Fall 2014

Digital human modeling for optimal body armor design

Nic Andrew Capdevila
University of Iowa

DIGITAL HUMAN MODELING FOR OPTIMAL BODY ARMOR DESIGN

by

Nic Andrew Capdevila

A thesis submitted in partial fulfillment
of the requirements for the Master of
Science degree in Electrical and Computer
Engineering in the Graduate College of
The University of Iowa

December 2014

Thesis Supervisor: Research Scientist Timothy Marler

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the Master's thesis of

Nic Andrew Capdevila

has been approved by the Examining Committee
for the thesis requirement for the Master of Science degree in
Electrical and Computer Engineering at the December 2014 graduation.

Thesis Committee: _____
Timothy Marler Thesis Supervisor

_____
Karim Abdel-Malek

_____
Jasbir Arora

_____
Jon Kuhl

To my parents, my family, my friends, and all of those who have guided and inspired me along the way. I would not be where I am without you.

ACKNOWLEDGMENTS

The work presented in this thesis, as well as the thesis itself, would not be possible without Dr. Timothy Marler, whose tireless effort, patience, guidance, knowledge, helpfulness, encouragement, understanding, and attitude will act as an inspiration to me always. I cannot thank him enough for being a wonderful teacher and person, and my respect for him knows no bounds. I would like to thank my academic advisor, Karim Abdel-Malek, for his time and support. I would like to thank all of my committee members for their valuable time, as well as their feedback. My deepest gratitude and affection to everyone at the Virtual Soldier Research program, especially Anith Mathai, Kimberly Farrell, Andy Taylor, Rich Degenhardt, and Jake Kersten, for all of their help along the way. Lastly, I can't thank my parents enough for granting me with the "Scholarship For Men Who Aren't Ready To Leave School," as well as for the quite literally endless love, pride, confidence and support.

This work was funded by the Office of Naval Research (ONR).

ABSTRACT

In order to leverage advances made in body-armor materials, as well as to further the design landscape, considering body armor as a complete human-centric system is becoming more prevalent. This trend necessitates a greater focus on human systems integration (HSI) and human-centric design. Digital human models (DHMs) provide a powerful tool for HSI, but modeling-and-simulation tools, let alone DHMs, have rarely been used with body armor. With respect to analysis, this is changing. New methods for evaluating body armor from a biomechanical perspective have been developed within the Santos™ DHM. It is now possible to import digital models of body-armor systems, place them on an avatar, simulate various tasks (i.e., running, aiming, etc.), and then virtually evaluate the armor's effect on performance, balance, mobility, bulk, etc. However, with respect to design, there are no available simulation tools to help users balance the goals of maximizing mobility and survivability concurrently.

In response to these growing needs, there are two new areas of work being proposed and discussed. First, this work leverages a series of new virtual evaluation capabilities for Personal Protective Equipment (PPE) and implements a filter that automatically evaluates and selects from a library of designs the most advantageous PPE system based on user-selected objectives and constraints. Initial tests have shown realistic results with minimal computational demand.

Secondly, this thesis proposes a new method for armor-system topology optimization that optimizes not only biomechanical metrics but also external (to the DHM system) metrics from potentially complex injury and protection models. The design variables for this optimization problem represent the position on the body of small body-armor elements. In addition, the existence of each element is modeled as a variable, such that unnecessary elements are determined and removed automatically. This inclusion of location in combination with the traditional existence variable is a novel inclusion to the topology optimization method. Constraints require that no two elements overlap. The objective functions that govern where the armor elements are moved must be general enough to function with any external data, such as survivability. Thus, a novel process has been developed for importing external data points (i.e., stress at points in the body resulting from a blast simulation) and using regression analysis to represent these points analytically. Then, by using sequential quadratic programming for gradient-based optimization, the armor elements are automatically positioned in order to optimize the objective function (i.e., minimize potential injury). This new approach allows any metric to be used in order to determine general body-armor concepts upstream in the design process. This system has the potential to become especially useful when trying to optimize multiple objectives simultaneously, the results of which are not necessarily intuitive. Thus, given a specified amount of material, one can determine where to place it in order to, for example, maximize mobility, maximize survivability, and maximize balance during a series of specified mission-critical tasks. The intent is not necessarily to provide a final design with one “click”; accurately considering all aspects of hard and soft

armor is beyond the scope of this work. However, these methods work towards providing a design aid to help steer system concepts.

