portion of the mesh to the attachment transition mesh is straightforward since the top of the transition mesh matches the subfigure's mesh pattern.

4.2.4.3 Root figure meshing

The `meshRootFigure` routine creates meshes for figures that have one or more attached subfigures but are not themselves attached to a host figure. The basic strategy of the `meshRootFigure` algorithm is to create the meshing pattern that was described for single figures but do it in such a way that the mesh lines on the surface of the figure are aligned with the footprints of the attached subfigures.

The explanation of root figure meshing involves the careful examination of a root figure's surface mesh, so this section includes several diagrams of root figure surface meshes. These diagrams are of the (a, b) parameter space of the root figure's surface. They are drawn as if the root figure has been sliced open and flattened on the page. Fig. 4.6 illustrates how these two dimensional diagrams are related to the three dimensional figure.

The algorithm consists of the following steps:

Root step 1: Compile a set of surface mesh line segments.

Prior to meshing a root figure, all of its subfigures figures are meshed. Therefore, the footprints of all the subfigures figures on the surface of the root figure are known in terms of the root figure's coordinates system. For an explanation of how the footprints are computed, see leaf figure meshing step 1.

The first step of the root figure meshing process consists of simply collecting the footprints of all the root figure's subfigures and forming a list of the line segments that must be included in the root figure's surface mesh in order for the root to attach properly to its subfigures. Fig. 4.29 illustrates what the surface mesh of a root figure looks like at this preliminary stage.

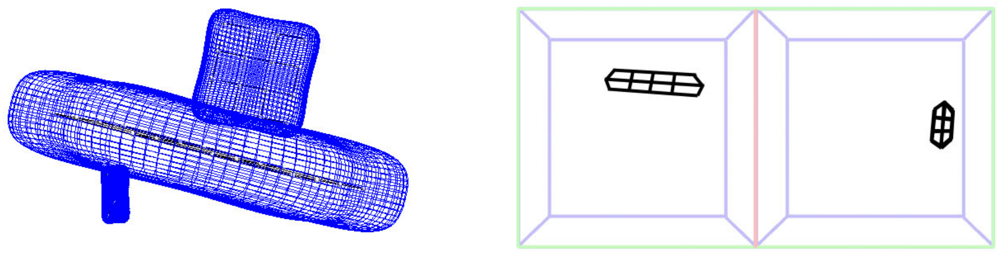


Figure 4.29: A root figure with two subfigures is shown on the left, and a diagram of the figure's surface with the footprints of its subfigures is shown on the right.

Root step 2: Extend and connect the surface mesh line segments to form continuous loops.

In the second step the surface mesh line segments are analyzed and categorized as either horizontal or vertical mesh edges, depending on the orientation of the line segments relative to the root figure's $\overrightarrow{u}$ and $\overrightarrow{v}$ directions. Next the line segments are sorted spatially and are assigned numbers according to ascending u coordinates for vertical lines and ascending v coordinates for horizontal lines. Line segments sharing the same orientation and number are then connected. All of the lines are extended and connected so that continuous loop lines are formed on the root figure's surface. Fig. 4.30 illustrates this process for the root figure shown in Fig. 4.29.

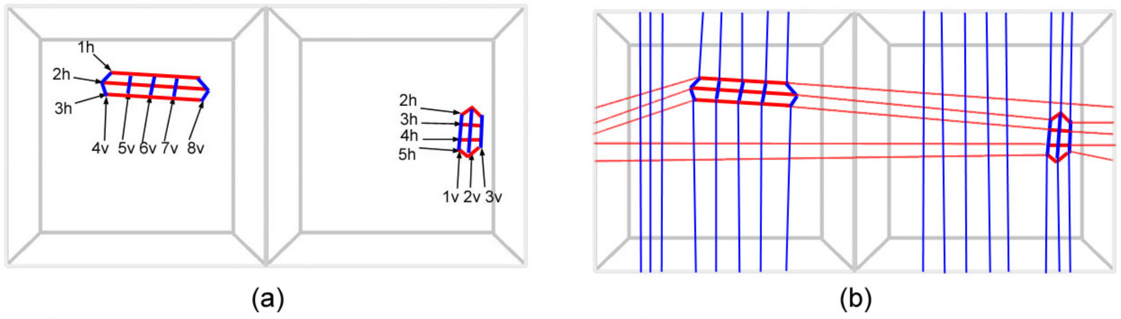


Figure 4.30: (a) The line segments in the subfigures' footprints are categorized as horizontal (h) or vertical (v) and are numbered. (b) The line segments are extended and connected to form continuous loops.

Root step 3: Add more surface mesh lines to fill in surface mesh gaps.

The average size of the gaps between neighboring surface mesh lines is computed and compared to the root figure's ideal element edge length. If a gap's average size is more than one and a half times the ideal edge length, one or more new mesh lines is generated to fill the gap. New mesh lines are generated by interpolating between two existing mesh lines or by interpolating between an existing mesh line and one of the root figure's crest edges. Fig. 4.31 provides a diagram of a root figure surface after additional mesh lines have been filled in.

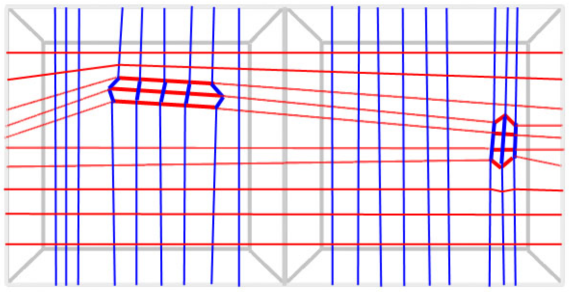


Figure 4.31: From Fig. 4.30, additional mesh lines have been interpolated to fill large gaps in the surface mesh.

Root step 4: Build hexahedral volume elements; produce interior nodes by interpolating coordinates between opposite sides of the surface mesh.

The topology of a hexahedral mesh built for a root figure is the same as the topology of the single figure meshes discussed in section 4.2.1. The difference lies in the way node coordinates are generated. For a single figure mesh, the node coordinates are generated based on a sampling grid defined on the medial sheet. For a root figure mesh, the node coordinates are generated by interpolating between mesh lines that lie on a figure's boundary surface. The difference is between propagating a mesh structure from the medial sheet outward and propagating a mesh structure from the boundary surface inward.

For root figures the portion of the surface mesh that lies on the t=1 side of the surface has the same structure and same number of rows and columns as the portion of the surface mesh that lies on the t=-1 side. However the curvature and shape of the mesh lines may be different on each side of the object in order to accommodate the footprints of subfigures on the figure's surface. This is evident by examining Fig. 4.32.

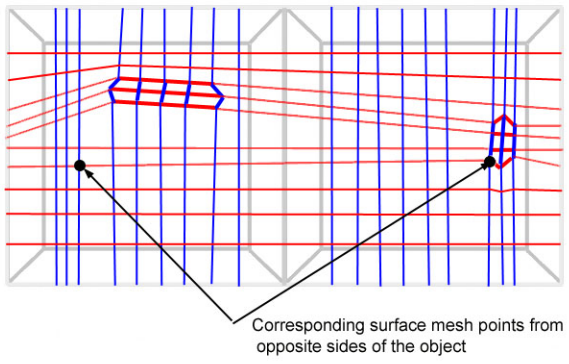

Corresponding surface mesh points from opposite sides of the object

Figure 4.32: This root figure surface diagram highlights matched points on opposite sides of the figure surface.

The interpolation of node coordinates is performed in the (A,B,C) coordinate system described in appendix A. This coordinate system is used instead of (u, v, t, τ) coordinates because straightforward linear interpolation with (A,B,C) coordinates produces the expected intermediate points, whereas interpolation with (u, v, t, τ) coordinates is complicated by the inclusion of the angle parameter, t. Also, if interpolating with (u, v, t, τ) coordinates the crest region must be handled as a special case.

Fig. 4.32 shows on a root surface diagram how surface mesh points have a matching point on the other side of the figure. Mesh nodes are interpolated between such

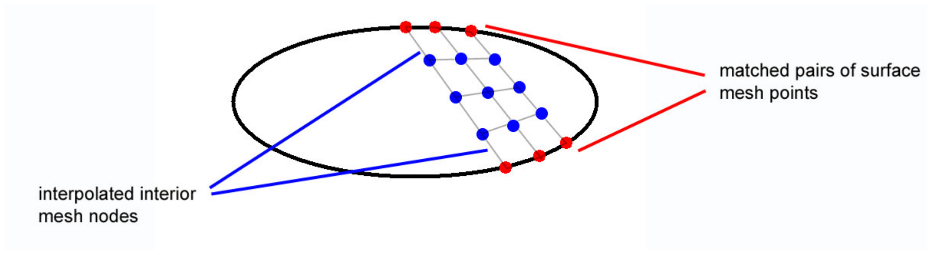

matched pairs of surface mesh points
interpolated interior mesh nodes

Figure 4.33: Interior mesh nodes are interpolated between matched pairs of surface mesh points.

matched surface points. Fig. 4.33 illustrates how interior nodes are created through this interpolation process. Fig. 4.34 shows an example of the result of the root figure meshing process.

4.2.4.4 General figure meshing

In the most general case a figure will have both a host figure and subfigures figures. The `meshGeneralFigure` routine applies to this case, and the steps of the algorithm are outlined below. All but one of these steps are the same as a step from either the root or leaf meshing routines. To avoid duplicating previous explanations, the steps that are shared with one of the other routines reference the appropriate previous section.

General step 1: Compile a set of surface mesh line segments. (See Root step 1.)

General step 2: Extend and connect the surface mesh line segments to form continuous loops. (See Root step 2.)

General step 3: Add more surface mesh lines to fill in surface mesh gaps. (See Root step 3.)

General step 4: Locate the host/subfigure intersection surface in both host figure and subfigure coordinates. (See Leaf step 1.)

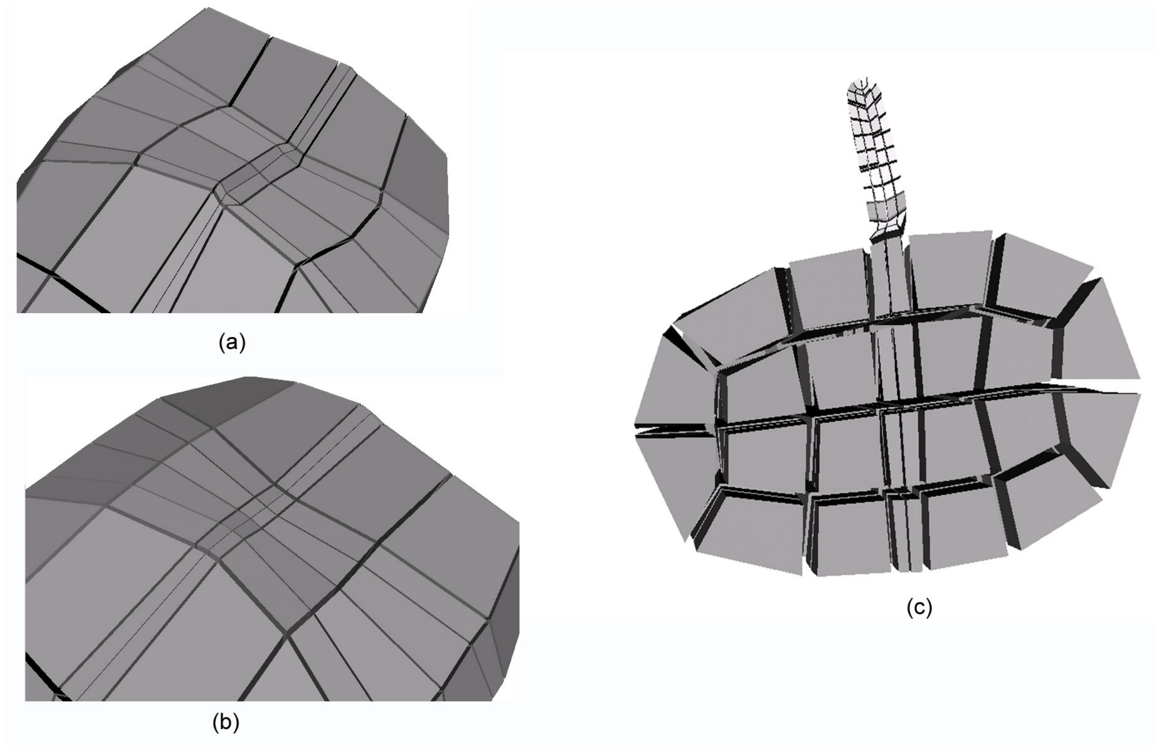


Figure 4.34: On the left side, a view of each side of the surface of a meshed root figure is shown, with the subfigure invisible. The topology of the mesh on either side of the surface is the same, but the shape of the surface mesh lines differ to account for the footprint of a subfigure on one of the sides. On the right side, a cutaway view of the same root figure is shown, this time with the subfigure visible.

General step 5: For the intersection surface, compute (1) the surface mesh topology that best fits the ideal element size and meshing pattern for the host figure and (2) the surface mesh topology that best fits the ideal element size and meshing pattern for the subfigure. (See Leaf step 2.)

General step 6: Create an attachment transition mesh that connects the host figure mesh and the subfigure mesh. (See Leaf step 3.)

General step 7: If necessary, compress the attachment transition mesh so it does not overlap any subfigure attachment surfaces.

When a subfigure has subfigures of its own, care must be taken to ensure that the attachment transition meshes do not interfere with each other. This meshing algorithm does not handle the case of an attachment transition mesh attaching to another transition mesh. Since the surface of an attachment transition mesh does not have the same Cartesian grid type topology as the rest of the mesh, the template based transition mesh algorithm would not necessarily be able to design an acceptable transition pattern. Therefore, a figure's own transition mesh is fit below the the attachment of any subfigures.

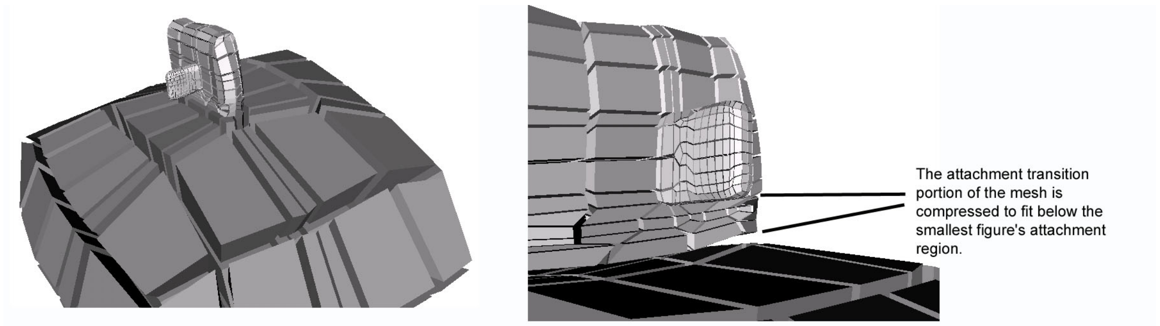

The attachment transition portion of the mesh is compressed to fit below the smallest figure's attachment region.

Figure 4.35: This mesh of a three figure m-rep object shows how the attachment transition portion of the mid-sized figure is compressed to fit below where the smallest figure attaches.

General step 8: Build hexahedral volume elements; produce interior nodes by interpolating coordinates between opposite sides of the surface mesh. (See Root step 4.)

4.2.4.5 Blend Region Meshing

Blending parameters may be specified for a multi-figure m-rep model to control the curvature of the object surface in the vicinity of a host/subfigure attachment. The manner in which surfaces are blended is more a modeling problem than a meshing problem, so the approach taken here is to modify the model from which the mesh is built to accommodate blending rather than modify the meshing algorithm. Within the meshing program, the object surface is represented as a collection of bicubic spline patches. When blending is applied, the spline surfaces of both the host and subfigures are adjusted to conform to the blended surface (see Fig. 4.36). The way this adjustment is performed is not critical to the meshing algorithm; examples have been generated by fitting the m-rep model to an image of a two figure object with a smooth transition between the figures. In these cases the blend region curvature was captured by the m-reps' surface displacement function described in section 4.1.2. With a properly adjusted spline surface, the meshing process proceeds normally and produces a mesh that conforms to the blended object surface. This approach is very general and would work with any blending method.

