# Thesis Proposal:
# Soft Body Simulation for Surgical Training

Nathan Mitchell
University of Wisconsin, Madison
Advisor: Eftychios Sifakis

September 4, 2014

## 1 Introduction

With the demand for medical care consistently rising, the ability of surgical schools to produce well trained surgeons is under intense pressure. Current training aids typically consist of physical models, which are expensive and often single use, or computer simulations, which are also expensive and are often highly specialized to a single type of operation. In the field of plastic surgery, which includes reconstructive surgery, training is even more critical as mistakes are easily visible after the surgery.

However, plastic surgery presents unique challenges which have traditionally made computer simulation impractical. The vast majority of plastic surgery involves operations on soft skin, fat, and muscle layers - each of which exhibit complex material behaviors. The operations themselves commonly include intricate cutting and suturing arrangements which are designed to produce a particular reconstructive goal, all while being mindful of the mechanical properties of the tissue. A successful simulation training tool needs to accommodate all of these aspects, while providing real-time performance. The tool should also be cheap to deploy, as to not place additional burden on an already expensive medical system.

Preliminary research into physics based simulation for surgical applications has historically focused on organ simulation. These simulations typically involve an organ, or collection of organs, being prodded or sliced. Initial attempts were plagued by low fidelity as available hardware and immature algorithms prevented high resolution models from achieving real time performance. However, modern hardware and algorithmic techniques present new opportunities for simulation - the exploitation of commodity hardware with high degrees of parallelism should allow practical real-time performance for large domains. As resolutions increase, interesting opportunities for realistic material behavior become possible which were simply impractical to consider in the past.

The overarching theme of my research is the exploration of soft body simulation in the context of training aids for plastic surgery. In particular, ***it is my thesis that we can construct effective plastic surgery training tools by combining deformable object simulation techniques with the***

***exploitation of modern parallel hardware, all while remaining cost effective and deployable as remote services.***

What makes a surgical training tool effective is a primary question of my research. While I presume several features to be important: interactivity, topology alteration, and collisions; we can currently only guess at other desirable features. Further, training tools are required to do more than instill knowledge into a student - they must also transmit knowledge to and from the instructor in order to successfully evaluate the student's progress.

For deformable object simulations, I intend to focus on techniques using embedding regular lattice based disretizations. Embedding techniques are interesting for their ability to support complex geometry and thin cuts without requiring expensive and difficult online remeshing. Regular lattices provide several advantages which make them suitable for interactive tools. Primarily, the regularity provides opportunities for exploiting parallelism. I will touch on these opportunities later in this document.

Modern parallel architectures exist in great variety, each exhibiting advantages and disadvantages towards any particular programming model. To achieve the highest interactive rates, I have no choice but to explore all available options to determine which combinations of hardware and software structure will contribute the most performance. Far from a simple engineering effort, I expect this branch of my research will contribute novel ideas about data organization and memory utilization.

Finally, I will present methods for deploying simulation tools over a network. This modality provides several potential benefits. By running simulations on central servers, training stations can be made cheaper by using low powered devices instead of customized workstations. Additionally, this allows for experimentation in computational and human interface hardware without requiring the replacement of the entire system. All combined, network deployment should allow easier incremental development and build out, without requiring expensive overhauls with each new feature.

## 2 Contributions

In this section I will detail the contributions I expect will result from my research into simulation training tools:

### Evaluation of Features for Skill Development

As will be covered in section 3.1, the unifying theme for my research is the creation of surgical training tools for cognitive skill development. An important unanswered question which I will address in my dissertation will be which software and simulation features contribute most to this goal. Towards this end, during the course of my research I will develop tools designed to explore this space and conduct studies to determine which features are most helpful for education.

## Biologically Realistic Materials

Accurate simulation of plastic surgery procedures will require faithful representation of biological materials like skin, muscle, and fat. I will present improved material models based on existing hyperelastic materials by introducing elements of plasticity, fracture, and realistic stress response. The construction of these models will be done under the guidance of domain collaborators, with the aim of producing biologically realistic materials without sacrificing the performance of the solver.

## Method for Online Topology Change

Involved surgical operations typically require multiple stages of incision and suturing in order to manipulate tissue into the desired configurations. I will present a method for online topology change to support sequential incision operations during a simulated surgery. Preliminary work with a hybrid mesh-lattice design already supports detecting thin cuts and correctly separates material, but future work will include the dynamic alteration of hybrid lattice topology.

## Method for High Performance Continuous Collisions

For surgical tasks, collision and contact between sections of flesh play an important role in how a surgeon plans and performs the final operation. I will present a method of efficiently handling sustained and wide area contact between deformable objects without impacting the overall stability of the system. This feature will be especially critical for types of surgical operations that depend on friction and contact forces to hold flesh in place rather than suturing.

## Method for Vectorizing Numerical Kernels

A common occurrence in many software projects, algorithms are designed and implemented without attention being paid towards parallelism, which is often considered an after thought. I will present a vectorization toolkit which will allow programmers to implement numerical algorithms in a serial fashion, while providing an automatic and transparent method to move those algorithms onto SIMD style architectures. More details on current progress and future work towards this contribution will be covered in my paper [26] and section 3.3.

## Method for Streaming Simulation Results

A primary goal of my research is the concept of “Simulation as a Service” and towards this end I will present a method for transmitting, or *streaming*, real-time results of simulations to clients on a network. The primary challenge for this contribution is one of bandwidth management. Currently I have only started preliminary investigation towards this goal.

### Method for Biological Model Authoring

The majority of computer modeling tools are mesh based, as such they typically ignore the internal, or volumetric, information of a model. Biological models, in contrast, contain rich internal details that are often crucial to the model's final dynamic behavior. In order for simulation tools to be useful as training aids, instructors and experienced surgeons will need to craft examples and scenarios containing biological objects, and thus will require tools to construct volumetric models. I will present a method for aiding novice users in constructing volumetric models suitable for surgical simulations.

## 3 Research Overview

In this section, I will discuss my research goals in more detail - explaining some background, my intended solution approach, how I intend to evaluate my solution, and any challenges I expect to encounter along the way.

### 3.1 Tools for Cognitive Surgical Skill Development

**Problem Background** Surgical skills suitable for computer assisted training have been divided [8] into two major categories: pyschomotor and cognitive. **Psychomotor** skills are those that involve the surgeon's hands to perform dexterous tool manipulations and other physical operations. These skills often involve physical sensations and help the surgeon build "muscle-memory" with tools like the scalpel. Laparoscopic operations, where surgery is conducted via thin probes inserted into the body, are common scenarios for computer simulations. These simulations can approximate laparoscopic operations through restrictive physical interfaces similar to real probes and are often equipped with force feedback to present the sensations of pressing on and moving organs. Many examples of previous work fall into this category of simulation [25, 7, 19, 21], along with several commercial products [36], [35].

In contrast, **cognitive** skills are those that are primarily mental in nature. Operation design, notably the selection and placement of primitive surgical maneuvers to achieve high level operation objectives, classically falls into the cognitive skill category. Needle steering [4], where the mechanical characteristics of the needle are considered for navigating the tip through tissue, is an example of a cognitive skill.

**Solution Approach** It is a hypothesis of my research that cognitive skill training does not require the physical realism that psychomotor skill simulators include. In particular, I believe features like haptic feedback will be unnecessary for cognitive skill training. Instead, important features will include realistic materials, expressive simulated surgical implements, and information feedback.