Test cases have been successfully run to maximize coverage of specific external data for internal organs (and thus survivability) and mobility, while minimizing weight. The weight metric has also been successfully used as a constraint in the optimal armor design. In summary, this work provides 1) initial steps towards an automated design tool for body armor, 2) a means for integrating different analysis models, and 3) a unique example of human-in-the-loop analysis and optimization.

PUBLIC ABSTRACT

In order to leverage advances made in body-armor materials, as well as to further the design landscape, considering body armor as a complete human-centric system is becoming more prevalent. This trend necessitates a greater focus on human systems integration (HSI) and human-centric design. Digital human models (DHMs) provide a powerful tool for HSI, but these modeling-and-simulation tools, have rarely been used with body armor. Currently, with respect to design, there are no available simulation tools to help users balance the goals of maximizing mobility and survivability concurrently.

In response to these growing needs, two approaches are discussed. First, this work leverages a series of new virtual evaluation capabilities for Personal Protective Equipment (PPE) and implements a filter that automatically evaluates and selects from a library of designs, the most advantageous PPE system based on user-selected objectives and constraints.

Secondly, this thesis proposes a new method for armor-system design optimization that optimizes not only biomechanical metrics but also external (to the DHM system) metrics from potentially complex injury and protection models. This new approach allows any metric to be used in order to determine general body-armor concepts upstream in the design process. This system has the potential to become especially useful when trying to optimize multiple objectives simultaneously, the results of which are not necessarily intuitive. Thus, given a specified amount of material, one can determine where to place it in order to, for example, maximize mobility, maximize survivability, and maximize balance during a series of specified mission-critical tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

There are essentially two major problems that will be addressed in this thesis. The first problem is deciding which armor system is optimal for a specific warfighter and set of tasks, based on user-defined metrics. This will be done by running a series of calculations, normalizing metrics, and supplying the user with the optimal armor system as well as the rating of the system. The second problem to be addressed is the design of an optimal armor system, based off any number of external data sources. This problem will be formulated by automatically determining the location and existence of small armor components on an avatar. There are two major difficulties to overcome in order to perform optimal armor system design. The first problem lies in forming a smooth, continuous objective function that is bounded along the surface of Santos. The second problem is to transform the available metrics within Santos in order to provide the basis for the objective function.

Personal protective equipment (PPE) is a critical component for Warfighter survivability and performance, and designing a suitable PPE system is a complex and time-consuming task with multiple design variables and constraints. To date most effort with PPE design

has centered on materials development in response to necessary blast and ballistics requirements. Recently, however, the design focus has shifted towards the complete body-armor system, not just the material(s) of a single plate or component. Improvement in survivability, stemming from incremental changes in material properties, can be minimal, and increased mobility can often be more advantageous than slightly stronger or lighter materials. Consequently, the field of human factors is starting to play an increasingly important role. Accordingly, digital human modeling (DHM) offers significant opportunities for improved body-armor system design with potential reduction in developmental cost and timing. This thesis provides a method to take advantage of DHM capabilities in order to design measurably improved armor systems, as well as to select the optimal armor system for the right task and warfighter. The armor systems designs are improved by calculating values for metrics considered important for optimal design, such as weight, coverage, mobility, etc. Design of the armor systems will be performed by using continuous optimization by determining the location and existence of armor components on an avatar. There are several challenges, such as:

1. Defining design objectives
2. Formulating a continuous optimization problem
3. Creating a system that allows for any new information to be processed and used as an objective function

These difficulties arise from the lack of easily bounded, continuous, and smooth variables to use to determine the location of the armor components on the avatar. Therefore, a transformation of variables is used in order to overcome this problem. This approach provides a method for systematically improving armor system design in a field that currently lacks any such tools.