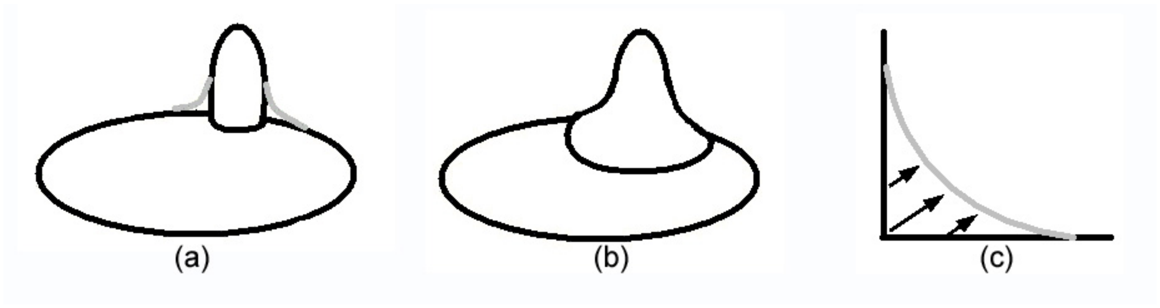


Figure 4.36: (a) An unblended two figure object, with gray curves showing how blending would change the object profile (b) A blended two figure object (c) The arrows show how figures' spline surfaces are adjusted in the attachment region to account for blending.

4.2.4.6 Limitations of multi-figure meshing

The multi-figure meshing algorithm works well for many object shapes and configurations, but there are some figure arrangements which cause difficulties for this algorithm. First, the multi-figure meshing algorithm may not produce good meshes for objects that have two branch points very close together. This is a difficulty because

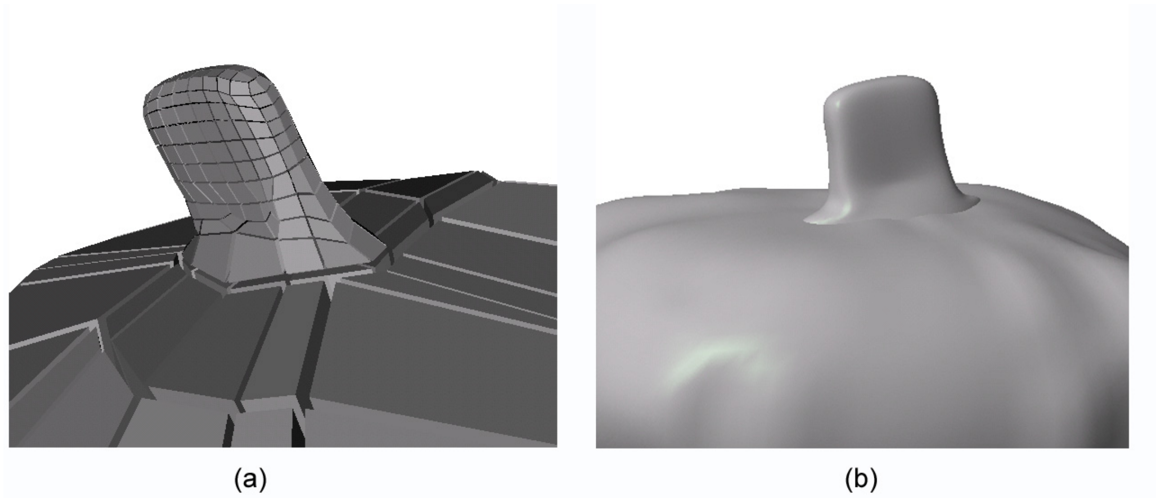


Figure 4.37: (a) A base level mesh of a blended two figure object. (b) A solid surface view of the same object.

the attachment transition portion of the mesh that connects host and subfigures must take up some space. If two branches occur close together, the attachment transition portion of the mesh may be compressed (see step 7 of the general figure meshing algorithm) to such a degree that mesh element quality is seriously degraded. In some cases, the mesh element quality optimization procedure can remedy this after the mesh is generated. In other cases, skewed or flattened elements may be unavoidable.

Second, the meshing pattern on the surface of a figure resembles a Cartesian grid over most of its surface. Only at the four corners of the (u, v) parametrization of the medial sheet does the surface mesh pattern have irregularities (see Fig. 4.38.) The attachment transition templates are all designed to attach to a host surface mesh with Cartesian grid topology. This means that using the existing templates a subfigure cannot be attached to the corner of a host figure. This limitation might be removed by either creating a more extensive template library or by allowing the corner points to be shifted on the surface of a host figure to avoid interfering with a subfigure attachment.

4.3 Meshing Space External to M-reps

In a multi-object deformation problem the space between the objects also needs to be meshed. In a single object situation it is often useful to mesh not just the object

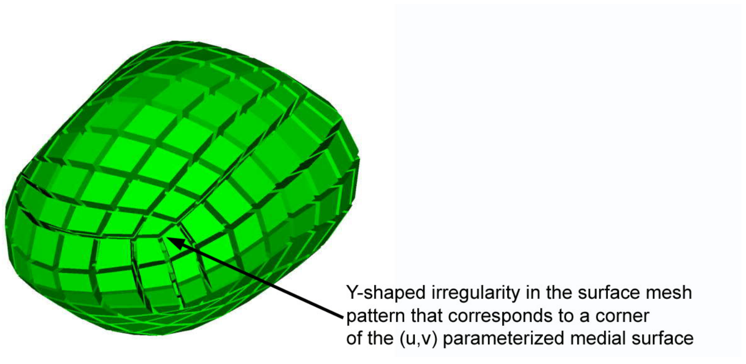

Y-shaped irregularity in the surface mesh
pattern that corresponds to a corner
of the (u,v) parameterized medial surface

Figure 4.38: Surface mash effect of the medial sheet's (u,v) parameter space corner.

of interest but also a region surrounding the modeled object. A mesh of the space exterior to modeled objects is necessary in order to compute a continuous deformation field for an entire region of interest. Such a mesh also allows deforming forces to be transmitted between different objects in a scene.

The m-rep meshing algorithm does not address the external space meshing problem because of the difficulty of creating m-rep models for the arbitrarily shaped space that lies between modeled objects. Instead, a tetrahedral mesh is created for the space between objects and the remainder of the volume of interest using CUBIT [1]. To provide a transition between the hexahedral elements of the m-rep models and the tetrahedral elements of the ambient space, a layer of pyramid elements is built on top of the exposed quadrilateral faces of the hexahedral elements.

The construction of the pyramids is straightforward. Each surface quadrilateral is taken as the base of a pyramid. The apex node is placed along the surface normal passing through the center of the quadrilateral base. The initial height of the pyramid is the average of the base edge lengths. When all the pyramids have been constructed, intersection tests are performed, and the height of any intersecting pyramid is reduced so that intersections are eliminated.

The construction of the tetrahedral mesh is performed using the tetrahedral meshing capability found in CUBIT [1]. The pyramid and tetrahedral elements that fill the space external to the models are shown in Figs. 4.39 and 4.40.

Subdivision of the pyramid and tetrahedral elements outside the m-rep modeled object volumes is performed using the nodes' world space (x,y,z) coordinates. See Fig. 4.15 for an illustration of the subdivision patterns.

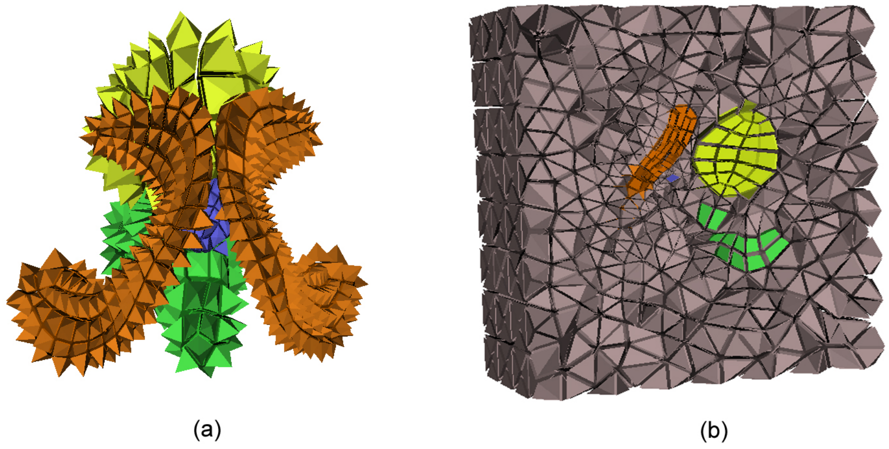


Figure 4.39: (a) Mesh of the male pelvis model with the transitional pyramid layer visible on top of the hexahedral elements (b) A sliced view of the meshed volume of the male pelvis model, including the pyramid and tetrahedral elements that fill the space between modeled objects.

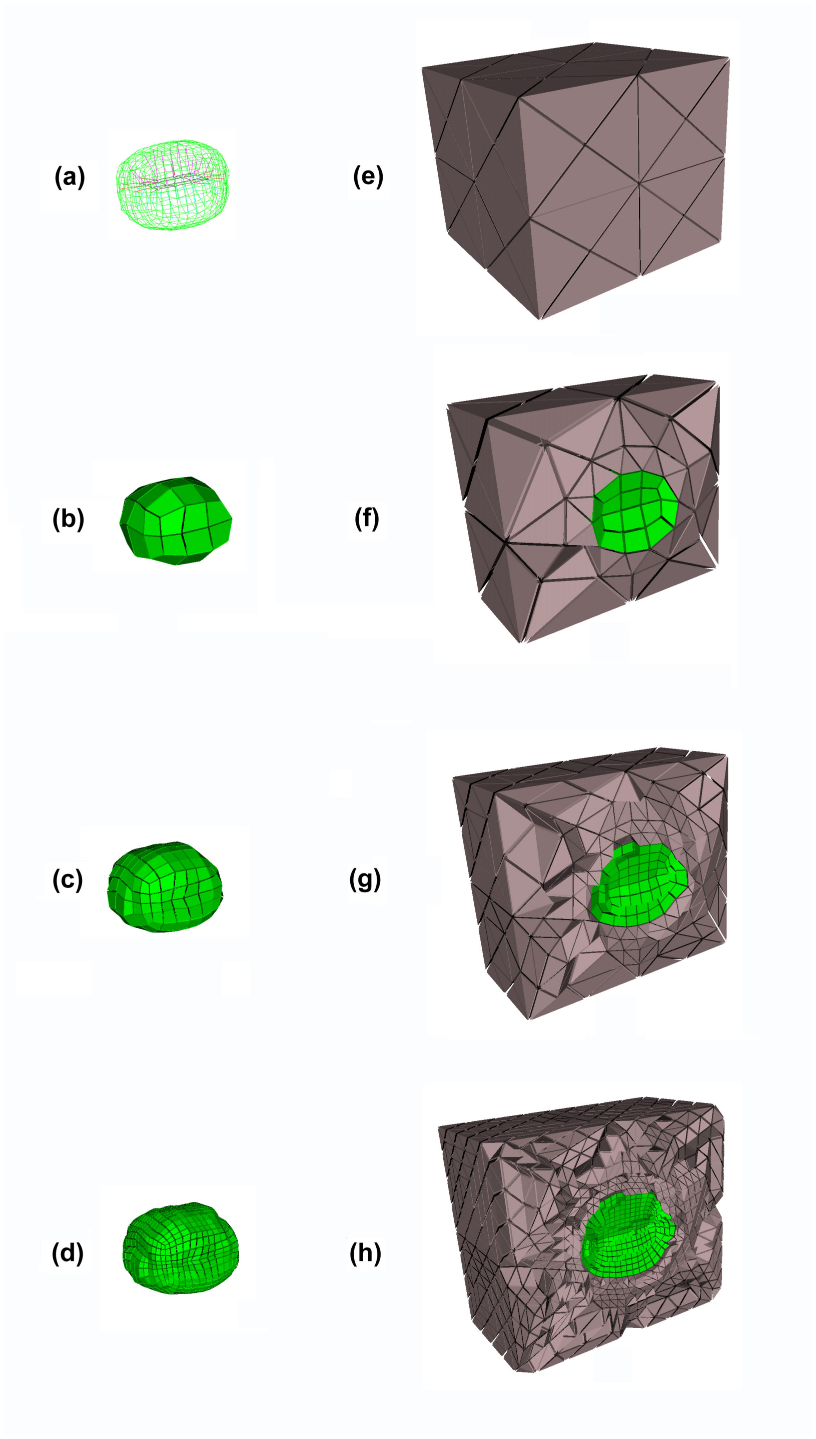


Figure 4.40: (a) Prostate m-rep model (b) Base level prostate mesh (c) Level 2 subdivided prostate mesh (d) Level 3 subdivided prostate mesh (e) Exterior view of the base level meshed volume of interest (f) Sliced view of the base level meshed volume (g) Level 2 subdivided sliced view of the meshed volume (h) Level 3 subdivided sliced view of the meshed volume

Chapter 5

Boundary Conditions

Boundary conditions must be specified for any finite element problem before a solution can be computed. The first section of the chapter examines why this is true and explores the role that boundary conditions play in the finite element analysis of linear elastic problems. The remaining two sections explain how m-rep object models can be used for the prediction and optimization of boundary conditions.

5.1 Boundary Conditions for Linear Elastic Finite Element Problems

Displacements and forces are the two varieties of boundary conditions that can be applied to linear elastic finite element problems. In both cases boundary conditions must be expressed as vectors applied to individual mesh nodes.

The finite element system of equations for linear elasticity was developed in chapter 2. The chapter concluded with the following concise representation for the system of equations. (See equation 2.49.)

$$K\underline{a} = \underline{f} \qquad \text{where} \qquad \begin{array}{l} K = \int_V B^T D B \, \mathrm{d}V \\ \underline{f} = \int_S N^T \overrightarrow{t} \, \mathrm{d}S + \int_V N^T \overrightarrow{b} \, \mathrm{d}V \\ \underline{a} \quad \text{contains the displacements at the nodes} \end{array}$$

An important characteristic of this system of equations is that the determinant of the stiffness matrix, K, is 0. A proof of

$$\det K = 0 \tag{5.1}$$

can be found in [51]. This implies that the stiffness matrix is singular and that an infinite number of solutions to the system of equations exists. However if K is reduced by eliminating any row of the matrix along with the corresponding column, the determinant of the reduced matrix, K_R, is not 0. A proof of

$$\det K_R \neq 0 \qquad \text{where} \qquad K_R \text{ is any submatrix of K} \tag{5.2}$$

is also given in [51]. As explained in [51], specifying a component of the displacement vector $\underline{a}$ allows the stiffness matrix to be reduced in the following way.

First, let the specified component of $\underline{a}$ be g_a, and partition the system of equations in this way:

$$\left[\begin{array}{c|cccc} K_{11} & K_{12} & K_{13} & \cdots & K_{1n} \\ \hline K_{21} & K_{22} & K_{23} & \cdots & K_{2n} \\ K_{31} & K_{32} & K_{33} & \cdots & K_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & K_{n3} & \cdots & K_{nn} \end{array}\right] \left[\begin{array}{c} g_a \\ \hline a_2 \\ a_3 \\ \vdots \\ a_n \end{array}\right] = \left[\begin{array}{c} f_1 \\ \hline f_2 \\ f_3 \\ \vdots \\ f_n \end{array}\right] \tag{5.3}$$

Then define these submatrices and vectors (recalling that K is symmetric):

$$A_1 = \begin{bmatrix} K_{11} \end{bmatrix} \qquad A_2 = \begin{bmatrix} K_{12} & K_{13} & \cdots & K_{1n} \end{bmatrix}$$

$$A_2^T = \begin{bmatrix} K_{21} \\ K_{31} \\ \vdots \\ K_{n1} \end{bmatrix} \qquad K_R = \begin{bmatrix} K_{22} & K_{23} & \cdots & K_{2n} \\ K_{32} & K_{33} & \cdots & K_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n2} & K & \cdots & K_{nn} \end{bmatrix} \tag{5.4}$$

$$\underline{g} = [g_a] \qquad \underline{a}_R = \begin{bmatrix} a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} \qquad \underline{r} = [f_1] \qquad \underline{f}_R = \begin{bmatrix} f_2 \\ f_3 \\ \vdots \\ f_n \end{bmatrix} \tag{5.5}$$

This allows the system of equations to be written as follows:

$$\begin{bmatrix} A_1 & A_2 \\ A_2^T & K_R \end{bmatrix} = \begin{bmatrix} \underline{g} \\ \underline{a}_R \end{bmatrix} = \begin{bmatrix} \underline{r} \\ \underline{f}_R \end{bmatrix} \tag{5.6}$$

Performing the matrix-vector multiplication gives these equations:

$$A_1 \underline{g} + A_2 \underline{a_R} = r \qquad \Rightarrow \qquad r = A_1 \underline{g} + A_2 \underline{a_R} \tag{5.7}$$

$$A_2^T \underline{g} + K_R \underline{a_R} = \underline{f_R} \qquad \Rightarrow \qquad K_R \underline{a_R} = \underline{f}_R - A_2^T \underline{g} \tag{5.8}$$

Equation 5.8 represents the reduced system of equations that is not singular and can be solved using any of the standard methods of linear algebra. Once equation 5.8 is solved and $\underline{a_R}$ is known, the vector can be plugged into equation 5.7 and $\underline{r}$ can be calculated. In this way all of the unknown quantities from equation 5.3 can be found.