Realistic materials are important for cognitive skill training. Beyond simply producing more accurate visual results, realistic material behavior enables a user of a training tool to better understand the effects of their manipulations on a simulated object. An example would be the maneuvering of a skin flap. If simulated using a simple, stretchy material like a corotated elasticity model, a user may be able to stretch the flap farther than realistically possible, thus allowing physically implausible operation designs. A more realistic material for the same skin flap will only stretch a short distance before becoming stiff and will force the user to take appropriate steps when considering where to rearrange tissue.

Similarly, I feel expressive tools will be equally important for cognitive training. While force feedback and dexterous actions like simulating the tying of sutures, are probably unneeded, there exist other forms of realism that structurally alter the capabilities of the simulation. As an example, see [21] where the scalpel tool is capable of not simply cutting through tissue, but also more delicate cutting motions, including slicing thin layers into simulated skin. Supporting advanced tool modalities will allow users the freedom to explore solution approaches to surgical operations that would be otherwise denied with simpler tools.

For the purposes of my thesis, I will define information feedback as the display of non-visual information resulting from deformation that reveals important details about the simulated object. As an example, this could include false color overlays of stress patterns which may be useful in planning where to perform incisions or place sutures.

**Evaluation and Challenges** Admittedly, this component of my thesis is the most undefined. As such, my current plan is to perform a series of experimental demo sessions in collaboration with the University of Wisconsin, Madison's School of Plastic Surgery. The goal of these sessions will be to expose the tools I have developed to a body of surgeons, both experienced and novice, and gather their feedback. This feedback will be used to refine the development simulation features and guide future research directions. As a side benefit, it is my hope that this collaboration will help strengthen ties between the fields of computer science and surgery, allowing for more interdisciplinary research to be performed in this area over the long term.

For the purposes of my research, the main outcome I hope to gain from these sessions is a better sense of what surgeons really want from a cognitive skill training tool and what techniques are actually effective in training. A potential issue I expect may arise is that these two goals will be in conflict. It is my suspicion that what trained medical personal believe they want in a simulator and what they actually find to work will be different.

## 3.2 Modeling Biological Tissues and Surgical Implements

**Problem Background** Human skin, subdermal muscle, and fat are inherently complex natural materials. Some degree of accuracy in simulation of these materials will be required for any surgical training tool. Unlike simple materials like linear elasticity, or even more advanced models like neohookean elasticity, biological materials often contain anisotropic or plastic behaviors. As an example, muscle tissue represents a classic anisotropic material, where the fiber direction of the muscle has a strong influence on its overall elastic behavior under deformation. Even more uniform tissue such as skin exhbits interesting behavior. Normally pliable, skin becomes very stiff once stretched beyond a certain limit. This effect, sometimes referred to as bi-phasic behavior, has important consequences for surgical operation design.

A surgeon must be aware of skin's limits when deciding which sections to move around. Supporting this behavior could be very beneficial for training purposes. An extreme example would be the effects of water loss from tissue. Uninjured, tissue is composed mostly of water and as a result can be considered a highly incompressible material. However, once cut, blood loss drains water from surrounding tissue, resulting in changes to its incompressiblity and elastic behavior. A question my research seeks to answer is what *degree* of realism is required for educational purposes.

In addition to materials, there exist several tools surgeons use on a regular basis that need to be supported within the simulation. From a mathamatical perspective, these tools could be considered to be basic operators, similar to the addition operator in mathematics. Specifically, these tools can be abstracted into three categories: incisions, hooks, and sutures. Incisions represent cutting tissue and may be arbitrarily complex during actual surgeries. Hooks fulfill the purpose of point constraints, used to pull back tissue and manipulate it into position. Finally, sutures encapsulate the inverse of an incision, joining edges of tissue together.

**Solution Approach** I have begun tackling the problem of supporting these features in real time simulation. My approach for material support is the creation of a generic framework capable of describing both isotropic and anisotropic materials. In my most recent work [26], I have demonstrated support for the isotropic materials known as neohookean and corotated elasticity. I show how isotropic materials can be decomposed into a set of simple kernels and material definitions can be changed by reimplementing two stress functions. Support for anisotropic materials is currently restricted to transverse anisotropic materials and, similar to isotropic support, behavior can be adjusted by altering two functions. Future work on material support will depend highly on the results of demos and studies, but possible directions include the addition of plasticity effects and more advanced anisotropic materials capable of supporting fibrous sheets.

I have also begun support for surgical tools. In [26], hooks and sutures are represented as point constraints, either to kinematically placed points or between two points embedded in tissue, respectively. While point constraints serve as a fair approximation for real surgical hooks, they are a poor approximation for real sutures. Ideally, sutures would be simulated by something similar to an inelastic string. However, like advanced material support, future trials will help determine whether a more accurate approximation is truly required for the purposes of education.

Unlike hooks and sutures, incisions are far more complex. Currently, the incision support included within the prototype tool described in [26] is limited to boolean carving operations on thin tissue layers, where the cut passes through from top to bottom. Compared to the cuts made possible on tissue with real scalpels, this implementation is only a shallow approximation. Future work in this direction may include techniques similar to those found in [21]. Haptic feedback for cutting could be another avenue of investigation, but as I proposed in section 3.1, haptics might be unnecessary for training cognitive skills.

**Evaluation and Challenges** I foresee two primary challenges with material development. First is the problem of accuracy. Accuracy in this context refers to how close the behavior of simulated material matches that of real tissue. For readily available materials like various metals, plastics, or rubbers, material behaviors can be quantified by placing samples within strain gauges. This allows us to experimentally determine values like tensile strength, breaking points and other material limits.

I expect biological samples, especially those from human bodies, will prove more difficult to acquire and run through standard tests. My plan to deal with this difficulty is to work with the university's School of Plastic Surgery and the UW Hospital in order to acquire material samples and gather data. Once collected, this data can be used to build more accurate material models.

The second problem with improving material realism is that the addition of material features, such as plasticity effects, may conflict with the parallelization strategy I have already begun developing for [26]. As I will discuss in sections 3.3 and 5, my current approach to vectorization depends on the assumption that forces for each hexahedral cell can be computed independently of every other cell. However, until the decision is made to include such enhancements, I am going to categorize this as a lesser concern.

There are also several potential challenges dealing with simulated tool design. Partly this is due to usability - what is the correct interface to present to a user and so forth; but tool design is also not an isolated component. The functions a tool can perform are intrinsically tied to the underlying simulation. As an extreme example, consider a tool that simulates the stimulation of muscles with an electric charge - unless the simulation has support for a muscle model in the first place, designing the user interface for a muscle stimulator tool is premature. Likewise, if we desire the scalpel tool to have the capability to slice horizontal layers into tissue in addition to existing support for perpendicular cuts, we must determine what a horizontal slice looks like in the simulation. For a completely uniform object, a horizontal slice may not be any different than a perpendicular cut apart from the direction of the incision. But for the human body, with its multiple distinct tissues, it may be more appropriate for a horizontal incision to separate whole tissues layers rather than support arbitrary slices. This is a question that must be posed to the surgical community.

## 3.3 Regular Hexahedral FE Methods on Parallel Hardware

**Problem Background** In years past, processor performance was tied to an ever increasing clock-rate. For the past several years however, improvements to raw speed have slowed as power requirements and heat management issues have come to forefront. Instead, processor designs have become more parallel with the addition of extra cores and SIMD, or Single Instruction Multiple Data, vector units. All major modern processors contain some form of vectorization and most are multicore designs. Other traditionally parallel processors, like GPUs, have only become more capable at generic computation via specifications like OpenCL and CUDA. Finally Intel has released a line of many-core coproccessor cards capable of handling sixteen channel wide floating point vector operations and hundreds of threads.