Another problem is the lack of currently available design and evaluation tools for PPE. Often, the design of PPE can be task specific. For instance, there are situations in which the armor must be as lightweight as possible, thus sacrificing coverage. Other scenarios require increased coverage, which may lead to reduced range of motion. Previously, these design decisions were handled by experimental trial and error. By leveraging the modeling and simulation capabilities of DHM software, however, one would be able to run a great number of tasks, modify and test new designs, and mathematically measure the effects of the design with regards to human factors. Thus, new capabilities have been developed for virtually importing body armor and parenting it to the Santos DHM. Santos can then execute a variety of tasks like reaching, aiming, walking, etc. By leveraging the capabilities within the DHM software, new information regarding the effects of the armor system with respect to human factors can be made available and therefore be used to improve the selection and design of armor systems.

While there are tools that exist to evaluate armor within the Santos DHM that is used, these tools required methods be developed to leverage their capabilities for design and optimal selection. Concurrently, the virtual armor is automatically tested with respect to

weight, coverage, bulk, geometric encumbrance, range of motion, joint torque, balance, and performance. This allows one to compare various PPE systems from a human-factors perspective. This method of optimal design has not been attempted with regards to armor systems. This new capability of quantifying the armor systems effects allows for the methods laid out in this thesis, which include 1) the down-selection of the optimal armor system based on user-defined criteria and tasks, and 2) the ability to optimally design new armor systems based on user-defined metrics or third-party data. Both of these processes have previously been performed manually and without thorough and measurable scientific merit. While it is true that some problems, such as finding the maximum amount of weight on a warfighter, can be done fairly quickly and intuitively with manual testing, this is simply not the case for more complicated and, in reality, meaningful design considerations. For example, armor coverage does not necessarily equate to survivability, as some areas are much more important to vitality. Mobility is also a serious concern, as it's not only important to ensure that the warfighter is capable of completing both routine and emergency tasks, but also to consider the effects of long-term use. Specifically, the biomechanical effect and propensity of injury due to the loads that the warfighter carries are of specific concern (Knapik, Reynolds, & Harman, 2004). If one were trying to design an armor system that allowed for maximum mobility and survivability, specifically including multiple sets of statistical survivability data while trying to minimize the weight of the armor system, the problem becomes too difficult to be done quickly and correctly by trial and error. It is for these complex systems that DHM and armor system design optimization is being proposed.

Currently, there is no quantifiably correct method for selecting an optimal armor system for a given warfighter and task. This thesis proposes leveraging these new evaluation capabilities and implementing an optimization-based filter that automatically and virtually evaluates and selects the most advantageous PPE system. First, a library of employable, applicable armor systems is compiled. The user then indicates which metrics are used as objectives (i.e., minimize weight) and which metrics are used as constraints (i.e., ensure that weight is no more than twenty pounds). Then, the user sets up a task or set of tasks during which the armor is evaluated. Finally, the optimization filer automatically selects the system that optimizes the specified objectives while satisfying the specified constraints. As new and successful armor systems are found, they can then be added to the library filtering system, which can provide optimal designs more quickly. Initial tests have shown realistic results with minimal computational demands.

Secondly, work involves approaching the problem of armor design from a dynamic optimization standpoint by actually creating new armor systems on the fly. Using metrics made available within the Santos software, armor components are automatically created or deleted and moved over Santos's surface. Eventually, higher-fidelity injury models pertaining to damage of internal organs will be used to provide additional objective functions and constraints. The proposed capabilities can be used not just with body armor, but with other types of equipment as well, and thus pave the way for automatic human-centric optimal design.

As a very brief overview of how the problem was approached, the system must:

1) Make use of any internal or third-party data in order to formulate the objective function
2) Determine the set of design variables that includes the number of armor pieces as well as the location for each armor piece, verifying that all design variables abide by any constraints
3) Determine the associated objective function value for each armor piece. The value is determined by the objective function, which is formulated based on the metric or metrics being considered
4) Repeat the process until an optimal solution is found, and place armor components in determined positions based on the objective functions design variables

1.2 Literature Review

The general concept of body armor design and continuous design optimization has been studied before. There are also areas of research that can be noted for their similarities and therefore be learned from and possibly applied to the problem at hand. This section includes an overview into the state of the art of armor design as well as human systems integration, specifically noting the gap between the two that this thesis attempts to address. There is also discussion on multi-level optimization, facility location problems, and topology optimization. These three topics have attributes similar to the optimal

armor system design problem and were therefore researched in order to determine if any techniques could be applied. There is also a section discussing collision detection, which needed to be researched in order to understand different possible methods of preventing overlap between armor components in the system.