The conclusion is that the computation of a deformation requires one or more node displacements to be specified. The previous exercise showed from a mathematical point of view that a displacement type boundary condition was necessary. Intuitively, consider that if the displacements of none of the nodes were specified a rigid body translation could be applied to an entire mesh without affecting the force equilibrium equations (see equation 2.23). Therefore given a solution to a finite element system of equations with no displacement type boundary conditions, another equally valid solution could be generated by shifting all of the nodes by some arbitrary vector.

For the experiments described in this document, displacement type boundary conditions were generated from pairs of m-rep models. The following section describes the m-rep based algorithm that was developed to automatically estimate displacement type boundary conditions.

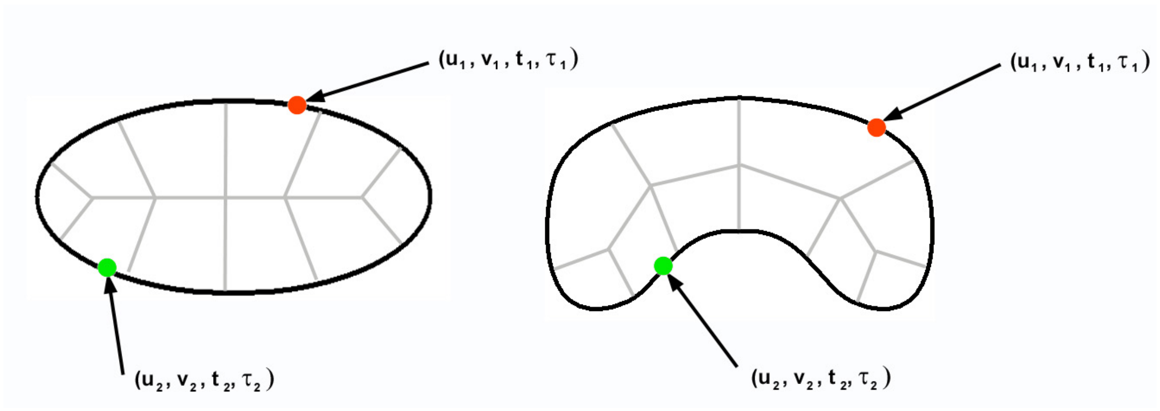
5.2 Initial Approximation

As previously mentioned, boundary conditions can be specified in terms of either forces or displacements applied to nodes. However for an image registration problem neither forces nor point displacements are available directly from the images. What is visible in the images is shifting and/or change in the shape of the object boundary. M-reps provide a way to derive an initial approximation to point displacements from observed boundary changes in an image.

In the prostate case, the m-rep model that was fit to the original image and used to guide mesh construction is transferred onto the image of the deformed prostate and automatically fit to it. The original and deformed m-rep models have the same topology, and their object based coordinates span exactly the same parameter space. This means that the m-rep's object based coordinate system defines a one to one geometry based correspondence between points in the original and deformed prostate. Fig. 5.1 illustrates this geometry based based correspondence. For each surface node with medial coordinates (u_i, v_i, t_i, τ_i) the approximate displacement vector d_i is defined in the following way.

$$\begin{aligned}
\begin{bmatrix} x_i \ y_i \ z_i \end{bmatrix} &= MedialToWorld(model, u_i, v_i, t_i, \tau_i) \\
\begin{bmatrix} x'_i \ y'_i \ z'_i \end{bmatrix} &= MedialToWorld(model', u_i, v_i, t_i, \tau_i) \\
\overrightarrow{d_i} &= \begin{bmatrix} x'_i \ y'_i \ z'_i \end{bmatrix} - \begin{bmatrix} x_i \ y_i \ z_i \end{bmatrix} \qquad (5.9) \\
&\text{where } MedialToWorld() \text{ maps} \\
&\text{medial coordinates to world space}
\end{aligned}$$

The nodes on the exterior surface of the whole meshed volume are assumed to be fixed, so their displacements are set to 0. These exterior surface nodes are the ones lying on the outermost faces of the tetrahedra seen in Fig. 4.10e.



(u_1, v_1, t_1, τ_1)
(u_1, v_1, t_1, τ_1)
(u_2, v_2, t_2, τ_2)
(u_2, v_2, t_2, τ_2)

Figure 5.1: The m-rep figure on the right is a deformed version of the figure on the left. The m-rep object coordinate system that is shared by both versions of the figure establishes a correspondence between boundary points. (u_1, v_1, t_1, τ_1) and (u_2, v_2, t_2, τ_2) mark two sets of corresponding points.

Using this set of boundary conditions will result in a deformed object that evidences the shape change observed in the image and captured by the m-rep. However, this geometry based approximation of the boundary conditions is not the only set of boundary conditions that accomplishes the desired shape change. In order to improve the accuracy of the computed deformation, the boundary conditions applied to the surface of the modeled object(s) are optimized so that the energy of the deformation is minimized.

5.3 Boundary Condition Optimization

In the optimization step the assumption is made that, given two sets of boundary conditions that both result in the observed shape change of the boundary, the one that requires the smaller energy to accomplish the deformation is more likely.

The boundary conditions being optimized are applied only to the nodes on an object surface, so the optimization is most efficiently performed using the nodes' (a, b) surface coordinates which are explained in appendix A. For any node i on the surface of an object, (a_i, b_i) references the location on the surface of the original *model* and (a'_i, b'_i) denotes points on the surface of the deformed *model'*. Then the displacement d_i is as follows.

$$\begin{aligned} \begin{bmatrix} x_i \; y_i \; z_i \end{bmatrix} &= SurfaceToWorld(model, a_i, b_i) \\ \begin{bmatrix} x'_i \; y'_i \; z'_i \end{bmatrix} &= SurfaceToWorld(model', a'_i, b'_i) \\ d_i &= \begin{bmatrix} x'_i \; y'_i \; z'_i \end{bmatrix} - \begin{bmatrix} x_i \; y_i \; z_i \end{bmatrix} \end{aligned} \tag{5.10}$$

Initially $(a'_i, b'_i) = (a_i, b_i)$. In the optimization process the (a'_i, b'_i) coordinates are adjusted so that the energy of the deformation is minimized. Fig. 5.2 illustrates the optimization.

The work required for the deformation is equal to the change in the object's elastic potential. The general formula for elastic potential energy of a three dimensional linearly elastic body is as follows [12] :

$$PE = \frac{1}{2}\int_V \sigma \cdot \varepsilon \mathrm{d}V - \int_V b \cdot u \mathrm{d}V - \int_S t \cdot u \mathrm{d}S \tag{5.11}$$

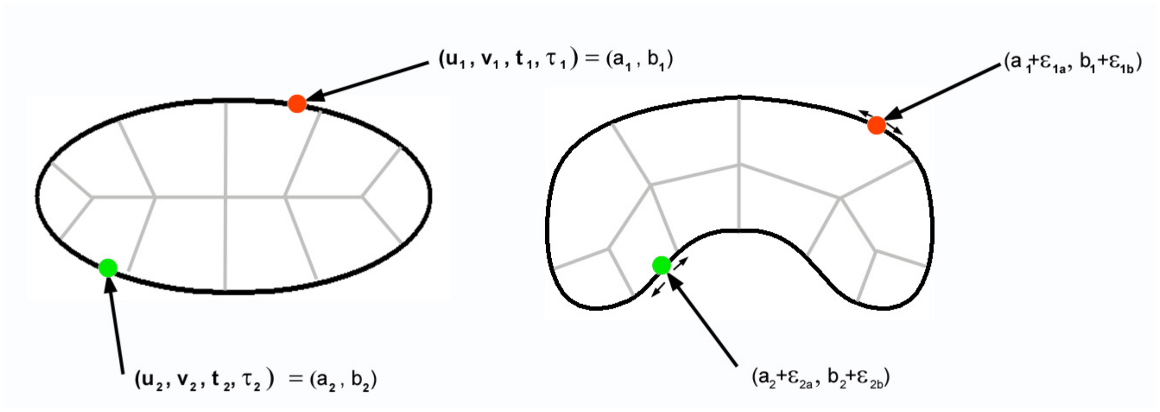

$(u_1, v_1, t_1, \tau_1) = (a_1, b_1)$
$(a_1+\varepsilon_{1a}, b_1+\varepsilon_{1b})$
$(u_2, v_2, t_2, \tau_2) = (a_2, b_2)$
$(a_2+\varepsilon_{2a}, b_2+\varepsilon_{2b})$

Figure 5.2: In the boundary condition optimization process, (u, v, t, τ) coordinates are converted to the more compact (a, b) surface coordinates. Corresponding points are then allowed to slide along the object surface, as shown on the right, to minimize the energy of a deformation.

where

u	is the vector of node displacement
b	is the body force applied to the object's volume
t	is the force applied to the object's surface
σ	is the stress
ε	is the strain

For the finite element problem constructed here, only boundary displacements are specified. No body forces or surface forces are considered. Therefore, for this problem the second and third terms of the potential energy formula are 0, and only the first term needs to be calculated.

For the prostate phantom registration problem, the boundary condition optimization resulted in a 20% reduction in the energy of the deformation and an average change of .12 cm in the optimized boundary displacement vectors that were applied to nodes on the prostate surface. Interestingly, this optimization had a small negative impact on the accuracy of the deformation computed with the base level mesh, and it had a negligible impact on the accuracy of the deformation computed using the second and third level subdivided meshes. These results are reported in more detail in chapter 7.

The likely explanation for the boundary condition optimization not resulting in any improvement in the accuracy of the prostate deformation is that the m-rep prediction algorithm provided a sufficiently accurate initial estimate of the boundary conditions. For deformations that involve little or no torsion or other complicated warps, the m-rep geometry predicted boundary conditions are probably sufficiently accurate for image registration problems. Additional testing of the boundary prediction algorithm with a variety of objects and a variety of deformations would be useful to determine the limits of the algorithm's reliability.

Chapter 6

Multiscale Solution Algorithm

Computing a deformation via the finite element method involves solving a system of linear equations. A number of methods exist for solving systems of linear equations, including direct methods such as Gaussian elimination and Cholesky factorization, iterative methods such as Gauss-Seidel [31] and conjugate gradient [31], and more sophisticated methods based on wavelet [59] or multigrid [78] theory. Any of these solution methods may be applied to the finite element system of equations, but the efficiency of the available methods varies widely. Computational efficiency is an important concern for finite element analysis because the system of equations can be quite large. When a mesh is subdivided, the number of nodes and therefore the size of the system of equations increases dramatically, making efficiency even more important.

In the first part of this chapter the solution algorithm applied to the base level mesh is motivated and explained. In the second part the multiscale solution algorithm that is applied to subdivided meshes is discussed.

6.1 Solution on a Base Level Mesh

For a three dimensional mesh with N nodes, the full size of the stiffness matrix, K, defined by equation 2.49 is $3N \times 3N$. After the system has been reduced through the application of boundary conditions as described in section 5.1, the size of the reduced matrix, K_R, is $(3(N - N_{BC})) \times (3(N - N_{BC}))$ where N_{BC} is the number of nodes for which displacement type boundary conditions are specified. The reduced matrix, K_R, is symmetric positive definite [51].

As mentioned, a variety of solution algorithms are available for this type of matrix; a primary criteria for choosing an algorithm is computational efficiency. LU factorization based on Gaussian elimination is a well known general purpose solution method. For an $N \times N$ matrix, LU factorization requires $O(N^3)$ operations. Cholesky factorization and other direct methods are also $O(N^3)$ algorithms.

Basic iterative methods such as Gauss-Seidel have similar computational complexity. Each iteration requires a matrix-vector multiplication which is ordinarily an $O(N^2)$ operation. $O(N)$ iterations are typically required, which means the algorithm needs $O(N^3)$ operations to reach a solution. However, the stiffness matrix is sparse, and when a sparse matrix data structure is employed, the cost of matrix-vector multiplication is reduced to $O(N)$. This represents a significant savings and brings the number of operations required by a basic iterative method down to $O(N^2)$.

The conjugate gradient method is an iterative algorithm that converges more quickly than the basic iterative methods. The exact rate of convergence is problem dependent and is related to a matrix's distribution of eigenvalues. The number of operations required to compute a solution with a conjugate gradient algorithm when using a sparse matrix data structure is somewhere between $O(N^2)$ and $O(N)$. It has been estimated that a properly preconditioned conjugate gradient method may perform as well as $O(N^{\frac{9}{8}})$ for a three dimensional finite element system of equations [80], although this has not been proven.

Because of its desirable convergence behavior, a preconditioned conjugate gradient solver was chosen as the solution method for base level meshes. As with any iterative method, an initial solution guess is required for the first iteration. While it is permissible to make an arbitrary guess, such as the zero vector, m-rep models provide a way to make a more accurate guess. As explained in section 5.2 and illustrated in Fig. 5.1, the m-rep based object coordinate system that is shared by deformed versions of an m-rep model can be used to establish a correspondence between points in different versions of the model. In chapter 5 the discussion was confined to finding corresponding points on the surfaces of objects, but the same technique can also be applied to interior points. Equation 5.10, reproduced below, is used to compute $\overrightarrow{d_i}$, an initial estimate for each node's displacement vector.

$$\begin{aligned}\begin{bmatrix} x_i \; y_i \; z_i \end{bmatrix} &= model_{xyz}(u_i, v_i, t_i, \tau_i) \\ \begin{bmatrix} x'_i \; y'_i \; z'_i \end{bmatrix} &= model'_{xyz}(u_i, v_i, t_i, \tau_i) \\ \overrightarrow{d_i} &= \begin{bmatrix} x'_i \; y'_i \; z'_i \end{bmatrix} - \begin{bmatrix} x_i \; y_i \; z_i \end{bmatrix} \\ &\text{where } model_{xyz}() \text{ maps medial coordinates to} \\ &\text{world space, as defined in equation 4.1.}\end{aligned}$$

With the base level prostate mesh the computational savings that were realized by using the m-rep based solution estimate versus an arbitrary estimate were small because the preconditioned conjugate gradient algorithm required few iterations to converge, as shown in Table 6.1. However, the savings become more noticeable for meshes that have more nodes and are represented by larger systems of equations. This observation is supported by the results reported in Table 6.1 for the subdivided meshes that have significantly more nodes than the base level mesh. The next section discusses in detail the issues of solving a finite element system of equations on a subdivided mesh.

Table 6.1: Solution Iterations Required for Subdivided Mesh Levels

In the rightmost two columns of the table each row reports the number of conjugate gradient iterations that were required at a mesh level, given that the initial solution approximation for that level was either arbitrary or m-rep predicted.

Subdivision level	Nodes	Elements	Conjugate Gradient Iterations	
			with arbitrary initial approximation	with m-rep interpolated initial approximation
1	254	836	2	2
2	1,836	6,792	5	4
3	14,068	54,960	26	13

6.2 Solution on a Subdivided Mesh

As previously noted, the size of a system of finite element equations grows quickly with mesh subdivision. Consider that the level 3 prostate mesh has 16,530 nodes as reported in table 6.1, so the size of the stiffness matrix for this mesh is $3(16,530 - N_{BC}) \approx 40,000$. This is the size of the matrix for a simple one figure mesh; meshes with more complicated geometry can be significantly larger. While it is reasonable to solve such large systems of equations with the preconditioned conjugate gradient algorithm discussed in the previous section, an additional option exists for a subdivided mesh that was created by refining a coarser mesh.

Any function defined on a coarse mesh can be approximated on a subdivided mesh through interpolation on the coarse mesh. Intuitively, it seems reasonable that a deformation computed on a coarse mesh would be a good estimate of the same deformation, computed with more precision on a subdivided mesh. With an iterative solution algorithm, the number of iterations required depends partially on the accuracy of the initial solution approximation. So one approach to solving a finite element system of equations on a subdivided mesh is to first solve the equations on the coarser mesh and then interpolate the solution to the subdivided mesh and refine the solution with additional solution iterations.

This idea of using a multiscale approach to solving systems of equations is explored in multigrid theory. The solution algorithm applied to subdivided meshes in this work is not an implementation of the full multigrid method but is based on the multigrid concept. An overview of multigrid theory is provided in 6.2.1, and the solution algorithm applied to subdivided meshes is covered in 6.2.2.

6.2.1 Overview of Multigrid

Multigrid theory is covered in depth in [78] and [80]. A brief synopsis is provided here to motivate the solution algorithm presented in 6.2.2.

The multigrid approach begins with an initial approximation to the the solution and finds the final solution by computing the error of the initial approximation and adjusting the approximation appropriately. A key observation of the multigrid method is that the error consists of both high and low frequency components. The smooth, low frequency part of the error can be well approximated on a coarse mesh, while the high frequency part can only be represented on a fine mesh. Prolongation and restriction operators are defined to transfer the error approximation from any given mesh onto a finer mesh (prolongation) or a coarser mesh (restriction).