In the face of these changes and advancements in processor design, parallelism should be

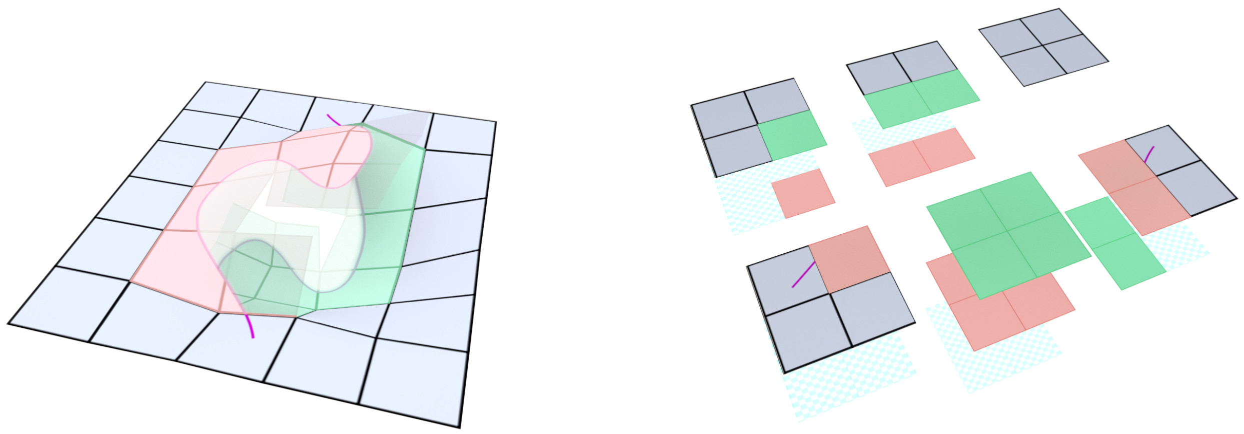

Figure 1: Left: A visualization of non-manifold topology captured by the hybrid mesh-lattice discretization presented in GRIDiron [26]. Right: Visualization of the blocking strategy employed by GRIDiron. Each layer represents a single block that be executed simultaneously with vectorization. Each stack of layers represents the geometric locations of each cell - multiple layers indicates nonmanifold topology.

considered as a first class citizen in software design. Unfortunately, the development of numerical kernels and solvers for finite element methods are often conducted in a serial mindset. Once shown to be correct, only then is the effort undertaken to optimize the serial algorithms as parallel code. There are two major problems with this methodology. First, there is wasted effort as the same algorithms are implemented twice and the first, serial, implementation is often useless in practical scenarios due to its poor performance. The second major problem occurs when alterations are made to the fundamental algorithm. These changes often invalidate highly specific optimizations that were applied during the initial parallelization effort. This typically results in the entire process of serial and parallel implementations being repeated instead of changes being applied directly to the original parallel codebase.

**Solution Approach** My planned strategy for getting the most use out of parallel hardware is threefold. First I will explore the performance enhancements possible from data structure organization. Second, I will develop programming aids for automatic vectorization capable of adapting to multiple SIMD architectures. Finally, I will explore the various new parallel processors coming to market, including GPUs and Intel's Phi, as well as considering designing new CPU architectures specifically for physics simulation.

A large component in simulation performance is the optimal use of hardware resources, specifically the utilization of memory bandwidth and available cache. Regular lattice discretizations have the advantage of implicitly encoding topology. This reduces the amount of data needed to be read from memory when operating on a single voxel. In prior work [26] [32], I have explored a data structure suited for SIMD platforms, further reducing memory pressure by collecting multiple cells into a geometric block. The forces for cells within the block are then computed in lockstep using vector instructions.

I will also approach vectorization via autovectorization techniques. Without autovectorization, migration to new hardware platforms becomes a time consuming and error prone manual task. I propose an autovectorization library which abstracts the specific vector instructions from their mathematical operations. This library makes heavy use of the generics feature in the C++ language, allowing the programmer to use standard operators while the compiler generates platform specific vectorized code. The primary advantages of this approach over other popular autovectorization schemes [12], is the lack of dependence on new language syntax or special compilers.

In addition to these ideas, which I have already begun work on in [26], I will also conduct a thorough survey of upcoming parallel hardware and their suitability for physics simulation. Additionally, I intend to investigate the possibility of hardware designed specifically for physics simulation. Expected modifications to traditional designs would include changes to cache and memory layout allowing more efficient geometric memory access patterns instead of linear access patterns.

**Evaluation and Challenges** In many ways, the challenges of adapting a simulation engine to a piece of hardware go hand in hand with the evaluations of the same process. Beyond the basic question of how to run a particular algorithm on a particular architectural design, the determination of, and relief from, bottlenecks is a continuous process of diminishing returns. For each hardware specific idiosyncrasy you work around in software to gain the next few percentage points of performance, you do so with the knowledge that the next improvement will be less effective than the last.

For standard CPUs, many tools exist for analyzing performance bottlenecks. Intel provides VTune, a full suite of analysis tools designed to identify instruction latency, cache pressure, and memory bandwidth issues in running code. The availability of these tools makes identifying poorly performing areas in code easier to locate and provides hints for possible solutions.

However, for more exotic architectures, such tools are not readily available. GPU processors, for instance, lack associated tools that provide line by line performance monitoring and instead provide a more global view of performance. This combined with the lack of any form of console output from within an executing GPU kernel, makes debugging GPU code especially tricky. It is one of the advantages Intel's Phi platform holds above GPU programming - as the coprocessor behaves like a completely separate computer, traditional debugging techniques are available and effective.

Vectorization poses its own set of, mostly theoretical, challenges. The main problem with vectorizing any algorithm is the removal of branching behavior. Branching, where code could continue down one of two execution paths, violates the assumption that each channel of the vector is executed by the exact same instruction. A traditional method of eliminating this issue is to execute both sides of the branch unconditionally and then conditionally choose the desired result with a blend operation at the end. While this works in most cases, it also effectively doubles the work the processor has to perform. Addressing this challenge, I intend to reexamine current and proposed algorithms with the goal of not only removing