The specific application of armor system design under consideration here has been severely under-researched. While there has been a great deal of study of the materials used to develop the armor systems, there has been very little actual research into the proper placement when considering human factors. By performing the proper studies and thoroughly looking into the effects of armor size and placement on the body, it may be possible to provide a more complete, comfortable, and protective armor system without restricting the user.

1.2.1 Armor Design

The bulk of armor design research has been done on the subject of material design. These studies look into the attributes of the material, including weight, flexibility, and durability, as well as how it performs under impact. There is a myriad of these materials and studies, as well as studies the focus on the correct performance metrics in order to properly evaluate the materials (Kaliraj, Narayanasamy, Rajkumar, Mohaideen, & Manickam, 2014), (Mahbub, Wang, & Arnold, 2014), (Valença, Griza, Oliveira,

Sussuchi, & Cunha, 2014), (Lou, Hsing, Hsieh, & Lin, 2013), (Li, Wang, Lou, & Lin, 2013). These material metrics focus on a range of topics, including life cycle, ballistic testing, damage tolerance, repair, through-thickness reinforcement, energy dissipation, and rate-dependent failure mechanisms, to name a few (Gama, et al., 2001). There are studies that seek to use these and/or other similar metrics in order to develop new materials that can be used in place of previous materials as body armor. Developing a lightweight ceramic-metal based armor is based on the pioneering work of Wilkins et al. (Wilkins, Cline, & Honodel, Fourth Progress Report Of Light Armour Programs, 1969), who also later studied the penetration mechanism of such armor (Wilkins, Cline, & Honodel, Mechanics Of Penetration and Perforation, 1978) (Wilkins, Cline, & Honodel, Computer Simulation of Penetration Phenomeon, 1980). Evaluation of PPE with regards to human factors and ergonomics is a subject less studied, but some work has been done (Hennessy & Zielinski, 2006), and a good deal of this work focuses on soft armor evaluation (Goldfarb, Ciurej, Weinstein, & Metker, 1975), (Gu, 2006), (Metker, Prather, Coon, Swann, & Hopkins, 1978).

In the realm of PPE, there is substantial room for growth in terms of the methods used for simulating, evaluating, and selecting armor systems. Studies have shown that the load that warfighters are required to carry can leave them with long-term injury (Knapik, Reynolds, & Harman, 2004). With this in mind, it becomes clear that every piece of equipment that the warfighter is required to wear should be designed optimally in order to be as effective and useful as possible.

With respect to modeling and simulation of body armor, there has been little, if any, work conducted in the context of a complete human model. There have been some experimental studies of the biomechanical and physiological effects of body armor with respect to additional loads during tasks like walking and aiming (Harman, Frykman, Pandorf, Tharion, & Mello, 1999), (Hasselquist, Bensel, Corner, Gregorczyk, & Schiffman, 2008). Other than these efforts, although there has been extensive work involving ballistic and blast analysis of body armor directly, there is little work that accounts for the interaction between body armor and human performance, especially with regard to computational models. Work has also been done to provide the first automated system for evaluating PPE from an HSI perspective, and this work leverages the underlying platform (Mathai, et al., 2010) (Marler, Mathai, Johnson, & Taylor, 2012).

1.2.2 Human Systems Integration

One of the important aspects of this work is that the optimization of the armor systems is done with a focus on the human element. Problems that focus specifically on human interaction are often in the category of Human Systems Integration (HSI). The objectives of HSI are to have a positive and significant effect on the relationship between the production and operations of systems and equipment and human interaction (Booher, 2003). This category of problems tends to focus primarily on how humans perform tasks and the optimal way to design the surrounding system in order to allow humans to

perform the task both comfortably and efficiently. A typical area of study is that of ingress/egress from vehicles (Chateauroux & Xuguang, 2010), (Gwynne & Kuligowski, 2010) . Here the digital human model (DHM) is used to simulate human motion in order to design a vehicle in such a way that it is easiest to enter and exit. This is the standard use of the HSI, and its study of how a human interacts with a static, or discrete, number of systems or environments. The novelty in the methods and approaches laid out in this thesis is found in the fact that the DHM is used to automatically optimize the system around it.