The error approximation on a mesh can be smoothed through the application of certain iterative solver, such as Gauss-Seidel. This is illustrated in Fig. 6.1. The multigrid method operates by recursively smoothing an error estimate and restricting it to a coarser mesh until some base level mesh is reached. The error is computed exactly on the base level mesh and then prolongated back onto finer meshes where additional solution iterations refine the error estimate.

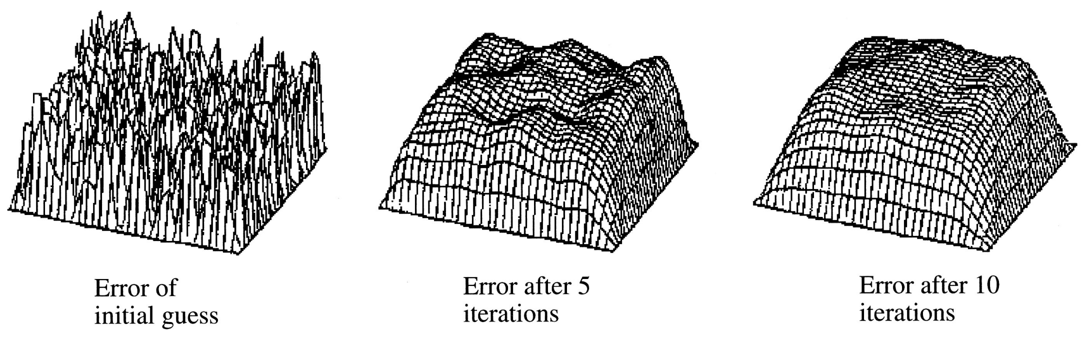


Figure 6.1: Error smoothing that results from Gauss-Seidel iterations. Figure copied from [78].

A variety of multigrid strategies exist, and one of the respects in which they differ is the order in which mesh levels are visited. The order depends on a particular method's scheduling policy; both fixed schedule and adaptive scheduling algorithms exist. Fig. 6.2 diagrams several popular fixed schedule schemes.

A highly attractive quality of the multigrid approach is its computational order of complexity. A full multigrid solution algorithm can produce a solution with $O(N)$ operations because

1. the multigrid algorithm can take advantage of the sparse structure of the stiffness matrix, so that matrix-vector multiplication is an $O(N)$ operation and

2. by exploiting the multiscale nature of the problem, only $O(1)$ iterations are required to produce a solution.

A proof of this order of complexity can be found in [78]. It should be noted that the ideal of $O(1)$ iterations only applies if a multigrid implementation's coarsest grid contains a fixed number of nodes that is unrelated to the size of the full mesh. In

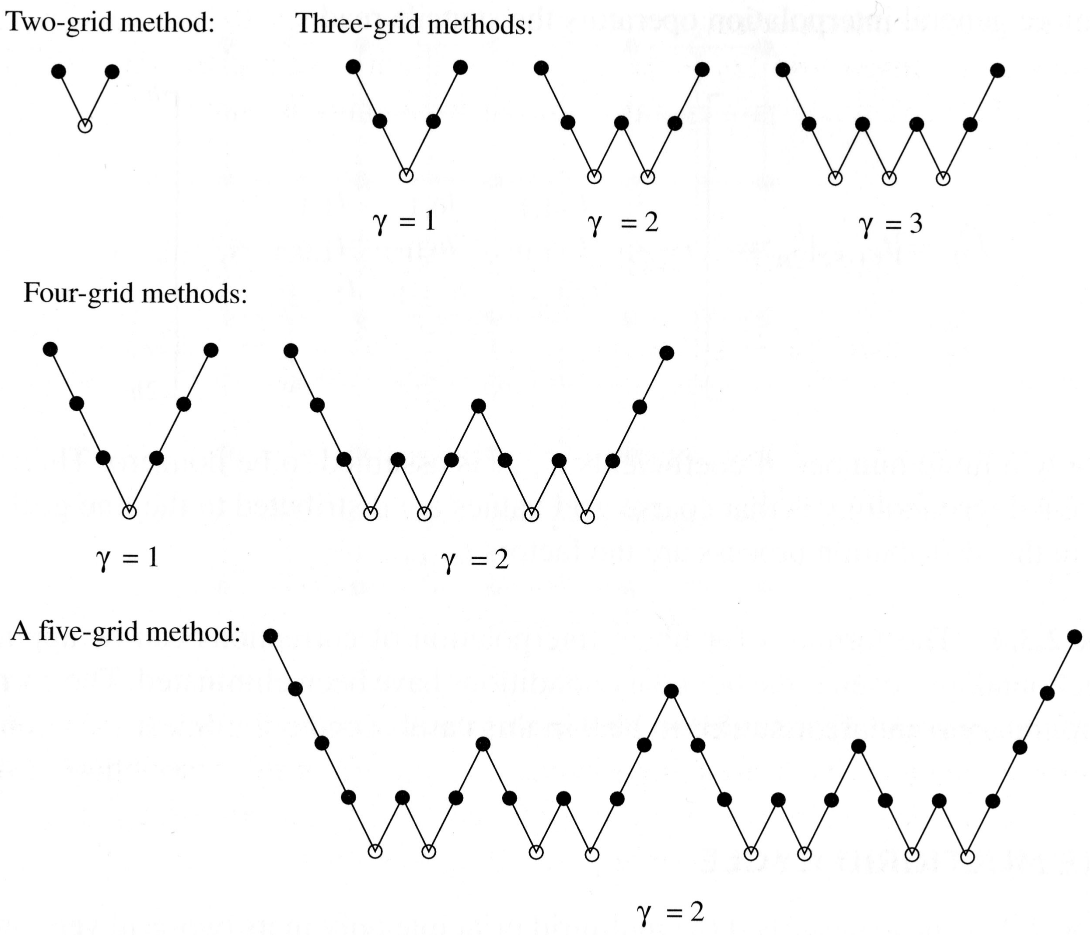


Figure 6.2: Sample multigrid schedules. Hollow circles indicate a base level mesh where the solution is computed exactly. Dark circles indicate a mesh level where smoothing is applied. A circle's vertical placement indicates which mesh level it represents. The progression of circles from left to right indicate successive time steps. Figure copied from [78].

the ideal case, the coarsest mesh contains only one element, as shown in Fig. 6.3. Many implementations stop short of this, so their actual complexity is O(N) plus the number of operations required to compute a solution on the base level mesh. For example, if an $O(N^2)$ algorithm is used to compute the solution for the base level mesh, the cost of the method is $O(N_{full}) + O(N_{base}^2)$ where N_{full} is the number of nodes in the full mesh and N_{base} is the number of nodes in the base level mesh.

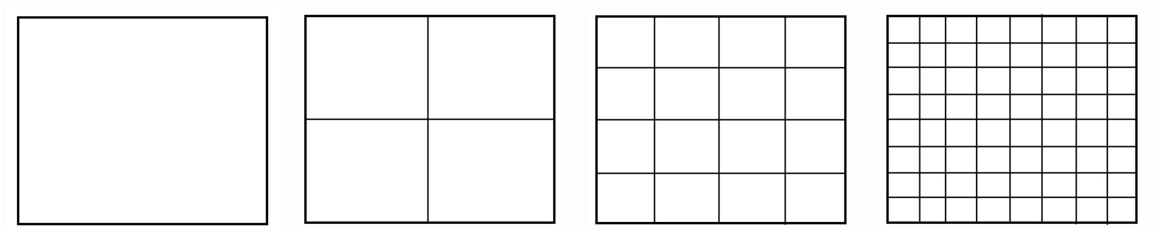


Figure 6.3: A single element coarse mesh on the left, with successively finer meshes to the right. The multigrid method relies on meshes that are subdivided so that edge lengths are halved with each subdivision.

The full multigrid method cannot be readily applied to the finite element problem considered here because there is no obvious method to construct an arbitrarily coarse mesh that maintains the necessary connectivity from a base level mesh generated by the m-rep meshing algorithm. This is due to the fact that the m-rep generated meshes are unstructured. However, the notion of interpolating the solution from a coarser mesh to a finer mesh is applicable to this work. As explained in the next section, the solution algorithm employed for subdivided meshes relies on the multiscale concept inherent in the multigrid algorithm.

6.2.2 Solution Algorithm

The solution approach taken for subdivided meshes is to solve the finite element system of equations on the initial mesh using the procedure outlined in section 6.1 and then interpolate that solution to the subdivided mesh and solve again using the conjugate gradient solver. This is similar to using the multigrid algorithm with the schedule shown in Fig. 6.4. It is also similar to a technique called nested iteration that was one of the predecessor algorithms of the multigrid method [78].

Sufficient information is not available to precisely state the complexity of this solution algorithm. However, upper and lower bounds can be established. The optimistic,

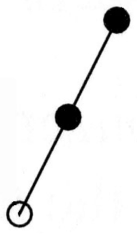


Figure 6.4: Multigrid schedule that corresponds to the approach taken by the solution algorithm applied to subdivided meshes.

best case scenario is as follows:

$O(N_{base}^{\frac{9}{8}})$	– for the preconditioned conjugate gradient algorithm applied to the base level mesh
$O(N_{full})$	– for the multiscale algorithm applied to the subdivided mesh levels

Taken together, these two bounds result in the following best case complexity bound:

$$O(N_{base}^{\frac{9}{8}}) + O(N_{full}) \tag{6.1}$$

There are two reasons to suspect that the complexity of the solution algorithm might not be as good as equation 6.1. First, the $O(N^{\frac{9}{8}})$ bound for the preconditioned conjugate gradient algorithm applied to the base level mesh has not been proven. Second, the $O(N)$ bound for the multiscale algorithm may not apply because the manner in which meshes are subdivided in this work is different from the way subdivision is defined in the proof of the $O(N)$ bound for the multigrid algorithm. The difference is that meshes in this work were subdivided in the m-rep object coordinate system, while the meshes that are assumed to exist in the multigrid proof are subdivided in the Euclidean coordinate system. This difference is illustrated in Fig. 6.5. The way the difference in subdivision technique impacts the theoretical convergence properties of the multigrid method is unknown. Therefore, the worst case complexity bound for this solution algorithm given in equation 6.2 assumes no benefit from the multiscale strategy and is the worst case bound for the preconditioned conjugate gradient algorithm applied to the full mesh. The worst case complexity bound is as follows:

$$O(N_{full}^2) \tag{6.2}$$

Empirically, table 6.1 shows that for the prostate deformation problem the number of operations required does not approach the worst case bound. The fourth column of the table lists the number of iterations required by the preconditioned conjugate gradient algorithm without the benefit of a good initial approximation to the solution. Clearly the number of iterations does not increase quadratically with the node count. The fifth column of the table lists the number of iterations required by the preconditioned conjugate gradient algorithm when the best available initial solution estimate was provided. For the base level mesh, the solution estimate was derived from m-rep geometry based correspondences as described in section 6.1. For the second and third level subdivided meshes, the solution approximations were generated by interpolating the solution from the coarser mesh one level up.

The reduction in the number of required iterations that resulted from the better initial solution approximations indicates that the multiscale approach still provides a benefit when m-rep based subdivision is employed. Table 6.1 shows that the computational savings gained by using the solution approximations increases with the subdivision level. These results indicate that the m-rep based solution algorithm provides better efficiency than the conjugate gradient algorithm alone and that the efficiency improvement increases for finer meshes. Additional results from the m-rep based multiscale solution algorithm are presented in the following chapter.

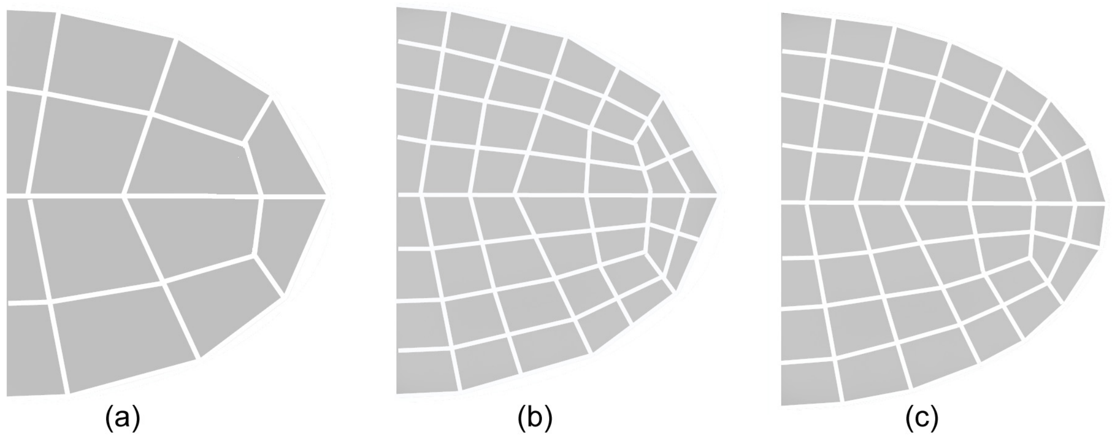


Figure 6.5: (a) A coarse mesh. (b) A subdivision of the coarse mesh using Euclidean space midpoint computations. (c) A subdivision of the coarse mesh using medial object coordinate midpoint computations. The subdivision in (c) provides a smoother approximation of the object's boundary than the subdivision in (b).

Chapter 7

Experimental Results

7.1 Prostate Phantom Experiment Overview

Because the clarity and resolution of CT images are superior to MRSI and ultrasound images, the initial validation study of the methodology was performed using CT images. CT images were acquired at Memorial Sloan-Kettering Cancer Center of a male pelvis phantom with an inflated and deflated MRSI probe in place. The phantom prostate was implanted with 75 seeds, so the accuracy of the computed deformation can be examined by comparing the computed seed displacement with the observed seed displacement. The magnitude of the observed seed displacements is reported in table 7.1.

The seeds closest to the rectum are displaced the most by the imaging probe, and the registration accuracy near the rectum is particularly important for radiation treatment purposes. Therefore, error estimates for the subset of 25 seeds nearest the rectum are computed separately. The magnitude of the observed displacements for this subset of seeds is reported in table 7.2.

For all of the tests reported in this chapter, the single object prostate m-rep model was used and the other pelvic structures were not explicitly modeled with m-reps. The area around the prostate was represented as a homogeneous region.

The linear elastic model has two elastic constants that describe a material's deformable characteristics: E, Young's modulus, and ν, Poisson's ratio. For the first series of experiments the prostate was assigned $E = 60kPa$ and $\nu = .495$ based on the prostate tissue test results published in [35]. The meshed area exterior to the prostate was assigned $E = 10kPa$ and $\nu = .3$. The effect of varying these elastic parameters is examined in section 7.5.

Table 7.1: Magnitude of seed movement

Estimates of actual seed movement between the uninflated and inflated prostate images are provided based on manual segmentation of seed locations in each of the images. The estimates are in units of millimeters and are averages of the measurements for each of the 75 seeds.

total movement	total std. dev.	x movement	x std. dev.	y movement	y std. dev.	z movement	z std. dev.
9.377	2.001	1.392	0.818	8.601	2.125	2.670	1.940

Table 7.2: Magnitude of seed movement for the subset of 25 seeds closest to the rectum

These are the same measurements as provided in table 7.1, except that only the 25 seeds nearest the rectum were included in the computations.

total movement	total std. dev.	x movement	x std. dev.	y movement	y std. dev.	z movement	z std. dev.
11.587	1.488	1.439	0.696	10.738	1.912	3.482	1.742

7.2 Mapping Uninflated → Inflated

The locations of 75 seeds implanted in the phantom prostate were identified manually in both the uninflated and inflated CT images with 3 mm slice thickness and .7 mm within slice resolution. The computed deformation was applied to the identified seed locations in the uninflated image in order to predict the seed locations in the inflated image. The predicted seed locations were then compared to the observed locations in the inflated image. The discrepancies between the predicted and observed seed locations for the entire set of 75 seeds are presented in tables 7.3 and 7.4. The discrepancies between the predicted and observed seed locations for the subset of 25 seeds nearest the rectum are presented in tables 7.5 and 7.6. Results are provided for the first three mesh subdivision levels and for optimized and unoptimized boundary conditions in table 7.3. Histograms of the error distribution are shown in Fig. 7.3 and Fig. 7.4. Fig. 7.1 depicts the shape change that the uninflated prostate mesh undergoes with this deformation. Fig. 7.2 shows how a slice of the CT image changes

Table 7.3: Error estimates for Uninflated → Inflated mapping

Estimates are based on the difference between (a) the observed seed locations in the inflated image and (b)the predicted seed locations that result from applying the computed deformation to the seed locations in the uninflated image. All errors are in units of millimeters and are averages of the measurements for each of the 75 seeds. The x and y components lie in a high resolution image plane. The z component lies across the image planes. The three subdivision levels of the prostate mesh used for these computations is shown in Fig. 4.40.