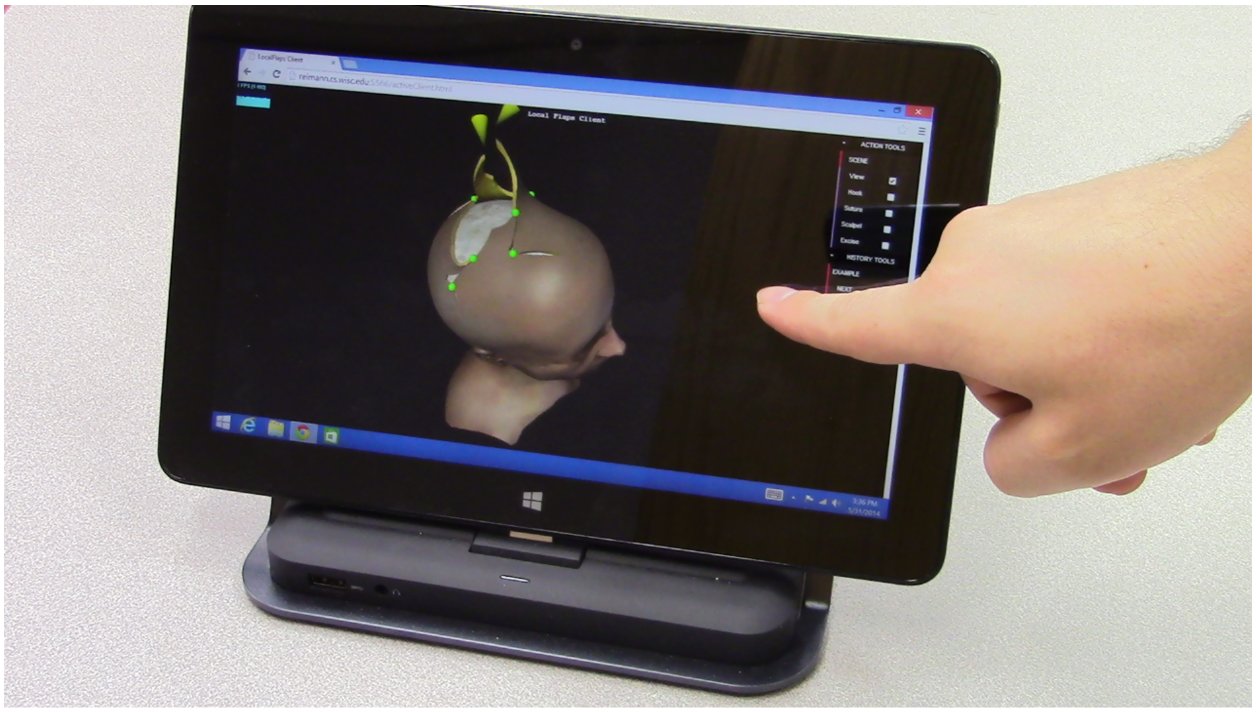

Figure 2: A surgical simulation displayed in real-time on a tablet. The simulation is running on a remote server and the resulting geometry is streamed into the client's web browser.

branching behavior with traditional techniques, but reworking those areas to not require branching in the first place.

## 3.4 Real-time Surgical Simulation as a Service

**Problem Background** Traditionally, computer aided training tools that exist for surgical education have been restricted to bulky and expensive stations. Depending on the type of operation they were designed for, these stations may support multiple users, but often they are designed for a single user. The end result is a system that is expensive to expand or upgrade. Apart from the clear disadvantages of expense, experimentation in training tool design is also disincentivized due to cost.

Moving to a centralized client-server model should help relieve these issues. By placing the majority of computation on a central server, one separates the effort from the interface. This is important for several reasons: First, appropriate interface design for education is a complex and highly volatile area of research. Keeping the interface separate allows for rapid and independent iterative development without affecting the core simulation code. Second, the hardware and software of the server can be upgraded for future performance and feature improvements without requiring the repurchase of every client station. And finally, costs can be driven down by intelligently allocating funds where they are needed. As an example, one could consider a setup employing cheap tablets as clients and an expensive backend server.

**Solution Approach** In the paper I will be submitting [26], I am proposing a three tiered network architecture. The first layer consists of the Client. The responsibilities of client include rendering simulation state and collecting user input. The current design for the client is a web browser interface which uses WebGL to handle rendering. The primary advantage of a browser interface is cross platform compatibility to both traditional desktops and mobile devices. One drawback of this approach is a heavy restriction on client-side processing. As such, this incarnation of the client is designed to be "dumb", performing no work beyond which is required to render the scene and provide a user interface.

The majority of user input processing is handled by the second layer, which I will refer to as the Application Host, or just the host. The host runs on the server and communicates with the Client via a WebSocket connection. This allows the host to push geometry to the client whenever the simulation state updates, instead of depending on the client to poll the host. The primary responsibly of the host is to manage the simulation: loading scenarios, communicating with clients, and converting user interactions into simulation constraints. The host is also responsible for any geometric processing required for handling incision operations and passing the results to the simulation to affect topology changes.

The final layer is the Simulation. The simulation can run in multiple locations. In [26], I discuss how the simulation layer, which I call the Accelerator in the paper, can run on the same server as the Host, or on separate hardware - potentially on a dedicated co-processor card such as an Intel Phi. The simulation layer is the most computationally complex layer as it is responsible for executing the solver and numerical kernels which describe the various material models and constraints.

**Evaluation and Challenges** The primary issues yet to solve with my current approach are mostly due to scale. While I have successfully demonstrated a few clients communicating with a server, the bandwidth requirements become extremely large once the number of clients grows beyond more than four or five. I am currently considering several solutions to this problem. One approach to be considered is compression: working with the assumption that between any two frames, the majority of the model's geometry doesn't change drastically, a compression scheme similar to those used in video streaming could be designed for geometry streaming. Alternatively, one might consider more classic approaches like multicast protocols, but the WebSocket standard does not support this mode.

Another issue to consider is one of lag. Traditionally, lag is classified as the time it takes for user input to sent, processed, and the results returned. While this definition of lag is still quite valid under these circumstances, there is a second form of lag, a simulation lag. Results from user interaction are returned as incrementally improving updates. While faster hardware and algorithmic improvements on the server can improve framerates, resolutions in the middle to high hundreds of thousands of degrees of freedom will likely remain at framerates around 5 to 10 frames per second. As my own investigations have shown in [26], many core accelerators, such as Intel's Phi coproccessor, can provide significant performance but that performance is only realized on extremely large domains - on the order of millions of degrees of freedom. Problems of this scale running on a Phi gain a 3X speed up compared to standard processors, but this still equates to roughly 4 seconds per frame as opposed to 12 seconds per frame on a 6-core Xeon processor.

The challenge then becomes how to fill in the gaps between frames. Ideally one would want the highest resolution possible while remaining interactive, but finer levels of detail can only properly represented by sufficient resolution. One solution might be to simply interpolate between frames in order to give an appearance of smooth motion. A large problem with this idea is the illusion of stiff materials. Since the simulation frames are generated slowly, even rapid material responses, in terms of simulation frame count, will appear sluggish under an interpolation scheme. Buffering may help, but this may be undesirable as well. If interpolation is insufficient, more simulation might be a solution.

While it may seem paradoxical to propose more simulation to fill in the gaps between simulation frames, keep in mind that not all simulations are equal. One idea might be to have multiple levels of simulation at coarser and coarser resolutions. We can build on the convenient fact that the human body is composed of layers - skin on top of fat, on top of muscle, on top of bone. While connected, these layers have different material properties and could conceivably be separate simulations. For instance on the human face, the skin might be simulated at a lower resolution to provide fast response times, but would be layered over a muscle and cartilage layer which would be stiffer and simulated at higher, and therefore slower, resolution. Apart from providing a performance boost, this idea could play nicely with goals from section 3.2, where a scalpel tool could be used to slice the skin layer away from the muscle layer.

Another idea worth considering is a predictive client. Similar in nature to the previous proposed solution, the client platform would provide a local simulation of the object, but tuned for speed instead of accuracy. The idea in this case is for the client simulation to use the previous frame received from the server to set initial conditions and then simulate forward until the next server frame, at which point it would interpolate its state into the "correct" state from the server. This would allow users to interact with a simulated surface with near instant feedback. This approach is heavily utilized in multiplayer games, where user interaction is handled immediately by clients and the behavior of game objects is predicted until the server can confirm the state of system.

The simulation model for a predictive client would need to be lightweight, especially as it may need to be implemented in a web browser. A simple linear elasticity model might suffice here, but building on ideas from [9], it also might be possible to introduce a surface only simulation on the client.