An analog can be drawn to the problem of designing the optimal armor system. However, there is a subtle difference in how one must think of the interaction between the system and the human. In the armor system design, it is not a direct human-to-system interaction, but rather an interaction caused as a byproduct of the human motion while an armor system is being worn. A similar problem is introduced in HSI with regards to clothing worn in hazardous environments (Adams, Slocum, & Keyserling, 1994), (Murray, Simon, & Sheng, 2011), (Li, et al., 2013). While these studies are important and informative, they differ from the armor system design problem in both their uses and complexity. In the armor system design problem, there are different variables, such as armor component size, location, and existence, as well as more metrics, such as penetration, survivability, coverage, etc.

1.2.3 Multi-Level Optimization

In the armor design optimization problem at hand, a number of different divisions will be investigated in order to find the optimal set of sub-divisions for the problem. When this problem was first being formulated, it was unclear whether these sub-divisions would be made per joint, per area of the body, or per armor piece. It was also considered that different objective function metric (i.e., survivability, mobility, etc.) be evaluated separately and returned to an upper level that manages each sub-objective function in order to find an overall optimal solution. The topic that discusses these types of approaches is known as multi-level optimization. There is no literature that uses this approach with regards to designing optimal human systems integration.

Multi-level optimization is an approach that allows for a system of optimization sub-problems that are solved independently in order to optimize a higher-level problem. Using decomposition techniques, the problem is separated hierarchically into two levels. The goal of the second level is to coordinate the actions of the first-level sub-problems. The first level works as an optimization level, using an algorithm to solve the sub-problems, each of which, when solved independently provide an overall optimum to the original problem. There are many different interconnection forms of the subsystems, but the hierarchical form is the most common. In essence, one can look at multi-level optimization as a systematic approach to a technique of solving problems known as "divide and conquer." Multi-level optimization works under the assumption that, by

finding the local minimum to a set of smaller, easy-to-solve sub-problems, we will tend towards the overall globally minimal and therefore optimal solution.

Multi-level optimization using decomposition is a method that is applicable in many different fields and cases, but is especially prevalent in the process of network optimization. Layered architectures form one of the most fundamental structures of network design. In network design, each layer controls a subset of decision variables and observes a subset of constant parameters and variables from the other layers. The layers work to support each other by hiding the complexity of the layer below it, in order to provide a service to the layer above.

In most optimization problems, there is often more than one way to divide and conquer. It is also safe to assume that some layering structures are more efficient than others. Examining the different possible choices of modularized design leads to the study of "how to" and "how not to" structure the layers of the optimization problem. Although the general principle of layering is widely recognized as an extremely effective and efficient method in certain types of optimization problems, there is little quantitative understanding to guide a systematic, rather than ad hoc, process of designing the layers. A large portion of the focus in the area pertains specifically to applying proper layering in computer network architectures. A short list of more formal approaches can be found to address this issue in papers such as *Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures* (Chang, Low, Calderbank, & Doyle, 2007), *On Two Level Optimization* (Bialas & Karwan, 1982), as well as books such as

Multilevel Optimization: Algorithms and Applications (Migdalas, Pardalos, & Värbrand, 1998).

1.2.4 Facility Location Problems

On a basic level, the armor placement optimization problem is analogous to the set of problems known as facility location optimization problems. These problems have been covered by a wide range of literature (Drezner, Klamroth, Schobe, & Wesolowsky, 2002) (Francis, Jr., & White, 1991) (Chan, 2011). The general facility location problem is concerned with the determination of the optimal number, size, and geographic configuration of facilities, in a manner that continuously optimizes a certain set of criteria (Moujahed, Simonin, & Koukam, 2009). The difference between the typical facility location problem and the optimal armor system design problem is that instead of changing the number, size, and geographic configuration of, say, a fire station in order to maximize the number of homes that it could reach in a minimum amount of time, similar techniques are being applied to pieces of armor to protect the maximum amount of vital area while minimizing the effect on range of motion, balance, weight, etc. The facility location problem has evolved to consider multiple criteria, which is especially important if it is to be used in this process. An extensive list of methods and applications of multi-criteria facility location problem approaches and uses can be found in the literature (Farahani, Steadieselfi, & Asgari, 2010).