	Mesh Subdivision Level	total error	total std. dev.	x error	x std. dev.	y error	y std. dev.	z error	z std. dev.
unoptimized	1	2.273	0.934	1.043	0.642	0.696	0.641	1.608	1.036
boundary	2	2.044	0.790	0.857	0.617	0.666	0.534	1.474	0.893
conditions	3	1.999	0.808	0.763	0.580	0.764	0.599	1.392	0.930
optimized	1	2.705	0.869	1.308	0.785	1.026	0.776	1.730	1.057
boundary	2	2.054	0.799	0.852	0.605	0.679	0.547	1.485	0.900
conditions	3	2.000	0.807	0.766	0.580	0.761	0.598	1.393	0.928

when the computed deformation is applied to it and compares the computationally deformed slice to the corresponding slice from the deformed prostate image.

The accuracy of the manual seed labelling was limited by the image resolution, which was coarser in the across slice direction. This limitation contributed to the error estimate, as evidenced by the fact that the error component in the z direction is significantly larger than x and y components. To the extent that errors in seed coordinates contributed to the error estimate, the estimate is indicative of a limitation of the validation procedure rather than a limitation of the registration methodology. For this experiment it was not possible to separate the error due to segmentation from the error due to inaccuracies in the deformation.

Table 7.4: Relative error estimates for Uninflated → Inflated mapping

Relative error estimates are the absolute errors reported in table 7.3 divided by the observed displacements reported in table 7.1. The relative errors in the x and z directions are large because the total amount of seed movement in those directions is small, being approximately the size of one or two voxels in those directions. Therefore the segmentation error in those directions is comparable to the amount of seed movement, making it impossible to evaluate the quality of the deformation in those directions. Most of the seed movement occurs in the y direction, and the relative error is smaller in that direction.

	Mesh Subdivision Level	total relative error	x relative error	y relative error	z relative error
unoptimized	1	24.2%	74.9%	8.1%	60.2%
boundary	2	21.8%	61.6%	7.7%	55.2%
conditions	3	21.3%	54.8%	8.9%	52.1%
optimized	1	28.8%	94.0%	11.9%	64.8%
boundary	2	21.9%	61.2%	7.9%	55.6%
conditions	3	21.3%	55.0%	8.8%	52.2%

Table 7.5: Error estimates for Uninflated → Inflated mapping based on the subset of 25 seeds nearest the rectum

	Mesh Subdivision Level	total error	total std. dev.	x error	x std. dev.	y error	y std. dev.	z error	z std. dev.
unoptimized	1	2.524	0.760	1.349	0.690	1.211	0.684	1.390	0.914
boundary	2	2.270	0.779	1.298	0.581	0.851	0.616	1.310	0.980
conditions	3	2.191	0.787	1.181	0.544	0.884	0.620	1.253	1.020
optimized	1	2.821	0.750	1.313	0.626	1.686	0.723	1.399	1.066
boundary	2	2.299	0.800	1.278	0.555	0.888	0.638	1.348	1.008
conditions	3	2.193	0.787	1.183	0.545	0.884	0.620	1.254	1.019

Table 7.6: Relative error estimates for Uninflated → Inflated mapping based on the subset of 25 seeds nearest the rectum

Relative error estimates are the absolute errors reported in table 7.5 divided by the observed displacements reported in table 7.2.

	Mesh Subdivision Level	total relative error	x relative error	y relative error	z relative error
unoptimized	1	21.8%	93.8%	11.3%	39.9%
boundary	2	19.6%	90.2%	7.9%	37.6%
conditions	3	18.9%	82.0%	8.2%	36.0%
optimized	1	24.4%	91.2%	15.7%	40.2%
boundary	2	19.8%	88.8%	8.3%	38.7%
conditions	3	18.9%	82.2%	8.2%	36.0%

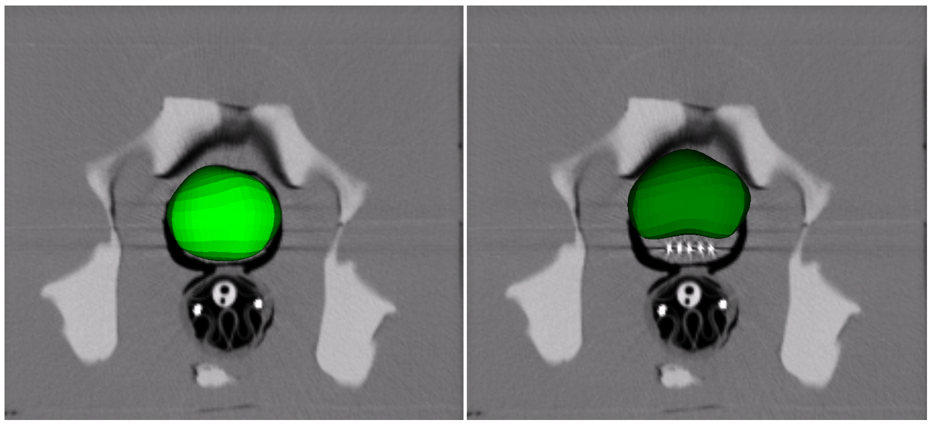


Figure 7.1: Left: Original mesh of the prostate superimposed on a slice of the uninflated CT image. Right: Deformed mesh of the prostate superimposed on the same slice of the uninflated CT.

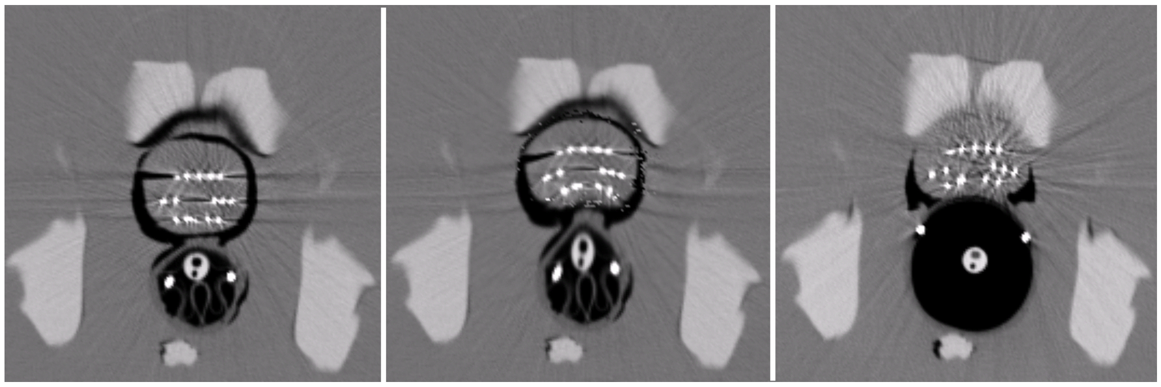


Figure 7.2: Left: CT Slice of the phantom prostate with uninflated probe; Center: Same slice as left, after computed deformation has been applied; CT slice of the phantom prostate with inflated probe

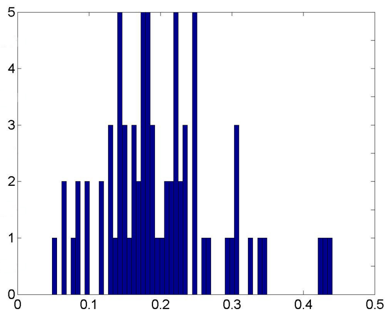


Figure 7.3: Histogram of error estimates for predicted seed locations in the Uninflated $\longrightarrow$ Inflated mapping

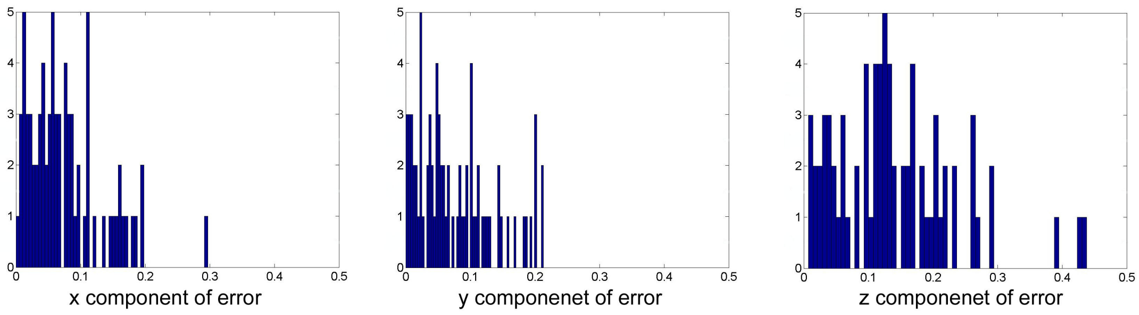


Figure 7.4: Histogram of the separated x, y, and z components of the error estimates for predicted seed locations in the Uninflated $\longrightarrow$ Inflated mapping

Table 7.7: Amount of change in predicted seed locations produced by computing the deformation using a higher subdivision level. Subdivision level $1 \rightarrow 2$ gives the difference in predicted seed locations between mesh level 2 and mesh level 1.

	Subdiv. Level	total change	change std. dev.	x change	x std. dev.	y change	y std. dev.	z change	z std. dev.
unoptimized boundary conditions	$1 \rightarrow 2$	1.584	0.572	0.680	0.636	0.973	0.644	0.637	0.457
	$2 \rightarrow 3$	0.401	0.213	0.154	0.117	0.239	0.194	0.191	0.195
optimized boundary conditions	$1 \rightarrow 2$	1.006	0.515	0.496	0.349	0.493	0.392	0.542	0.470
	$2 \rightarrow 3$	0.374	0.169	0.148	0.111	0.224	0.182	0.174	0.146

7.3 Mapping Inflated → Uninflated

For the brachytherapy application, the MRSI planning image shows a prostate deformed due to the MRSI probe. This planning image with an inflated probe in place must be registered with an intraoperative image of the prostate without an inflated probe. Therefore, it is important to examine the result of applying the registration methodology to the inverse of the previous section's problem.

The image with an inflated probe must be deformed to match the image with an uninflated probe. The results of this mapping are given in table 7.8. The error estimates for this problem are of the same order as the error estimates for the uninflated → inflated mapping. The error is perhaps slightly greater for this problem because the linear elastic model is likely most applicable to a prostate in an unstressed state. For the inflated → uninflated mapping, the initial prostate is in a pre-stressed state due to the pressure from the inflated probe.

Table 7.8: Error estimates for the Inflated → Uninflated mapping

Estimates are based on the difference between (a) the observed seed locations in the uninflated image and (b)the predicted seed locations that result from applying the computed deformation to the seed locations in the inflated image. All errors are in units of millimeters and are averages of the measurements for each of the 75 seeds. The x and y components lie in a high resolution image plane. The z component lies across the image planes.

	Mesh Subdivision Level	total error	total std. dev.	x error	x std. dev.	y error	y std. dev.	z error	z std. dev.
unoptimized	1	3.049	1.337	1.093	0.708	2.038	1.204	1.610	1.101
boundary	2	2.253	0.901	0.721	0.502	1.196	1.007	1.378	0.887
conditions	3	2.153	0.915	0.639	0.471	1.204	0.947	1.309	0.893
optimized	1	2.312	0.953	0.870	0.512	1.135	0.969	1.450	0.960
boundary	2	2.188	0.858	0.718	0.495	1.135	0.960	1.345	0.870
conditions	3	2.161	0.918	0.632	0.472	1.219	0.943	1.315	0.893

7.4 Error Sources

There are three primary sources of error in the prostate registration process.

Table 7.9: Relative error estimates for Inflated → Uninflated mapping

Relative error estimates are the absolute errors reported in table 7.8 divided by the observed displacements reported in table 7.1. See the caption of table 7.4 for further explanation of relative error.

	Mesh Subdivision Level	total relative error	x relative error	y relative error	z relative error
unoptimized	1	32.5%	78.5%	23.7%	60.3%
boundary	2	24.0%	51.8%	13.9%	51.6%
conditions	3	23.0%	45.9%	14.0%	49.0%
optimized	1	24.7%	62.5%	13.2%	54.3%
boundary	2	23.3%	51.6%	13.2%	50.4%
conditions	3	23.1%	45.4%	14.2%	49.3%

1. *Differences between prostate tissue mechanics and the linear elastic model* - As discussed in section 2.4, prostate tissue is not an ideal, homogeneous linearly elastic solid. To the extent that the mechanical behavior of prostate tissue deviates from the linear elastic model, the model will provide inaccurate predictions. There is some evidence to suggest that prostate tissue behaves in an approximately linear elastic way within limited ranges of stress and strain [35]. An inappropriate choices for Poisson's ratio, one of the linear elastic model parameters, could also increase the error. This effect is examined in section 7.5.

2. *Boundary condition error* - Boundary conditions for the prostate deformation problem are specified as displacements of nodes on the prostate surface. Error in these point displacements can be due to inaccuracies in segmentation of the prostate surface and/or inaccuracies in the correspondence established between points on the surfaces of the m-reps representing the deformed and undeformed prostates. The segmentation error is a concern, especially in images where the prostate's border is unclear. The effect of segmentation error is discussed in section 7.6. The correspondence problem probably contributes a small amount to the total error and is managed by the boundary condition optimization procedure described in chapter 5.

3. *Discretization error* - The solution to the linear elastic equations is not analytically derived, but rather approximated on a mesh. There is always some error

in the solution due to this discrete approximation. However, the discretization error can be made arbitrarily small with mesh subdivision. The results in tables 7.3 and 7.8 clearly show that at mesh subdivision level 3 and higher, other components of the error greatly outweigh the discretization component.

Additionally, error in the manual identification of seed locations contributes to the reported error estimates. However this is an error in the validation process, not the registration process.

The largest contributors to the registration error are most likely the overly simplified linear elastic model and inaccuracies in prostate segmentation. If greater accuracy were desired, improvements could potentially be made by developing a more sophisticated and realistic model of tissue mechanics and by using clearer images that allow for more accurate segmentation.

7.5 Effect of Varying Elastic Parameters

The isotropic linear elastic model has two material parameters: Young's modulus and Poisson's ratio. Section 2.6.1 details the significance of Young's modulus and Poisson's ratio and explains how they figure into the linear elastic equations.

Interestingly, if only displacement type boundary conditions are specified for a linear elastic finite element problem and body forces are not considered, the value of Young's modulus does not affect the deformation field solution. Examining the linear elastic PDE shows why this is true mathematically. The linear elastic PDE given in eqn. 2.27 is as follows:

$$\mu\nabla^2\overrightarrow{u} + (\mu+\lambda)\nabla\left(\nabla^T\overrightarrow{u}\right) + \overrightarrow{b} = 0$$

Substituting the definitions of μ and λ given in equation 2.17 produces this equation:

$$\frac{E}{2\nu+2}\nabla^2\overrightarrow{u} + \left(\frac{E}{2\nu+2} + \frac{E\nu}{(1-2\nu)(1+\nu)}\right)\nabla\left(\nabla^T\overrightarrow{u}\right) + \overrightarrow{b} = 0 \qquad (7.1)$$

Since body forces are not considered the $\overrightarrow{b}$ term can be dropped and Young's modulus, E, can be factored out of the equation.

$$E\left[\frac{1}{2\nu+2}\nabla^2\overrightarrow{u} + \left(\frac{1}{2\nu+2} + \frac{\nu}{(1-2\nu)(1+\nu)}\right)\nabla\left(\nabla^T\overrightarrow{u}\right)\right] = 0 \qquad (7.2)$$

The insensitivity of the deformation computation to the Young's modulus value is demonstrated experimentally by the trials presented in the first segment of table 7.11. It should be noted that although Young's modulus does not affect the solution to this type of problem, it does have a scaling impact on the energy of the deformation. If any of the boundary conditions were specified in terms of applied forces rather than node displacements, the value of Young's modulus would affect the solution.

Poisson's ratio lies in the range of .3 - .5 for most living tissues. An incompressible material has a Poisson's ratio of .5, and a Poisson's ratio of .495 is recommended by [35] for the prostate. The computed prostate deformation is sensitive to the value of Poisson's ratio, as is evidenced by the results of the trials shown in the second segment of table 7.11.

Table 7.10: Error estimates for predicted seed locations with varied elastic parameters

All errors are in units of millimeters and are averages of the measurements for each of the 75 seeds.