Evaluation for the above techniques should be fairly straightforward. Network performance depends on two values being as low as possible: bandwidth utilized and latency. Hopefully intelligent compression techniques will help reduce total bandwidth requirements and some form of layered simulation will cut down on the perceived latency, if not the actual latency. As I continue to investigate these ideas, I plan to include prototypes into future experimental demos and collect user impressions to determine how much latency is acceptable.

# 4 Related Work

Computer graphics researchers have produced a large number of techniques for model deformation over the past several decades. However, while procedural techniques [16], [40], [18] generally work well for traditional realtime tasks such as character animation, they don't exhibit the necessary realism needed in surgical simulations. Consequently, initial attempts at surgical simulation turned to elastic deformation models [39] that responded more realistically under conditions of probing and cutting [2], [25],[31]. However, these early works were limited in their effectiveness due to computational complexity.

There have been numerous efforts to improve the efficiency and realism of finite element methods, or FEMs, since the initial work done by [39]. Most of this work can be broken down into three major directions that directly apply to surgical applications: Material models, topology alteration, and collisions.

While early work on elastically deformable surgical models used linear elasticity due to its lower computational cost, the need for more realistic materials that handle the extreme deformations possible in soft tissue has clearly been recognized by the community [22]. Fortunately, in recent years several groups have developed efficient solvers for nonlinear material models. [10] used an energy minimization approach to build a fast co-rotational elasticity model, while [13] applied an additional pressure term to support incompressible materials, which are of particular importance in surgical simulations.

In order to accommodate complex organic shapes and detailed cutting operations, many different techniques have been attempted. Early work in fracture was performed by [38], where element connectivity was altered due to exceeding material stress limits. [31] split tetrahedra near cut boundaries and then used vertex snapping to more closely approximate the cut. One major problem with this, and other older subdivision schemes, is the possible creation of poorly conditioned elements leading to poor simulation stability. More recent work, including mine, has concentrated on variations of the virtual node algorithm [27]. [37] showed how this idea could be extended to an arbitrary number of virtual elements, and later some of the same authors demonstrated the technique applied to an embedded surface mesh [34], eliminating the need for a full tetrahedralization of the model. In contrast, [41] avoids the need to maintain well conditioned tetrahedral elements by allowing convex polyhedra as their elements, permitting cuts to divide initial tetrahedrons arbitrarily.

Embedding is a known method to simulate detailed models without requiring a complex simulation mesh [28]. Including this obvious benefit, embedding designs can employ regular lattices allowing for performance optimizations by exploiting the implicit element connectivity and consistent rest shape [43], [33], [24]. One perceived drawback with regular lattices is that they fail to capture small, but important topological details unless the grid resolution is very high. [30] used an octree to coarsen a high resolution lattice and explored non-uniform material properties across elements to approximate missing material. They later expanded this method to support non-manifold embedding lattices to support objects with a branching structure [29]. [14], employing a method similar to my own, uses a finer voxel grid to capture material topology to be embedded in a coarser, non-manifold voxel grid. Finally, [42] demonstrated the use of multiple voxel grid domains to segment a model hierarchically, which they used to simulate plants at interactive rates.

In addition to the many approaches that use classical FEM discretizations, some authors have achieved success with more exotic variations. [15] employed extended FEM methods, where discontinuities are introduced into the element's shape functions, to model cutting. In a similar vein, [17] used discontinuous galerkin FEM formulations. Other authors have dispensed with mesh based discretizations completely, preferring meshless methods [6]. Later they demonstrated a successful surgical simulation including collisions [7].

While much of the previously discussed work is geared towards general elastic body simulation in computer graphics, it is important to cover more surgery specific work. Many surgical simulation projects focus on the mechanical manipulation of organs and other soft internal objects [31], [19]. Even expensive commercial simulators like the Lap Mentor and GI Mentor primarily focus on pushing and cutting simulated internal organs [36], [35]. These types of simulations are so common that several open source frameworks have been built to specifically support further development [1],[3]. These provide easy access to common components like haptic feedback and APIs to connect multiple simulated components. Other surgical simulations are highly specialized, like the work [4] did on interactive needle insertion. However due to the real-time constraints on simulation rate required for virtual surgery, many of these prior works lacked the resolution to resolve realistic behavior.

Producing elastic body simulations that run at realtime speeds is a long time problem in computer graphics and surgery simulation only increases the need. Since volumetric elastic material simulation is inherently computationally expensive, due to its cubic scaling and often a desire for nonlinear materials, most optimization approaches have focused on either data structure or algorithmic design improvements.

A number of authors have focused on parallelism to improve performance. [11] analyzed data flow in their simulations to inform a parallel scheduler for multicore systems. To avoid write hazards during parallel code execution, [20] proposed a system of computation phases with coalesced memory writes, which allowed them to parallelize force computation. [5] developed a parallel version of the Gauss-Seidel algorithm that can run on GPUs.

However, from a software engineering standpoint the transition from even highly parallel CPU code to a GPU implementation is not a trivial task.

However, GPUs present significant development challenges in terms of memory architecture and scheduling that pose a barrier to moving CPU code, even parallel CPU code, onto a GPU. A classic method for optimizing lattice based simulations is to use multigrid methods which are well known in the fluid simulation domains[23]. [24] demonstrated an effective multigrid solver for elastic bodies in their work on character skinning. However, multigrid generally has issues with thin shells of material, common in plastic surgery simulation, and coarsening non-manifold grids is challenging.

## 5 Work to Date

### Elastic Deformers

In this paper [32], my co-authors and myself present a nonlinear elastic body simulator on regular hexahedral lattice discretization. The core contributions of this work include a method for support-

Figure 3: A hollow tube with thin walls being twisted. Left: Sub-voxel accuracy. Right: No sub-voxel accuracy. On the right image, lack of sub-voxel accuracy artificially thickens the tube as all partial cells are treated as if they were full of material.

ing materials with arbitrary amounts of incompressibility, a computationally efficient method for sub-voxel accuracy, a vectorizable second order accurate quadrature scheme, and finally an initial attempt at data organization designed for vectorizing finite element methods.

Support for incompressibility is an important component for biologically accurate materials. As most of soft bodily tissue is composed of water, and water is incompressible, the majority of biological materials found in surgical simulations will be incompressible to some degree. Additionally, depending on the realism required, the incompressiblity of skin and muscle can change due to blood loss from incisions.

In this paper we present a modification of the energy equations to depend on both displacements and a new pressure variable. This change allowed us to deal with a common problem facing incompressible materials - locking. As an incompressible material is compressed, the energy term responsible for the incompressiblity, $M^2(F)$, grows towards infinity. This increased stiffness leads to poor conditioning and slow convergence. By introducing a pressure variable into the energy equation, the resulting problem changes from finding a minimum to finding a saddle point. These saddle points can be shown to overlap with the critical points in the unmodified energy function and are thus a equilibrium configuration. As shown in this paper, the modified equation remains well conditioned even at the incompressible limit and we can solve for the saddle points by using a QMR algorithm.

Sub-voxel accuracy is also important for biological simulations. Thin features are common in naturally occurring objects, including the human body. An example would be the thin muscles within the skin of the scalp. The resolution to fully resolve these features within a single volumetric cell would be impractical.

We provide sub-voxel accuracy in this paper by introducing a second-order accurate quadrature scheme for arbitrary integration domains. We compare our technique to a standard 8-point quadrature method which is second order accurate for hexahedral volumes, but is also expensive to compute and not easily adapted to fractional domains. Our method employs only four points. We achieve second order accuracy by strategically choosing points which match the first and second-order moments of the distribution of points $X$ in the integration domain $\Omega$. We compute the locations of these four points with a Monte-Carlo process as a preprocessing step.