Specifically, the armor system optimal design problem can be thought of as a dynamic location-relocation problem, which is described in detail in the literature (Reza, Abedian, & Sharahi, 2009). This location-relocation can be summarized by saying that the decision maker selects a primary location, followed by relocation and development times and new facility location after relocation. For our purposes, the development times have no effect. The uniqueness here is that instead of having a "decision maker," we use an optimization process to determine a new location for each armor component (or "facility") on each iteration. Applying the topic of facility location problems is a novel approach to design with regard to human factors, and especially to a wearable human system.

1.2.5 Topology Optimization

Topology optimization is a form of optimization that focuses on material distribution (Bendsoe & Sigmund, 2003). It is often used in structural problems to optimize the layout of material within a given design space for a given set of loads and boundary conditions. Topology optimization is typically used as a high-level approach for arriving at a conceptual design proposal, which is then improved by human designers. This is very similar to what will need to be done for the system armor design optimization. In a sense, the armor system design problem could be approached as a topology optimization problem. This would be done by using topology optimization techniques to decide the

existence of armor pieces, trying to maximize the objective function while remaining within any bounds of the problem. This approach was eventually modified due to the problem of checkerboarding. Checkerboarding is a common problem with topology optimization (Sigmund & Petersson, 1998) (Zhou, Shyy, & Thomas, 2001) (Bendsoe & Sigmund, 2003) in which the optimization process removes material in order to satisfy constraints, but this often results in enough gaps so that the end solution becomes unusable and unrealistic. These results are currently accepted; however, a designer is often required after the fact to correct the unrealistic aspects of the design.

1.3 Motivation

Given the current state of the art, there is a distinct need for human modeling-and-simulation capabilities that bridge the gap between ballistic armor models and the Warfighters who use them. Armor must be analyzed, designed, and selected not just with consideration of ballistic capabilities, but also with consideration of usability and human interaction. Recent tools have been developed within the Santos DHM software (Mathai, et al., 2010) that allow for assessment of armor tools, and it is by leveraging these tools that this thesis proposes a way to select an optimal armor system for a specific task, and furthermore to be able to design new armor systems based on quantifiable armor system metrics. While it would be possible to design armor systems within the Santos software manually, the work in this thesis allows for automation of the design. These improved

designs could allow for major improvements in design time, usability, and performance from the Warfighters who use them.

There has been work in the area of noting the exact effect that armor systems can have on the Warfighters (Hasselquist, Bensel, Corner, Gregorczyk, & Schiffman, 2008). Studies like this acknowledge the effects that body armor has on the physiology, biomechanics, and performance of the body. There exists a gap, however, as to how to address the armor systems' shortcomings, as well as how to compare between different armor systems.

1.4 Hypotheses

The primary focus of this work can be broken down into three main hypotheses. This work proposes that, by leveraging Santos's already existing armor evaluation tools, it is possible 1) for existing armor systems to be selected based on performance during task simulation, and 2) for new armor systems to be automatically and optimally designed based on user-specified criteria and third-party objectives. If the second hypothesis holds true, then 3) predictive DHM capabilities and product performance can be integrated and embedded within an overarching optimization process.

With regards to the first hypothesis, relating to the method of down-selecting an armor systems based on performance during task selection, the evaluation of the method

validity is straightforward. In order to accurately test this method, it is a primary concern to have a large number of both armor systems and tasks. The number of armor systems is important to ensure that there is a wide variety of options for the method to select, and the number of tasks is important to provide a number of different scenarios for which those armor systems are selected. Once both of these components are made readily available, the testing can be performed extensively by modifying any number of the performance metrics, as well as combining any number of tasks together.