Young's Modulus	Poisson's Ratio	total error	total std. dev.	x error	x std. dev.	y error	y std. dev.	z error	z std. dev.
6 kPa	.495	2.044	0.790	0.857	0.617	0.666	0.534	1.473	0.893
60 kPa	.495	2.044	0.790	0.857	0.617	0.666	0.534	1.474	0.893
600 kPa	.495	2.044	0.790	0.857	0.617	0.666	0.534	1.474	0.893
60 kPa	.3	2.207	0.818	0.900	0.621	0.720	0.619	1.592	0.961
60 kPa	.35	2.156	0.802	0.870	0.608	0.694	0.583	1.570	0.945
60 kPa	.4	2.101	0.781	0.834	0.594	0.666	0.551	1.543	0.929
60 kPa	.45	2.048	0.761	0.799	0.583	0.643	0.533	1.511	0.909
60 kPa	.495	2.044	0.790	0.857	0.617	0.666	0.534	1.474	0.893

7.6 Effect of Segmentation Error on Registration Results

To examine the effect of segmentation error, five perturbed versions of the prostate m-rep model from the inflated probe image were created. The uninflated $\rightarrow$ inflated mapping was computed for each of the perturbed models, and the seed displacement error estimates were calculated for each of those mappings. The original m-rep model

Table 7.11: Relative error estimates for predicted seed locations with varied elastic parameters

Relative error estimates are the absolute errors reported in table 7.11 divided by the observed displacements reported in table 7.1. See the caption of table 7.4 for further explanation of relative error.

Young's Modulus	Poisson's Ratio	total relative error	x relative error	y relative error	z relative error
6 kPa	.495	21.8%	61.6%	7.7%	55.2%
60 kPa	.495	21.8%	61.6%	7.7%	55.2%
600 kPa	.495	21.8%	61.6%	7.7%	55.2%
60 kPa	.3	23.5%	64.7%	8.4%	59.6%
60 kPa	.35	23.0%	62.5%	8.1%	58.8%
60 kPa	.4	22.4%	60.0%	7.7%	57.8%
60 kPa	.45	21.8%	57.4%	7.5%	56.6%
60 kPa	.495	21.8%	61.6%	7.7%	55.2%

from the inflated probe image has a volume of 39.22 cm^3, and is pictured in Fig. 7.5 along with the perturbed models. A description of each of the perturbed models follows, and the error estimates for each model can be found in Table 7.12.

1. An m-rep model for the prostate with the inflated probe has one end flattened, since it was built from from a stack of contours that were missing the last 4 slices (1.2 cm) of the prostate. Model volume is 36.06 cm^3.

2. An m-rep model for the prostate with the inflated probe has a depression .6 cm deep on its top surface. Model volume is 38.46 cm^3.

3. An m-rep model for the prostate with the inflated probe has a bump .6 cm high on its top surface. Model volume is 40.06 cm^3.

4. An m-rep model for the prostate with the inflated probe has a depression .6 cm deep on its bottom surface. Model volume is 38.38 cm^3.

5. An m-rep model for the prostate with the inflated probe has a bump .6 cm high on its bottom surface. Model volume is 40.23 cm^3.

These results show that segmentation errors do lead to an increase in registration error, as expected. The amount of registration error is related to the extent of the

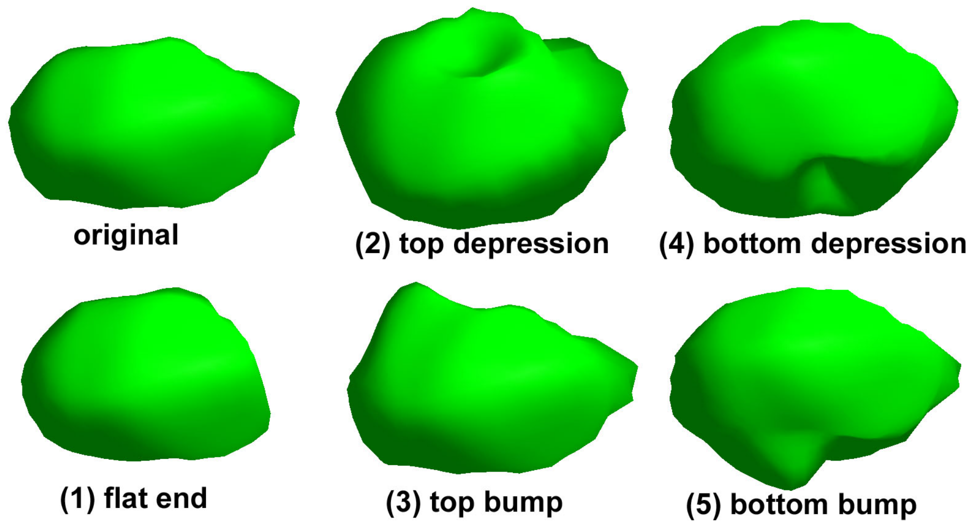


Figure 7.5: Perturbed prostate models used to examine the effect of segmentation errors on the prediction of seed displacement.

Table 7.12: Error estimates for seed locations computed at mesh subdivision level 2 using perturbed m-rep shape models

All errors are in units of millimeters and are averages of the measurements for each of the 75 seeds.

perturbed model	% vol. change	total error	total std. dev.	x error	x std. dev.	y error	y std. dev.	z error	z std. dev.
original	0%	2.044	0.790	0.857	0.617	0.666	0.534	1.474	0.893
1 - flat end	-8.1%	3.016	0.995	1.162	0.809	1.568	0.772	1.828	1.309
2 - top depression	-1.9%	2.062	0.835	0.850	0.623	0.757	0.684	1.408	0.911
3 - top bump	+2.1%	2.176	0.827	0.979	0.690	0.786	0.601	1.474	0.922
4 - bottom depression	-2.14%	2.567	1.719	1.045	0.906	1.417	1.586	1.562	1.117
5 - bottom bump	+2.6%	2.771	1.237	1.068	0.823	1.443	1.476	1.524	0.912

Table 7.13: Relative error estimates for seed locations computed at mesh subdivision level 2 using perturbed m-rep shape models

Relative error estimates are the absolute errors reported in table 7.12 divided by the observed displacements reported in table 7.1. See the caption of table 7.4 for further explanation of relative error.

perturbed model	total relative error	x relative error	y relative error	z relative error
original	21.8%	61.6%	7.7%	55.2%
1 - flat end	32.2%	83.5%	18.2%	68.5%
2 - top depression	22.0%	61.1%	8.8%	52.7%
3 - top bump	23.2%	70.3%	9.1%	55.2%
4 - bottom depression	27.4%	75.1%	16.5%	58.5%
5 - bottom bump	29.6%	76.7%	16.8%	57.1%

segmentation error and the location of the segmentation error. For the prostate model, a segmentation error on the prostate's lower surface had a more detrimental effect than a similar error on the prostate's upper surface.

7.7 Solution Efficiency

The solution process described in chapter 6 requires a few seconds to a few minutes to compute a deformation, depending on the size of the mesh. The benefits of the m-rep predicted solution are realized most fully at higher levels of mesh subdivision. At subdivision level 3, the m-rep solution prediction reduces the number of required conjugate gradient iterations by half for the prostate problem. Table 7.14 lists the number of nodes and elements for each mesh subdivision level, so that mesh complexity can be associated with the solution iteration data presented in table 7.15 and the stiffness matrix computation times reported in table 7.16.

Given the efficient solution strategy, the most computationally intensive and time consuming part of the deformation calculations is the computation of a mesh's stiffness matrix. Stiffness matrix computation times are given in table 7.16 for three levels of

Table 7.14: Complexity of Subdivided Mesh Levels

Mesh Subdivision Level	Node Count	Element Count	Hexahedral Element Count	Pyramid Element Count	Tetrahedral Element Count
1	254	836	76	52	708
2	1,836	6,792	608	312	5,872
3	14,068	54,960	4,864	1,872	48,224

Table 7.15: Solution Iterations for Subdivided Mesh Levels

	Mesh Subdivision Level	Conjugate Gradient Iterations with Arbitrary Initial Guess	Conjugate Gradient Iterations with M-rep Predicted Initial Guess
unoptimized boundary conditions	1	2	2
	2	5	4
	3	26	13
optimized boundary conditions	1	2	2
	2	5	5
	3	26	13

the prostate mesh. A compiled and optimized finite element package would certainly produce stiffness matrices faster than the times reported in the table. However, even with optimized software the stiffness matrix often poses a computational bottleneck. The main technique available for speeding this part of the process is the use of parallel computation.

One mitigating factor is that a mesh's stiffness matrix can be computed before any boundary condition information is available. Therefore in a clinical situation the mesh and stiffness matrix could be pre-computed for a planning image. To produce a registration between a planning image and an intraoperative image, only the relatively fast solution step would need to be done in the clinical setting.

Table 7.16: Time Required for Stiffness Matrix Computation

Mesh Subdivision Level	Stiffness Matrix Computation Time in Seconds
1	6.3
2	859
3	50,431

7.8 Geometry Based Deformation vs. Mechanical Deformation

The initial finite element solution is approximated using m-rep geometry based correspondences. As described in chapter 6, the solution is refined by applying an iterative conjugate gradient solver to the finite element system of equations. The initial geometry based correspondences can be derived more quickly than the finite element mapping, and for some applications the geometry based correspondences may be sufficiently accurate. Therefore, it is important to understand precisely how accurate geometry based correspondences are compared to the finite element mapping.

This section examines how the m-rep geometry correspondences compare with the correspondences generated by the finite element method. Table 7.17 reports the differences between the geometry based prediction of node displacements and the node displacements computed with the finite element method for the prostate uninflated → inflated mapping. For the results computed using unoptimized boundary conditions, only the interior nodes were considered. This was because the m-rep correspondences defined the unoptimized displacement boundary conditions. For the results computed using optimized boundary conditions, all the prostate mesh nodes were included in the results.

Tables 7.18 and 7.19 report the error estimates for seed displacements computed from m-rep correspondences for both the uninflated → inflated and the inflated → uninflated mappings. Tables 7.20 and 7.21 report the same error estimates for the subset of 25 seeds nearest the rectum. These results were computed using the level 2 mesh. Displacements were assigned to the mesh nodes using the m-rep correspondences, and the displacements were interpolated through the mesh to the seed locations.

Table 7.17: Node displacements: Agreement of m-rep geometry based displacements with finite element computed displacements for level 2 mesh

	total diff.	diff. std. dev.	x diff.	x std. dev.	y diff.	y std. dev.	z diff.	z std. dev.
unoptimized boundary conditions	2.536	1.626	0.574	0.494	2.053	1.665	0.975	0.752
optimized boundary conditions	2.561	1.633	0.603	0.544	2.060	1.666	0.984	0.762

Table 7.18: Error estimates for m-rep geometry based prediction of seed displacements

Compare these error estimates to the FEM errors in tables 7.3 and 7.8. All errors are in units of millimeters and are averages of the measurements for each of the 75 seeds.

mapping	total error	total std. dev.	x error	x std. dev.	y error	y std. dev.	z error	z std. dev.
uninflated $\rightarrow$ inflated	2.957	0.914	1.524	0.888	0.937	0.715	1.911	1.205
inflated $\rightarrow$ uninflated	2.920	0.895	1.313	0.778	1.346	1.054	1.730	1.053

Table 7.19: Relative error estimates for m-rep geometry based prediction of seed displacements

Relative error estimates are the absolute errors reported in table 7.18 divided by the observed displacements reported in table 7.1. See the caption of table 7.4 for further explanation of relative error.

mapping	total relative error	x relative error	y relative error	z relative error
uninflated $\rightarrow$ inflated	31.5%	109.5%	10.9%	71.57%
inflated $\rightarrow$ uninflated	31.1%	94.3%	15.65%	64.79%

Table 7.20: Error estimates for m-rep geometry based prediction of seed displacements for the subset of 25 seeds closest to the rectum.

Compare these error estimates to the FEM errors in tables 7.5. All errors are in units of millimeters and are averages of the measurements for each of the 25 seeds nearest the rectum.

mapping	total error	total std. dev.	x error	x std. dev.	y error	y std. dev.	z error	z std. dev.
uninflated $\rightarrow$ inflated	3.108	0.686	1.835	0.933	1.379	0.812	1.567	0.979

Table 7.21: Relative error estimates for m-rep geometry based prediction of seed displacements for the subset of 25 seeds closest to the rectum.

Relative error estimates are the absolute errors reported in table 7.20 divided by the observed displacements reported in table 7.2. See the caption of table 7.4 for further explanation of relative error.

mapping	total relative error	x relative error	y relative error	z relative error
uninflated $\rightarrow$ inflated	26.8%	127.5%	12.8%	45.0%

7.9 M-rep Generated Hexahedral Mesh vs. Tetrahedral Mesh

The superiority of linear hexahedral meshes to linear tetrahedral meshes was discussed in section 3.2.1. An experiment comparing the seed prediction errors that occur with an m-rep generated hexahedral mesh to the errors that occur with a linear tetrahedral mesh supports the view that hexahedral meshes typically perform better.

The tetrahedral mesh used for comparison purposes was generated by the tetrahedral meshing tool available in CUBIT [1] which is based on the TETMESH program developed at INRIA [2]. This meshing program requires a triangulated surface as input and takes element size cues from the size of the surface triangles. To make the comparison as equitable as possible, the quadrilateral faces that lie on the surface of the level 1 hexahedral prostate mesh were split into triangles and passed to the tetrahedral meshing program as the input surface.

As shown in table 7.22, the output tetrahedral mesh had a few more nodes than the level 1 hexahedral mesh and it had many more elements. Since the computation of the finite element system of equations is an $O(N_{nodes}) + O(N_{elements})$ operation, the higher node and element counts imply that computing the solution with the tetrahedral mesh is more computationally expensive.

The displacement boundary conditions applied to the surface nodes of the hexahedral and tetrahedral meshes were identical. Table 7.23 shows that even though the tetrahedral mesh contains more nodes and more elements, its seed prediction errors were worse than those of the hexahedral mesh. Therefore for the phantom prostate deformation problem the m-rep generated hexahedral mesh had a smaller computational cost for the solution step and provided better solution accuracy than the linear tetrahedral mesh. This result does not indicate that a good deformation result could not be obtained with a tetrahedral mesh, since using quadratic tetrahedral elements or smaller linear tetrahedral elements would likely improve the result. However, using smaller elements or higher order elements would incur a computational cost, further increasing the efficiency gap between the tetrahedral and hexahedral meshes. This result illustrates the reason that hexahedral meshing is an active area of research.

The comparison of the hexahedral and tetrahedral prostate meshes concludes the series of experiments that were performed with the phantom prostate images. The implications of the experimental results presented in this chapter are discussed further in the following chapter along with possible future studies.

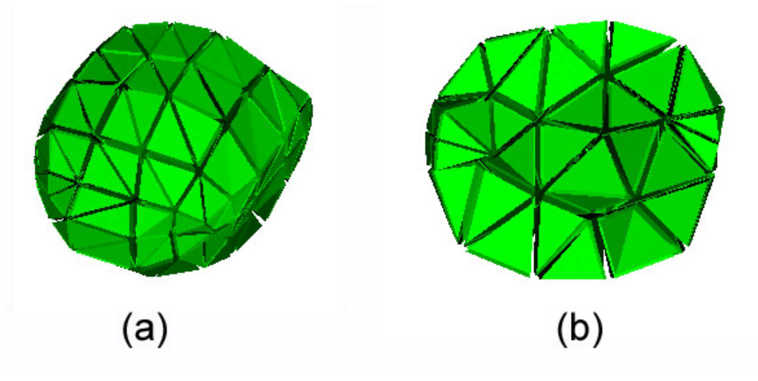


Figure 7.6: The tetrahedral prostate mesh generated for the hexahedral / tetrahedral comparison experiment. (a) exterior view of the prostate mesh surface (b) interior slice view of the prostate mesh

Table 7.22: Mesh size information for level 1 hexahedral and tetrahedral prostate meshes

This table reports the node and element counts for the prostate volume only, unlike table 7.14 that included the counts for the tetrahedral/pyramid mesh that filled the surrounding volume of interest.

mesh type	node count	element count
hexahedral	114	76
tetrahedral	127	431

Table 7.23: Comparison of error estimates for predicted seed locations: m-rep generated hexahedral mesh vs. tetrahedral mesh

The hexahedral mesh used is the level 1 prostate mesh shown in Fig. 4.40(b) and Fig. 4.40(f). The tetrahedral mesh used is shown in Fig. 7.6. All errors are in units of millimeters and are averages of the measurements for each of the 75 seeds.

mesh type	total error	total std. dev.	x error	x std. dev.	y error	y std. dev.	z error	z std. dev.
hexahedral	2.273	0.934	1.043	0.642	0.696	0.641	1.608	1.036
tetrahedral	2.724	1.357	1.083	0.876	1.454	1.300	1.562	1.048

Table 7.24: Comparison of relative error estimates for predicted seed locations: m-rep generated hexahedral mesh vs. tetrahedral mesh

Relative error estimates are the absolute errors reported in table 7.23 divided by the observed displacements reported in table 7.1. See the caption of table 7.4 for further explanation of relative error.

mesh type	total relative error	x relative error	y relative error	z relative error
hexahedral	24.2%	74.9%	8.1%	60.2%
tetrahedral	29.0%	77.8%	16.9%	58.5%

Chapter 8

Conclusion

The first part of this chapter reviews and discusses the accomplishments presented in this dissertation in relation to the claims made in chapter 1. Future work and long term research goals are discussed in the second part of this chapter.