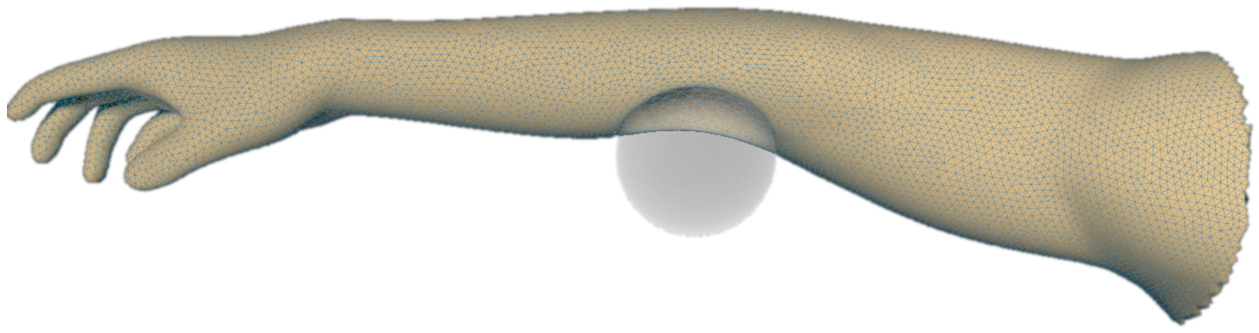

Figure 4: A simulated ball collides with an arm model that uses extrinsically rotated elasticity.

The last two contributions are directly related to the third major focus of my research, exploitation of parallel hardware. In particular, the fourth contribution from this paper eventually lead to the development of the Number library detailed in [26] and Section 3.3. These initial ventures consisted of manually writing x86 vector intrinsics for numerical kernels, which in turn motivated my creation of increasingly more automated vectorization schemes from which the Number library emerged.

## Steklov-Poincare Skinning

This paper [9] presents a method of physics based skinning where only surface degrees of freedom are used to compute deformations. The primary contributions of this paper are: a new material model *extrinsically rotated elasticity* which approximates rotation fields from an underlying skeletal pose, a formulation of quasistatic equilibrium equations which depend only on surface degrees of freedom, and finally a preconditioner method for accelerating convergence of the surface only method. Also demonstrated in this paper is support for collisions while keeping with the surface only model.

While at first glance the ideas presented in this paper may appear to have little to do with surgical simulation, I have already proposed several future work ideas that tie nicely with the concepts proposed in this paper. The most obvious being the surface only quasistatics model. By limiting the degrees of freedom to the surface and utilizing the described preconditioner, it is possible to gain two distinct benefits for my proposed low powered network client. By limiting the simulation to the surface, the paper demonstrated extremely rapid convergence rates, which when combined with collisions, might be suitable for interactive user interfaces. And since we only need the surface degrees of freedom, we don't need to send any volumetric information over the network.

The first contribution is also potentially useful. Because many sites for surgical operations involve tissue close to the underlying bones, it may be possible to use the principles of extrinsically rotated elasticity to present a numerically simpler surface material suitable for high performance and responsiveness to user interactions.

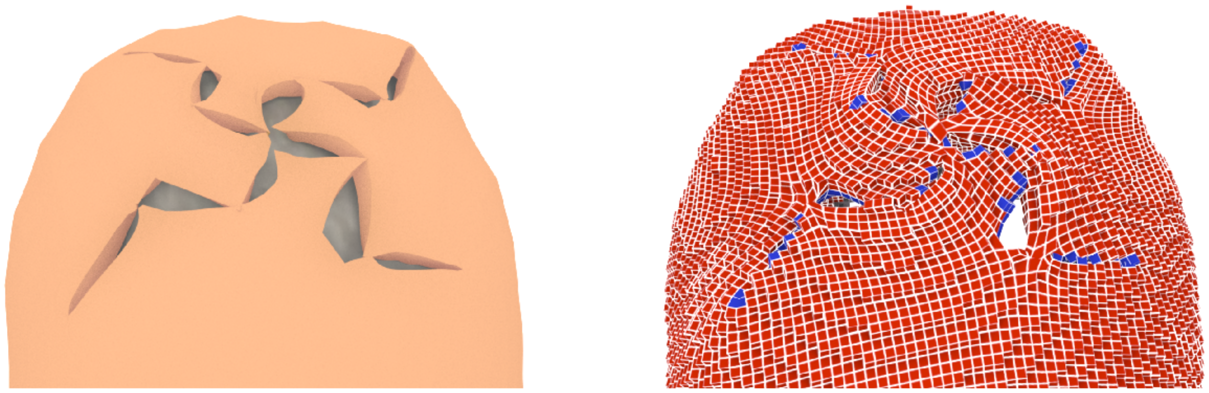

Figure 5: A dufourmental mouly operation, used to close large holes while not causing high amounts of stress, nearing its completion. Shown in red are the grid cells, while mesh cells are shown in blue.

## GRIDiron: Plastic Surgery Simulation

This paper has not yet been submitted to ACM Transactions on Graphics, but is planned to be submitted at the end of Summer 2014 or early Fall. The primary goals of this paper were to lay the groundwork for all future research I will be performing into plastic surgery simulation. The major contributions of this work include the following: a hybrid lattice-mesh discretization capable of resolving sub-voxel nonmanifold features such as thin incisions, a object-oriented autovectorization library capable of mapping numerical kernels onto SIMD style hardware, and a multilayer deployment strategy which separates the user interface and numerical processing as a client-server network architecture.

As discussed in Section 3.2, topology change is required for surgical operations which involve making incisions. The hybrid lattice-mesh technique presented in this paper augments the regular lattice grid used in [32] with an explicit mesh data structure. There are mesh forms of both nodes and cells, where the rule states that all grid cells contain only grid nodes. Therefore, mesh cells implicitly contain a mix of mesh and grid nodes. The explicit nature of the mesh allows the discretization to encode non-manifold features and gaps smaller than the cell diameter. Detectable features are determined by a modification of techniques shown in [37] but for regular hexahedra.

As part of this paper, I discuss the Number library - the autovectorization scheme developed for accelerating regular lattice deformers. Number consists of two components. The first is a C++ template library which abstracts vector operations under a data type which appears scalar externally. The intent of this design was to provide a straightforward transition between a scalar implementation of an algorithm and a vectorized version, all while remaining human readable. This last point is important as the second component of Number is a collection of numerical kernels which implement the nodal force equations for several material models. By grouping cells into geometric blocks, these kernels can be executed in parallel, each computing the forces for a single block, which in turn simultaneously compute between four and sixteen cells depending on available hardware.

Finally, as discussed in section 3.4, this paper details the my initial forays into simulation as a network service. Section 3.4 describes the general layout, but in this section I will cover more details about this implementation. The GRIDiron presents a four layer architecture, although conceptually it follows the three layer design I discussed previously. The lowest layer is a javascript client that runs in a web browser and presents a WebGL visualization of the simulation. Currently, this frontend only supports visualization and does not designed to handle user interactions through tools. Currently all tools are handled by a second level client, with which the web client communicates to receive scene data. This level is a hybrid between the Client and Host classifications from section 3.4. It implements the behavior of several surgical tools, including an incision tool which performs destructive mesh alterations to create cuts. The second and third layers are typically compiled together into one executable, but an optional RPC interface exists allowing these layers to run on separate platforms.