8.1 Claims Revisited

Each of the four claims made in 1.4 is restated in italics and is followed by a summary and discussion of the related accomplishments.

1. *A novel method of automatically generating quality hexahedral meshes from 3D images has been developed. This m-rep based method is designed to mesh anatomical objects found in medical images and is capable of meshing objects that can be modeled with a tree of m-rep protrusion figures.*

   Chapter 4 explained the novel m-rep meshing algorithm in detail. The main features of this algorithm are as follows:

   (a) It generates meshes for single figure and multi-figure m-rep models.

   (b) It creates only hexahedral elements inside modeled figures.

   (c) The process of constructing a mesh from an m-rep model is entirely automatic.

   Example meshes were generated for the prostate and male pelvis models. The quality of these meshes was evaluated using a standard element quality metric and was determined to be sufficient to support stable computation of deformations via finite element analysis. This was an encouraging result, especially since the pelvis model included figures that had some medial sheet torsion and significant width changes.

The single figure meshing algorithm is straightforward, elegant, and effective for the following two reasons:

(a) M-rep models explicitly store both global and local shape information, enabling a top-down approach to mesh design. This is a key attribute that successful hexahedral meshing algorithms share. A number of tetrahedral meshing algorithms generate mesh elements based only on local knowledge of element shape, but various attempts to adapt these algorithms to work with hexahedral elements have failed. This is because it is considerably more difficult to locally alter the structure of a hexahedral mesh than to locally alter the structure of a tetrahedral mesh. The possible configurations of hexahedral elements are more limited than those of tetrahedral elements, necessitating careful top-down planning of a hexahedral meshing pattern.

(b) The m-rep object based coordinate system provides a way to create and subdivide elements in a well behaved and well understood parameter space, and then map the elements into world space. This is advantageous because the construction of a meshing pattern in object based coordinate space is more straightforward than in world coordinates. It is easier to apply mesh templates or a standardized meshing pattern such as the one described in chapter 4 when working in a parameter space with simple, regular boundaries than when dealing with the intricacies and unconstrained curvature of an object boundary in world space.

The multi-figure meshing algorithm works, which is significant since hexahedral meshing of arbitrarily shaped objects is a difficult and open research problem. However, the multi-figure algorithm lacks the elegance and simplicity of the single figure algorithm. This is due to the fact that the object based coordinate system exploited by the single figure meshing algorithm is not continuous at host/subfigure intersections. Host figures and subfigures have separate, overlapping object coordinate systems. This means that an entire multi-figure mesh cannot be constructed in the same parameter space. Instead, the mesh for each figure of a multi-figure model must be constructed in a separate parameter spaces and then the figure meshes must be pieced together to form a single continuous mesh. While this is a reasonable approach which produces acceptable results, both the concept and the implementation have messy details. An obvious remedy is to develop an object based coordinate system that is continuous across host/subfigure boundaries. However, this is a difficult problem, and it is doubtful whether it is possible to construct a continuous object based coordinate system for arbitrarily complicated objects. Another possible alternative for meshing multi-figure models is discussed in the future work part of this chapter in section 8.2.2.1.

2. *A mesh subdivision algorithm based on m-rep object coordinates has been developed. A subdivided mesh produced by this algorithm represents object shape more smoothly and accurately than the original mesh, thereby reducing the error in the finite element solution that is due to the discrete spatial representation of a continuous, curved geometric object. This is a benefit not provided by straightforward Euclidean mesh subdivision.*

   The mesh subdivision algorithm was explained in section 4.2.3 and was illustrated in Figs. 4.15 and 4.16. The subdivision algorithm divides elements in the standard way topologically by creating new nodes at edge midpoints and at the centers of quadrilateral faces and hexahedral volumes. Its distinctiveness lies in the fact that the midpoint and center computations are performed in the medial object coordinate space rather than in Euclidean world space.

   The main advantage of this subdivision algorithm is that each finer mesh created by it more accurately represents the object shape captured by an m-rep model. Therefore the geometric accuracy of a sufficiently subdivided mesh is limited only by the accuracy of its underlying m-rep.

   One limitation of the current subdivision algorithm is that it will only perform subdivision globally; each subdivision step involves subdividing every element in the mesh. It is often desirable to perform subdivision locally so that new nodes are only created within a limited region. Local subdivision is a possible extension of the current algorithm and is discussed as future work in section 8.2.1.1.

3. *The m-rep based correspondence between source and target objects provides a good initial approximation for both the finite element boundary conditions and the solution. An approximation to the boundary conditions is necessary for the*

application of the finite element method. A good solution approximation can reduce the number of iterations required to solve the finite element system of equations.

Chapter 5 covered the estimation and optimization of boundary conditions based on the point correspondences that exist implicitly between deformed versions of an m-rep model. The main features of the boundary condition estimation algorithm are as follow:

(a) It accepts as input two instantiations of the same m-rep model.

(b) It defines a vector field displacement function that maps the surface of one model to the surface of the other model.

(c) The mapping is automatically generated by computing the difference (in world space coordinates) between pairs of surface points that share the same medial object coordinates.

Chapter 6 explained the solution algorithm that is applied to the finite element system of equations. This algorithm derives an initial solution guess from m-rep geometry based correspondences and refines the guess using an iterative solver. The solution on subdivided meshes is computed in a multiscale fashion, where the initial solution estimate on a subdivided mesh is provided by the solution computed on the mesh one level coarser.

For the prostate deformation problem the m-rep based correspondences worked well for both defining boundary conditions and estimating the solution to the finite element equations. However it is not known how well these m-rep based correspondences would work in the case of larger deformations. It is likely that the difference between m-rep based correspondences and physical correspondences would be greater for larger deformations, but additional experimentation is needed to evaluate this.

The degree of difference between m-rep and physical correspondences is closely related to the method of deforming an m-rep. The current method implemented in Pablo is to optimize medial atom quantities (medial sheet position, object radius, etc.) in order to minimize an objective function that combines an image match term and a geometry term. The image match term indicates how well the m-rep's shape matches the shape of an imaged object. The geometry term indicates how well the m-rep's shape matches the expected configuration of the m-rep. There are many ways each of these terms could be defined; the choice of definitions affects how m-reps will deform and therefore what the m-rep correspondences will be. Currently, the geometry term used in Pablo is based on neighbor relationships between atoms. The factors examined includes changes in distance between neighboring atoms and changes in frame between neighboring atoms. Other factors that could be included in the geometry term include the following:

- Multiscale neighbor relationships - If an m-reps' lattice of medial atoms is defined at multiple scales, then the relationships between neighboring m-rep atoms can be examined at different scale levels.
- Medial sheet properties - These include curvature of the medial sheet, compression, stretching, or other distortion of the (u, v) coordinate system on the medial sheet.
- Potential energy metric - A physically based measurement of the energy of a deformation, such as defined by equation 5.11, might indicate the likelihood of a deformation.

Using a physically based geometry term in the m-rep deformation process would probably improve the agreement between m-rep correspondences and physical correspondences, especially for large deformation problems. In any case, if m-rep based correspondences were poor for a large deformation problem, then the methods described in this dissertation would still work. However in such a case the following would be true:

(a) The boundary condition optimization would be more important because the initial estimate would be unreliable.

(b) The solution efficiency gains provided by m-rep correspondences would diminish.

4. *The method presented in this dissertation produces good results for the problem of non-rigidly registering prostate images.*

   Chapter 7 detailed deformation results for a phantom prostate. These results were encouraging and were as good as could be hoped given the method of validation and the resolution of the phantom prostate images. Notable findings included the following:

(a) The accuracy of the computed deformation increased with the subdivision level of the mesh.

(b) Boundary condition optimization did not improve the deformation accuracy for the phantom prostate.

(c) The computed deformation that mapped the uninflated prostate image to the inflated prostate image was slightly more accurate than the computed deformation that mapped the inflated prostate to the uninflated prostate.

(d) Varying the value of Poisson's ratio by 30% or more affected the accuracy of the computed deformation in a non-negligible way.

(e) The deformation accuracy achieved with the second and third level subdivided meshes was in all cases better than the accuracy provided by m-rep correspondences.

These results show that the finite element method can produce useful non-rigid registrations of prostate images and that the automation provided by the m-rep framework supports the application of the finite element method to such non-rigid registration problems. However, all of these results are based on prostate phantom CT images. Further validation and experimentation is needed with clinical images and with image modalities other than CT, not to mention other anatomical areas. This future work is discussed further in section 8.2.1.2.

Finally, the thesis statement is revisited. The thesis statement was as follows:

M-rep based multiscale mesh generation, m-rep derived boundary conditions, and multiscale solution of a finite element system of equations are techniques that improve the automation and efficiency of finite element analysis as it is applied to medical imaging applications and to the prostate brachytherapy application in particular.

Mesh generation, the derivation of boundary conditions, and the multiscale solution algorithm have been reviewed in the survey of the first three claims, and the results of applying these methods to the prostate deformation problem was reviewed in the final claim. The final conclusion is that the automatic methods presented in this dissertation relieve a user of the tasks of managing mesh construction, specifying boundary conditions, and overseeing the solution process without the loss of solution stability or deformation accuracy. This automation makes finite element analysis more accessible and brings it a step closer to being a clinically practical tool.

8.2 Future Work

Finally, consideration is given to ways that this work could be extended to improve performance, add functionality, and provide stronger validation. The future work is divided into three categories based on the complexity of the ideas and the amount of time that would likely be required to explore and develop them.

8.2.1 Further elaboration and validation of current work

The first three future work ideas are straightforward and could be implemented in the near future. These ideas concern local mesh subdivision, clinical validation, and implementation improvements. Detailed explanations are provided in the following sections.

8.2.1.1 Local mesh subdivision

The current mesh subdivision algorithm subdivides an entire mesh and does not provide an option for local refinement. The ability to refine a mesh in a limited neighborhood is valuable, as it allows additional precision to be gained for areas of particular interest without incurring the computational cost associated with reducing the element size throughout the mesh. Local subdivision of a hexahedral mesh must be performed carefully to avoid the creation of hanging nodes. (See section 3.2.4 for a discussion of hanging nodes.) Schneiders [62] presents a nice group of template patterns (Fig. 8.1) that can be applied in hexahedral mesh subdivision and that do not create hanging nodes if properly applied. For m-rep mesh subdivision, these patterns could be applied to elements in the regions that require refinement. The coordinates of newly created nodes could be generated by interpolating between the medial object coordinates of pre-existing nodes, as is done in the current m-rep mesh subdivision algorithm. This approach would provide local mesh refinement functionality while retaining the benefits of subdivision with m-rep object based coordinates.

8.2.1.2 Experimentation and validation with clinical images

Validation of this work has only been performed using prostate phantom computed tomography (CT) images. Further experimentation would determine the applicability of the methods presented in this dissertation to clinical images of the prostate. Validation could be performed using various imaging modalities, including magnetic reso-

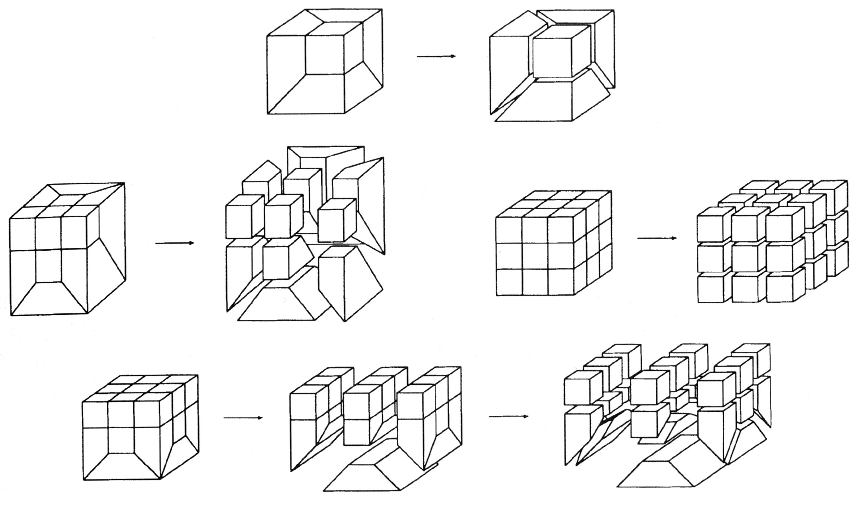


Figure 8.1: Hexahedral element subdivision patterns

nance, magnetic resonance spectroscopy, ultrasound, and CT. Plans have been made to apply the methodology to the registration of magnetic resonance spectroscopy and ultrasound images in a prostate brachytherapy study at Memorial Sloan-Kettering Cancer Center.

Experiments with larger and more complicated deformations, perhaps involving significant torsion, would provide a better understanding of the limits of the m-rep geometry based correspondences and the associated boundary condition prediction method. Good performance for the prostate phantom case does not ensure good performance for all geometries and deformations, and such experiments are needed to develop an understanding of the circumstances in which the boundary condition prediction method works and the quality of the result.

It would also be interesting to apply the methodology to parts of the anatomy other than the prostate. Application to the liver is being considered. The meshing algorithm could be applied to air and fluid filled structures such as the lungs or the heart as part of computational fluid dynamic analysis. Application to other soft tissue organs as well as hard structures, i.e. bones, would be appropriate. An intriguing idea for bone meshing is to assign elastic parameters to elements based on the elements' medial object coordinates. For example, elements with nodes whose coordinates satisfy $|\tau| > 1-\varepsilon$ are near the surface of the bone and should be assigned parameters consistent with cortical bone. Interior elements whose nodes do not satisfy the relation should be assigned parameters consistent with trabecular bone. These are just a few of the ways m-rep based meshing could be applied to anatomical modeling.

8.2.1.3 Implementation issues

In the process of implementing the algorithms described in this dissertation, software was both developed by the author and acquired from other sources. The result is a collection of programs which taken together provide all of the functionality of the described algorithms. This collection is composed of the following:

- Matlab functions and scripts written by the author to generate hexahedral meshes for m-rep figures and to generate transitional pyramid elements.
- Matlab functions and scripts written by the author and adapted from the CALFEM package [25] to assemble and solve finite element systems of equations.
- A C++ program written by the author for viewing image, meshes, and deformation results. This program uses Visualization Toolkit (VTK) classes for image visualization and image warping.
- Pablo, a C++ program for building m-reps and segmenting images using m-reps. Pablo was developed by a number of people in the Medical Image Display and Analysis Group (MIDAG) at the University of North Carolina at Chapel Hill.
- CUBIT, a meshing program from Sandia National Laboratories that was used to create the tetrahedral parts of the meshes.

These programs were adequate for the purpose of prototyping and testing the efficacy of this dissertation's methodology. However, for more widespread distribution fewer programs and faster running times are needed. Ideally the functionality of all these programs would be condensed into a single program. Plans have been made to combine at least the first three items listed above into a single compiled executable.

8.2.2 Major extensions to current work

This section discusses two ideas that would expand the applicability of this work to situations with more complicated geometry and more complicated physics. First, the possibility of using an overset meshing scheme with m-reps is discussed, along with the advantages and disadvantages of such a scheme. Second, the possibility of modeling objects that undergo non-linear deformation is considered.

8.2.2.1 Overset meshing with m-reps

An alternative to the multi-figure meshing algorithm described in section 4.2.4 is overset meshing (section 3.2.5). Overset meshing is simpler and more flexible than the multi-figure meshing algorithm, but it necessitates additional complexity in the solution algorithm.

Overset meshing involves creating a hierarchy of independent meshes. For the m-rep meshing application, a top level hexahedral background mesh with uniformly spaced rows, columns, and layers could be created to fill the entire area of interest and ensure that the space exterior to the m-rep modeled objects is meshed. Then a separate hexahedral mesh could be constructed for each of the figures in an m-rep model using the single figure meshing algorithm described in section 4.2.1. No attempt would be made to join host and subfigure meshes in a compatible fashion; they would simply overlap. The hierarchy of figures in an m-rep model would lead to a hierarchy of meshes. By placing the background mesh at the top of the hierarchy, a complete overset mesh would be generated.

Such an overset mesh would offer several advantages. First, generating an overset mesh as described would be more straightforward and robust than the multi-figure meshing algorithm. In particular, the overset meshing scheme would not suffer the limitations that the multi-figure meshing algorithm has regarding the proximity of branch points or the attachment of subfigures near the corners of the medial sheet's (u, v) parameter space. Second, an overset meshing scheme could handle objects with holes and indentations that the multi-figure meshing algorithm cannot handle. A hole or indentation could simply be modeled with an m-rep figure and meshed like other figures. It would become the responsibility of the solution algorithm to detect the overlap of a hole figure with a solid figure and manage the relationship properly. Finally, an overset mesh would be more appropriate for situations in which there is relative motion between objects, such as occurs between bones in a joint. Elements in a non-overset mesh often become excessively skewed when relative motion occurs. Therefore simulation of such motion can require remeshing at multiple time steps. Using an overset mesh, each figure's mesh could travel with it, reducing the need for costly remeshing.

The main disadvantage to overset meshing is the additional complexity required by the solution algorithm to manage a hierarchy of meshes and transmit deforming forces between meshes appropriately. If an overset meshing scheme was implemented for m-reps, it would be most appropriate to make use of one of the existing finite element or fluid dynamic solvers that accepts overset meshes.

8.2.2.2 Non-linear material models

Linear elasticity is the simplest material model and does not account for all the complexities of human tissue deformation. The most accurate anatomic deformable models incorporate a non-linear material model that describes the way a tissue responds to stress over time and represents any anisotropic aspects of the tissue. For soft tissue organs a viscoelastic material model such as those described in section 2.3.3 is probably most appropriate. However for some tissues, including prostate tissue, testing has not been performed to determine what the best viscoelastic model would be. Therefore the implementation of more advanced material models is dependent on those models first being developed and validated. There is no reason to invest time in implementing and computing deformation simulations with complicated material models unless there is evidence to suggest that a particular complicated model is more accurate than a simple one such as linear elasticity.

Spatio-temporal analysis of object motion and deformation is another research direction that could be pursued with the m-rep methodology. Examples of problems requiring spatio-temporal analysis include breathing motion, heart contraction, and joint movement. Each of these problems involves non-linear deformation, and an accurate simulation would require solutions for a number of time steps. Once again accurate mathematical models of the tissues would be needed before the computational models could be constructed.

The m-rep meshing and multiscale solution approach are applicable for all of these non-linear problems. In each case the necessary changes would involve (1) altering the system of equations that is assembled for a meshed object and (2) altering the solution algorithm to compute solutions for multiple time steps rather than just a single end state. The other aspects of the deformation modeling process could remain as they have been described for linear elastic problems.

8.2.3 Long range possibilities of physically based modeling with medical images

An interesting long range possibility for physically based modeling with medical images is the automated creation of patient specific simulations. Suitably enhanced, anatomic atlases could facilitate the creation of such simulations. The Visible Human Project has created an atlas of human anatomy that goes far beyond the traditional artists' illustrations and provides real medical images of every part of the body, segmented and tagged. Research efforts are underway to further expand the atlas concept by including a characterization of the anatomic variability of a population along with individual instances of the anatomy. These efforts could be further extended by tagging atlas data with not only name and function information but tissue constitutive equations as well. Databases of anatomic variability could include not just the geometric variability of anatomy but the variability of the deformable model parameters as well.

Such an enhanced atlas could be used to automatically create individualized simulations for patients. An image of a patient could be acquired and automatically registered with the atlas. Based on the registration, the atlas' segmentation and tagging could be transferred to the patient image. From the segmentation, a mesh could be automatically constructed. The mesh together with the material models associated with the atlas are sufficient to construct a finite element model for the patient.

Such a model would have many uses including surgical simulation and the simulation of a variety of other medical procedures. Since there would likely be some uncertainty about material parameters such as stiffness and viscosity, several different simulations could be constructed to determine the range of possible outcomes. This kind of tool could help doctors avoid complications and plan procedures for the best possible result.

In order to make this kind of automated simulation possible, advancements are needed in several research areas. First, accurate constitutive equations need to be developed and validated for the organs and tissue types of interest. Studies are also needed to determine the variability of the deformable model parameters across a population. Second, automatic atlas based registration and segmentation methods need to improve. Some positive results have been achieved in this area, but more reliable and general purpose atlas based registration and segmentation software is needed.

Finally, the time required to compute non-linear deformations needs to be reduced in order for such simulations to be practical on a routine basis. Speed improvements may come from hardware acceleration and from improvements in algorithm design. Once these pieces are in place, automated and individualized simulations will be possible.

Bibliography

[1] CUBIT information available at http://endo.sandia.gov/cubit/.

[2] Tetmesh-ghs3d information availabe at http://www.simulog.fr/mesh/gener2.htm.

[3] Fundamental concepts and approaches. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[4] F Azar. A deformable finite element model of the breast to register mr images of the same breast under different plate compressions. *World Congress on Medical Physics and Biomedical Engineering*, 2000.

[5] F Azar. A deformable finite element model of the breast for predicting mechanical deformations. *Journal of Academic Radiology*, pages 965–975, October 2001.

[6] T J Baker. Delaunay-voronoi methods. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[7] K Bathe. *Finite Element Procedures*. Prentice-Hall, New Jersey, 1996.

[8] S E Benzley, E Perry, K Merkley, B Clark, and G Sjaardama. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. *Proceedings, 4th International Meshing Roundtable*, pages 179–191, October 1995. www.andrew.cmu.edu/user/sowen/abstracts/Be172.html.

[9] A Bharatha, M Hirose, N Hata, S Warfield, M Ferrant, K Zou, E Suarez-Santana, J Ruiz-Alzola, A D'Amico, R Cormack, R Kikinis, F Jolesz, and C Tempany. Evaluation of three-dimensional finite element-based deformable registration of pre-and intraoperative prostate imaging. *Medical Physics*, 28:2551–2560, Dec 2001.

[10] H. Blum and R. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10, 1978.

[11] F L Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6), June 1989.

[12] A Bower. Lecture notes for EN224: Linear Elasticity. *Division of Engineering, Brown University*, 1997-1998. http://www.engin.brown.edu/courses/En224/notes.htm.

[13] A Bower. Lecture notes for EN222: Mechanics of Solids II. *Division of Engineering, Brown University*, 2003. http://www.engin.brown.edu/courses/En222.

[14] C Broit. Optimal registration of deformed images. *Dissertation, The Moore School of Electrical Engineering at the University of Pennsylvania*, 1981.

[15] L G Brown. *ACM Computing Surveys*, 24(4), December 1992.

[16] J R Cebral and R Lohner. From medical images to CFD meshes. *Proceedings, 8th International Roundtable*, pages 321–331, October 1999.

[17] M Chabanas and Y Payan. A 3D finite element model of the face for simulation in plastic and maxillo-facial surgery. *Proceedings, Third International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2000.

[18] W M Chan. Hyperbolic methods for surface and field grid generation. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[19] J.M. Chung and N. Ohnishi. Matching and recognition of planar shaped using medial axis properties. *First International Workshop on Multimedia Intelligent Storage and Retrieval Management*, October 1999.

[20] Stphane Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1):62–73, January-March 1999.

[21] J Crouch, S Pizer, E Chaney, and M Zaider. Medially based meshing for finite element analysis of prostate deformation. *submitted to: Information Processing in Medical Imaging*, 2003.

[22] J Crouch, S Pizer, E Chaney, and M Zaider. Medial techniques to automate finite element analysis of prostate deformation. *submitted to: IEEE Transactions on Medical Imaging*, Feb 2003. http://www.cs.und.edu/ jrc/TMI_JRC.pdf.

[23] M Davis. A physics-based coordinate transformation for 3-d image matching. *IEEE Trans. Medical Imaging*, 16:317–327, 1997.

[24] F Boux de Casson and C Laugier. Modeling the dynamics of a human liver for a minimally invasive surger simulator. *Proceedings, Second International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, September 1999.

[25] Department of Mechanics and Materials, Lund University. *CALFEM, a finite element toolbox to MATLAB.* Lund, Sweden, 1999.

[26] M Ferrant, S K Warfield, A Nabavi, F A Jolesz, and R Kikinis. Registration of 3D MR images of the brain using a finite element biomechanical model. *Proceedings, Third International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2000.

[27] M Filipiak. Mesh generation. *Technology Watch Report, Edinburgh Parallel Computing Centre*, 1996.

[28] N T Folwell and S A Mitchell. Reliable whiskerweaving via curve contraction. *Proceedings, 7th International Meshing Roundtable*, pages 365–378, October 1998.

[29] YC Fung. *Biomechanics: Mechanical Properties of Living Tissues.* Springer, New York, 2 edition, 1993.

[30] P L George, F Hecht, and E Saltel. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering*, 92:269–288, 1991.

[31] G H Golub and J M Ortega. *Scientific Computing and Differential Equations.* Academic Press, 1992.

[32] C M Hoffmann. Geometric approaches to mesh generation. pages 31–52, 1995.

[33] R A Katz and S M Pizer. Untangling the blum medial axis transform. *International Journal of Computer Vision*, 2002.

[34] P Knupp. Hexahedral mesh untangling and algebraic mesh quality metrics. *Proceedings, 9th International Meshing Roundtable*, pages 173–183, October 2000.

[35] T A Krouskop, T M Wheeler, F Kallel, B S Garra, and T Hall. Elastic moduli of breast and prostate tissues under compression. *Ultrasonic Imaging*, 20:260–274, 1998.

[36] Y W Kwon and H Bang. *The Finite Element Method Using Matlab.* CRC Press, Boca Raton, Florida, second edition, 2000.

[37] R J Lapeer and R W Prager. Finite element model of a fetal skull subjected to labour forces. *Proceedings, Second International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 1999.

[38] H Lester and S R Arridge. A survey of hierarchical non-linear medical image registration. *Pattern Recognition*, 32:129–149, 1999.

[39] J. Maintz and M. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998.

[40] GN Maksym and JHT Bates. A distributed nonlinear model of lung tissue elasticity. *Journal of Applied Physiology*, 82:32–41, 1997.

[41] K V Mardia and J T Kent. Kriging and splices with derivative information. *Biometrika*, 83(1):207–221, 1996.

[42] R L Meakin. Composite overset structured grids. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[43] R J Meyers, T J Tautges, and P M Tuchinsky. The "hex-tet" hex-dominant meshing algorithm as implemented in CUBIT. *Proceedings, 7th International Meshing Roundtable*, pages 151–158, October 1998.

[44] M I Miga, K D Paulsen, F E Kennedy, A Hartov, and D W Roberts. Model-updated image-guided neurosurgery using the finite element method: Incorporation of the falx cerebri. *Proceedings, Second International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 900–909, September 1999.

[45] K Miller and K Chinzei. Modelling of brain tissue mechanical properties: Biphasic versus single-phase approach. *Proceedings of the 3rd International Symposium on Computing Methods in Biomechanical and Biomedical Engineering*, 1997.

[46] S A Mitchell. A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of the enclosed volume. *Proceedings STACS*, 1996.

[47] A Mohamed, C Davatzikos, and R Taylor. A combined statistical and biomechanical model for estimation of intra-operative prostate deformation. *Proceedings, Fifth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 452–460, 2002.

[48] L. R. Nackman and S. M. Pizer. Three-dimensional shape description using the symmetric axis transform I:theory. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-7(2):187–202, 1985.

[49] GH Olivera, EE Fitchard, PJ Reckwerdt, KJ Ruchala, and TR Mackie. Delivery modification as an alternative to patient repositioning in tomotherapy. *Proceedings of the 13th International Conference on Computers in Radiotherapy*, pages 297–299, 2000.

[50] J O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1998.

[51] N Ottosen and H Petersson. *Introduction to the Finite Element Method.* Prentice Hall, 1992.

[52] X Papademetris, P Shi, DP Dione, AJ Sinusas, RT Constable, and JS Duncan. Recovery of soft tissue object deformation from 3D image sequences using biomechanical models. *Information Processing in Medical Imaging*, pages 352–357, 1999.

[53] J Peraire, J Peiro, and K Morgan. Advancing front grid generation. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[54] RM Pidaparti, Y Liu, and RA Meiss. A viscoelastic material model to represent smooth muscle shortening. *Journal of Bio-Medical Materials and Engineering*, 7:171–177, 1997.

[55] S M Pizer, J Z Chen, P T Fletcher, Y Fridman, D S Fritsch, A G Gash, J M Glotzer, M R Jiroutek, S Joshi, C Lu, K E Muller, A Thall, G Tracton, P Yushkevich, and E L Chaney. Deformable m-reps for 3D medical image segmentation. *International Journal of Computer Vision*, submitted Sept. 2002. http://midag.cs.unc.edu/pubs/papers/IJCV01-Pizer-mreps.pdf.

[56] S M Pizer, P T Fletcher, Y Friedman, D S Fritsch, A G Gash, J M Glotzer, S Joshi, A Thall, G Tracton, P Yushkevich, and E L Chaney. Deformable m-reps for 3D medical image segmentation. *Medical Image Analysis*.

[57] S M Pizer, D S Fritsch, P Yushkevich, V E Johnson, and E L Chaney. Segmentation, registration, and measurement of shape variation via image object shape. *IEEE Transactions on Medical Imaging*, 18(10):851–865, 1996.

[58] S M Pizer, A L Thall, and D T Chen. M-reps: A new object representation for graphics. *ACM Transactions on Graphics*, qqq.

[59] W H Press, S A Teukolsky, W T Vetterling, and B P Flannery. *Numerical Recipies in C*. Cambridge University Press, New York, 1992.

[60] M A Price and C G Armstrong. Hexahedral mesh generation by medial surface subdivision: Part II. solides with flat and concave edges. *International Journal for Numerical Methods in Engineering*, 40:111–136, 1997.

[61] Julia A. Schnabel, Chritine Tanner, Andy A. Castellanp-Smith, Andreas Degenhard, Martin O. Leach, D. Rodney Hose, Derek L.G. Hill, and David J. Hawkes. Validation of non-rigid image registration using finite element methods: Application to breast MR images. *IEEE Transactions on Medical Imaging*, 22(1), January 2003.

[62] R Schneiders. Quadrilateral and hexahedral element meshes. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Florida, 1998.

[63] J Shaw. Hybrid grids. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[64] A Sheffer, M Etzion, A Rappaport, and M Bercovier. Hexahedral mesh generation using the embedded voronoi graph. *Engineering with Computers*, 15:248–262, 1999.

[65] M S Shephard, H L Decougny, R M O'Bara, and M W Beall. Automatic grid generation using spatially based trees. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[66] R E Smith. Transfinite interpolation. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[67] R W Soutas-Little. *Elasticity*. Dover Press, Mineola, New York, 1973.

[68] S P Spekreijse. Elliptic generation systems. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[69] M Styner, G Gerig, S Joshi, and S Pizer. Automatic and robust computation of 3D medial models incorporating object variability. *International Journal of Computer Vision*, to appear spring 2003.

[70] T J Tautges and S Mitchell. Progress report on the whisker weaving all-hexahedral meshing algorithm. *Proc. 5th International Conference on Numerical Grid Generation in Computational Fluid Simulations*, pages 659–670, 1996.

[71] A Thall. Fast C^2 interpolating subdivision surfaces using iterative inversion of stationary subdivision rules. *Technical Report TR02-001, UNC-Chapel Hill*, 2002.

[72] A Thall. *dissertation, UNC-Chapel Hill department of computer science*, 2003.

[73] JP Thirion. Non-rigid matching using demons. *IEEE Conference on Computer Vision and Pattern Recoginition*, 1996.

[74] D S Thompson and B K Soni. Generation of quad- and hex-dominant, semistructured meshes using and advancing layer scheme. *Proceedings of the 8th International Meshing Roundtable*, pages 171–178, October 1999.

[75] J F Thompson, B K Soni, and N P Weatherill. Preface: An elementary introduction. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[76] J F Thompson and N P Weatherill. Fundamental concepts and approaches. In N P Weatherill J F Thompson, B K Soni, editor, *Handbook of Grid Generation*. CRC Press, Boca Raton, Florida, 1998.

[77] S P Timoshenko and J N Goodier. *Theory of Elasticity*. McGraw Hill, New York, 3rd edition, 1970.

[78] U Trottenberg, C Oosterlee, and A Schüller. *Multigrid*. Academic Press, 2001.

[79] M Viceconti, L Bellingeri, L Cristofolini, and A Toni. A comparative study on different methods of automatic mesh generation of human femurs. *Medical Engineering and Physics*, 20:1–10, 1998.

[80] P Wesseling. *An Introduction to Multigrid Methods*. John Wiley and Sons, 1992.

[81] D R White and P Kinney. Redesign of the paving algorithm: Robustness enhancements through element by element meshing. *Proceedings, 6th International Meshing Roundtable*, pages 323–335, October 1997.

[82] J Yao and R Taylor. Tetrahedral mesh modeling of density data for anatomical atlases and intensity-based registration. *Proceedings, Third International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 531–540, October 2000.

[83] P Yushkevich, P T Fletcher, S Joshi, A Thall, and S M Pizer. Continuous medial representations for geometric object modeling in 2D and 3D. *Proc. 1st. Generative Model Based Vision Workshop*, 2002.

[84] M Zaider, M J Zelefsky, E K Lee, and et al. Treatment planning for prostate implants using magnetic-resonance spectroscopy imaging. *Int J Radiat Oncol Biol Phys*, 47:1085–1096, 2000.