The third and fourth layers are primarily concerned with simulation. The third layer is the externally facing part of the simulation. It is responsible for building the hybrid mesh-lattice discretization from provided geometry and managing the less performance sensitive parts of the simulation, like updating embedded surfaces. The fourth layer, corresponding to the Simulation in section 3.4, contains the solver and state information of the system. This component can also be compiled together with the second and third layers, but also supports an MPI interface allowing it to execute independently on an accelerator card, such as Intel's Phi.

## 6 Research Plan

In this section, I will lay out a tentative schedule for my research, with the intent to finish my dissertation by the end of two years.

### 6.1 Complete and Submit GRIDiron paper

**Estimated Completion** September-October, 2014

**Details** The paper is mostly complete. All relevant data has been gathered and all that remains is to polish the current draft into a submittable document.

### 6.2 Complete a Series of Experimental Demos

**Estimated Completion** Ongoing (2014-2016)

**Details** I will conduct a series of demos to present the software I have developed to the plastic surgery community on campus. The purpose of these demos is multifaceted.

- Promote the research and build contacts with the medical community.
- Evaluate what techniques I have already implemented work and don't work in terms of training potential.

- Gather feedback and desired features from the medical community, which will help focus my research into productive directions.

The first demo is scheduled for September 12th, 2014, several days after my scheduled preliminary exam. This demo will be critical in determining my direction later this year and early into the spring of next year. Subsequent demos have yet to be scheduled, but I expect the success of the first will help build enthusiasm for future trials.

## 6.3 Continue Developing the Web Client into the Primary Client

**Estimated Completion** Winter, 2014

**Details** The current web client provides several major advantages over the older C++ client.

- Platform Independence - Capable of running on all major platforms, including mobile devices, due to its running within a web browser.
- Network Centric Development - Requiring network connectivity be treated as a baseline requirement forces me to remain honest and never ignore how future changes might affect network performance.
- Rapid Development - Developing user interfaces in javascript allows for a fast development pace and rapid iteration on features.
- Constrained Environment - Similar to the second benefit, running within a web browser will force me to keep the client's resource usage as low as possible, keeping the client friendly for low powered, cheap terminals.

As it currently stands, the web client lacks many features supported by the C++ client, namely a functional tool interface and scene loading capabilities. These will need to be added before the C++ client can be retired.

## 6.4 Feature Exploration

**Estimated Completion** Spring 2016

**Details** With a functional platform and at least one demo completed, I will be able to explore more specific features. Figure 6 displays the space of topics that fall under possible consideration. Below I will talk about several I personally find more important, but as I explained earlier the experimental demos will play a large role in determining which features are truly important to the medical community. As such, the order in which these features will be approached may vary so I have decided to group them together.

**Continuous Collisions** Support for the expected mode of deformable objects is to be colliding instead of being free of collisions. This is important as during a surgery, it can be assumed that tissue will be colliding with other tissue unless acted upon by a surgeon. As an example, consider an incision. When cut, skin will naturally fall back together, each side of the

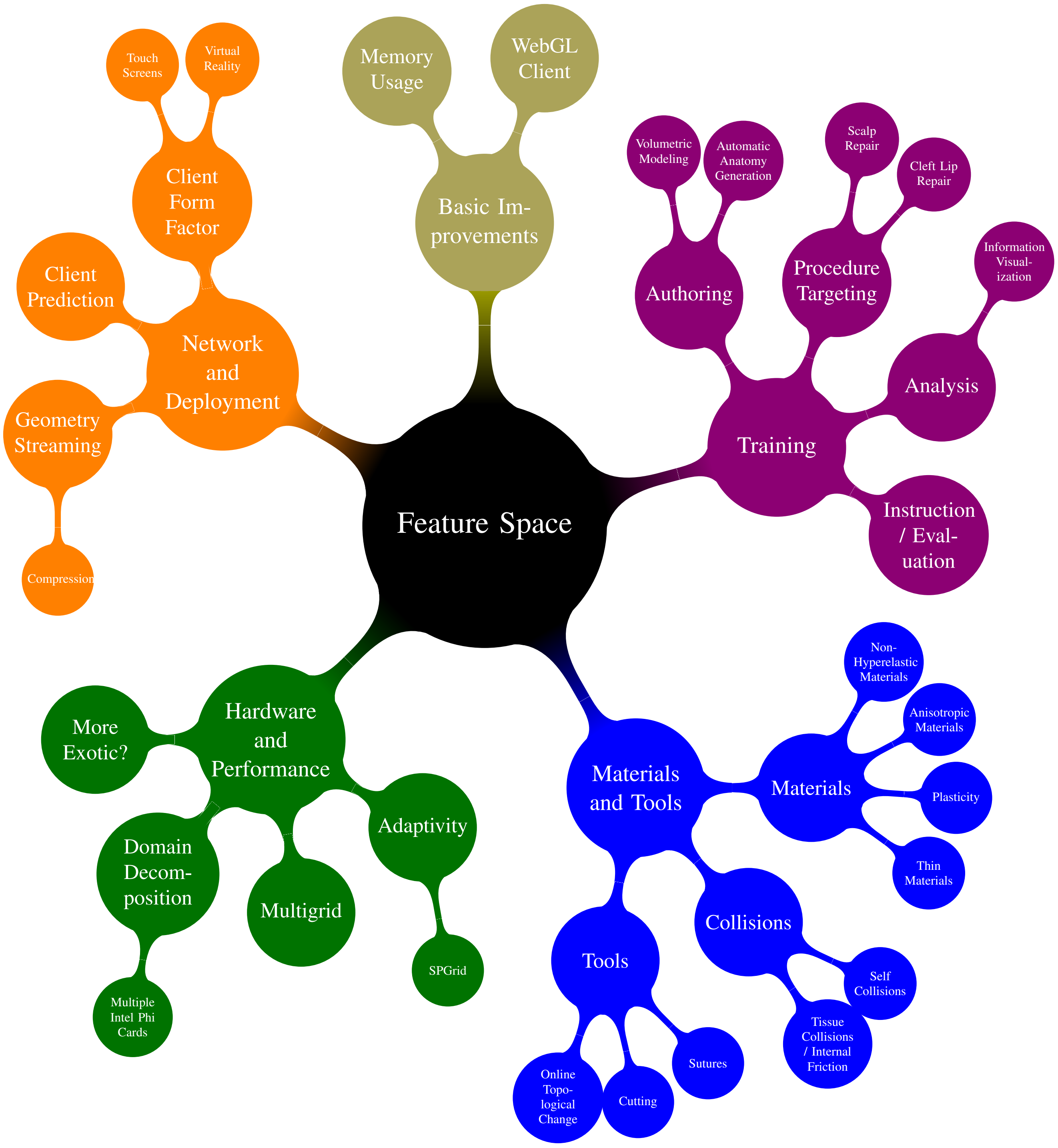

Figure 6: Potential future directions for research.

incision resting against the other. My current model, where an incision creates a wide empty gap, is woefully unrealistic.

**Online Topology Change** Support for altering topology without stopping or losing current simulation state. This is important as it is unrealistic to assume all incisions must be performed before moving any flesh.

**Realistic Materials** Support for more realistic material behaviors. This may include bi-phasic behaviors, plasticity effects, and additional anisotropic materials. This feature may require physical measurements to be conducted from real tissue samples.

**Visualization** Support for the client to visualize important non-visual properties of materials. An example could be stress patterns, but consultation with the medical community might reveal more information that would be valuable to visualize.

## 6.5 Dissertation Writing

**Estimated Completion** Fall, 2016

**Details** At this point I should have implemented and evaluated three to four major features. I expect to begin writing at the end of spring and finish by the beginning of fall. I should be prepared to defend before that winter.

# References

[1] J. Allard, S. Cotin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette, L. Grisoni, et al. Sofa - an open source framework for medical simulation. In *Medicine Meets Virtual Reality, MMVR 15*. IOS Press, 2007.

[2] M. Bro-nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer Graphics Forum*, pages 57–66, 1996.

[3] M. C. Cavusoglu, T. G. Goktekin, and F. Tendick. Gipsi: A framework for open source/open architecture software development for organ-level surgical simulation. *Information Technology in Biomedicine, IEEE Transactions on*, 10(2):312–322, 2006.

[4] N. Chentanez, R. Alterovitz, D. Ritchie, L. Cho, K. K. Hauser, K. Goldberg, J. R. Shewchuk, and J. F. O'Brien. Interactive simulation of surgical needle insertion and steering. *ACM Trans. Graph.*, 28(3):88:1–88:10, July 2009.

[5] H. Courtecuisse and J. Allard. Parallel dense gauss-seidel algorithm on many-core processors. In *High Performance Computation Conference (HPCC)*, pages 139–147. IEEE CS Press, June 2009.

[6] S. De and K. Bathe. The method of finite spheres. *Computational Mechanics*, 25:329–345, 2000.

[7] S. De, J. Kim, Y.-J. Lim, and M. A. Srinivasan. The point collocation-based method of finite spheres (pcmfs) for real time surgery simulation. *Computers & Structures*, 83(1718):1515 – 1525, 2005. Advances in Meshfree Methods.

[8] A. G. Gallagher, E. M. Ritter, H. Champion, G. Higgins, M. P. Fried, G. Moses, C. D. Smith, and R. M. Satava. Virtual reality simulation for the operating room: Proficiency-based training as a paradigm shift in surgical skills training. *Annals of Surgery*, 241(2), 2005.

[9] M. Gao, N. Mitchell, and E. Sifakis. Steklov-Poincar Skinning. In *Eurographics / ACM SIGGRAPH Symposium on Computer Animation*, pages 139–148, 2014.

[10] J. Georgii and R. Westermann. Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*, 2008.

[11] E. Hermann, B. Raffin, and F. Faure. Interactive physical simulation on multicore architectures. In *Proceedings of the 9th Eurographics Conference on Parallel Graphics and Visualization*, EG PGV'09, pages 1–8, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[12] Intel Corporation. ISPC: intel spmd program compiler. `http://ispc.github.io/`, 2011–2014.

[13] G. Irving, C. Schroeder, and R. Fedkiw. Volume conserving finite element simulations of deformable models. *ACM Transactions on Graphics (SIGGRAPH Proc.)*, 26(3), 2007.

[14] L. Jerabkova, G. Bousquet, S. Barbier, F. Faure, and J. Allard. Volumetric modeling and interactive cutting of deformable bodies. *Progress in Biophysics and Molecular Biology*, 103(2-3):217–224, Dec. 2010. Special Issue on Biomechanical Modelling of Soft Tissue Motion.

[15] L. Jeřábková and T. Kuhlen. Stable cutting of deformable objects in virtual environments using xfem. *IEEE Comput. Graph. Appl.*, 29(2):61–71, Mar. 2009.

[16] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26(3), July 2007.

[17] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. Flexible simulation of deformable models using discontinuous galerkin fem. *Graph. Models*, 71(4):153–167, July 2009.

[18] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27(4):105:1–105:23, Nov. 2008.

[19] J. Kim, C. Choi, S. De, and M. A. Srinivasan. Virtual surgery simulation for medical training using multi-resolution organ models. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 3(2):149–158, 2007.

[20] J. Kim and N. S. Pollard. Fast simulation of skeleton-driven deformable body characters. *ACM Trans. Graph.*, 30(5):121:1–121:19, Oct. 2011.

[21] A. Lindblad and G. Turkiyyah. A physically-based framework for real-time haptic cutting and interaction with 3d continuum models. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, pages 421–429, New York, NY, USA, 2007. ACM.

[22] M. Marchal, J. Allard, C. Duriez, and S. Cotin. Towards a framework for assessing deformable models in medical simulation. In P. J. E. Fernando Bello, editor, *ISBMS '08 - International Symposium on Biomedical Simulation - 2008*, volume 5104 of *Lecture Notes in Computer Science*, pages 176–184. Springer Berlin Heidelberg, 2008.

[23] A. McAdams, E. Sifakis, and J. Teran. A parallel multigrid poisson solver for fluids simulation on large grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 65–74, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[24] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.*, 30(4):37:1–37:12, July 2011.

[25] C. Mendoza and C. Laugier. Simulating soft tissue cutting using finite element models. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 1109–1114. IEEE, 2003.

[26] N. Mitchell, C. Cutting, and E. Sifakis. Gridiron: An interactive authoring and cognitive training foundation for reconstructive plastic surgery procedures. To be submitted to ACM Trans. Graph., preprint.

[27] N. Molino, Z. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph.*, 23(3):385–392, Aug. 2004.

[28] M. Müller, M. Teschner, and M. Gross. Physically-based simulation of objects represented by surface meshes. In *Proc. Computer Graphics International*, pages 156–165, June 2004.

[29] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph.*, 28(3):52:1–52:9, July 2009.

[30] M. Nesme, Y. Payan, and F. Faure. Animating shapes at arbitrary resolution with non-uniform stiffness. In *Eurographics Workshop in Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, Madrid, nov 2006. Eurographics.

[31] H.-W. Nienhuys and A. F. van der Stappen. A surgery simulation supporting cuts and finite element deformation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001*, pages 145–152. Springer, 2001.

[32] T. Patterson, N. Mitchell, and E. Sifakis. Simulation of complex nonlinear elastic bodies using lattice deformers. *ACM Trans. Graph.*, 31(6):197:1–197:10, Nov. 2012.

[33] A. Rivers and D. James. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Transactions on Graphics (SIGGRAPH Proc.)*, 26(3), 2007.

[34] E. Sifakis, K. G. Der, and R. Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 73–80, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[35] Simbionix USA Corporation. gastrointestinal simulator - gi mentor simbionix. `http://simbionix.com/simulators/gi-bronch-gi-mentor`, 2002–2014.

[36] Simbionix USA Corporation. Laparoscopic simulator - lap mentor simbionix. `http://simbionix.com/simulators/lap-mentor`, 2002–2014.

[37] J. Teran, E. Sifakis, S. S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *Visualization and Computer Graphics, IEEE Transactions on*, 11(3):317–328, 2005.

[38] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. *SIGGRAPH Comput. Graph.*, 22(4):269–278, June 1988.

[39] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *SIGGRAPH Comput. Graph.*, 21(4):205–214, Aug. 1987.

[40] X. C. Wang and C. Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, pages 129–138, New York, NY, USA, 2002. ACM.

[41] M. Wicke, M. Botsch, and M. Gross. A finite element method on convex polyhedra. In *Computer Graphics Forum*, volume 26, pages 355–364. Wiley Online Library, 2007.

[42] Y. Zhao and J. Barbič. Interactive authoring of simulation-ready plants. *ACM Trans. on Graphics (SIGGRAPH 2013)*, 32(4):84:1–84:12, 2013.

[43] Y. Zhu, E. Sifakis, J. Teran, and A. Brandt. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph.*, 29(2):16:1–16:18, Apr. 2010.