The second hypothesis is slightly less able to be objectively validated. Not only does the method need to design an armor system, but the design needs to be optimal based on the criteria provided to formulate the objective function. Once it becomes possible to determine the existence and location of any number of armor components, it is possible to design what could be defined as an "armor system." However, this result is only useful if the solution can be mathematically defined as optimal based on the criteria that it was supplied to meet. That is, a problem in which the system is intended to maximize coverage and minimize weight is considered sufficient and optimal if no other solution can be provided without lowering the value of objective function. In order to test this hypothesis, one must first develop an optimization method for determining the location and/or existence of armor pieces. By modifying certain design variables, it becomes possible to move these armor pieces in order to maximize or minimize certain user-defined values, such as coverage, weight, range of motion, etc. Once the system for moving the armor components is in place, the next step is to determine a method to import, interpret, and make use of any third-party information. This information needs to

be easy for a user to incorporate in the optimization process, as well as meaningful, in order to provide relevant results. While this general process is considered the realm of human systems integration, and is not new, the novelty of the application is. The approach also contains novelty in its use of a modified version of topology optimization, which includes location along with existence in order to prevent the checkerboard problem.

The third hypothesis states that predictive DHM capabilities and product performance can be integrated and embedded within an overarching optimization process. The proposed method leverages the existing DHM capabilities of Santos as a possible driver for the generation of performance data to be used in the optimization's objective function. In order to verify that this is possible, data from Santos must be used as input for either the constraint or the objective function. The resulting output must modify the armor system and affect the performance metrics in Santos. This continual optimization loop, combined with third-party metrics, would allow for an overarching design optimization process. This process would only currently be able to be used through the armor system optimization filter, although it could be extended to the armor system optimal design in the future.

1.5 Goals

The primary goals for this work are as follows:

To create an Armor System Optimization Filter to automatically select the optimal armor system for a given warfighter and task based on user-defined criteria of the importance of a set of metrics as well as design constraints:

1) Implement a method of down-selection that allows for control over all the available metrics
2) Down-selection should be able to take any number of armor systems, tasks, or combination of metrics

To create an Armor System Design Optimization that will implement third-party data in order to optimize the location and existence of a number of armor components:

3) Implement a technique that will allow for the location of an armor component to be placed exclusively on the surface of the Santos avatar
4) Implement a method to constrain the armor components from overlapping each other
5) Implement a method to allow third-party data sets to be used as objective functions in order to determine the location and existence of the armor components

6) Allow for constraints, such as total weight of an armor system, to be placed on the objective function. These constraints determine the existence of armor.
7) Allow for multiple metrics to be combined in order to make a single objective function.

1.6 Overview

Chapter 2 focuses on Santos, the underlying work that provides the digital human model that is used to calculate armor system design metrics. Specifically, it lays the foundation of how Santos's joints are calculated, motion is predicted, and different performance metrics are determined. The driving force behind the down-selection process is the performance metrics that exist within Santos, and they are explained in detail within the chapter.

Chapter 3 details the concept, approach, and results of the armor optimization filter. This is a tool that is used to select the correct armor configuration under specific circumstances. The user defines a set of criteria as well as any number of tasks, and the armor optimization filter will correctly down-select, through all available armor sets, the system that maximizes or minimizes all metrics.

Chapter 4 focuses on the system design optimization. This is the process of determining the location and existence of armor components in order to maximize/minimize user-defined metrics. Following the introduction is an in-depth review of the approach. This section contains detailed information for the design variables, constraints, and objective functions. The chapter concludes with a section detailing the test cases as well as the results.

The final chapter includes a summary, discussion, and future work.

CHAPTER 2

BACKGROUND

2.1 Introduction

In order to understand some of the processes and methods used later in this work, it is important to be introduced to some fundamental background information. The majority of this work focuses on leveraging the already existing DHM software Santos. These capabilities provide the ability to accurately model human motion, as well as human interaction. Because the goal of this thesis is to focus on the human interaction aspect of armor design and use, these tools are critical. The following chapter provides an overview of the capabilities that are used in Santos, as well as a basic understanding of how they are performed.

2.2 Santos

2.2.1 Overview

In order to understand how motion is simulated in Santos, which is used for the filter, and therefore how a portion of the armor metrics is calculated, it is important to understand the capabilities and foundation of Santos. This section lays out how the avatar is

structured and how posture is predicted, which allows for motion simulation (Mathai, et al., 2010).

The ways in which human posture and motion are simulated depend on how the skeleton and joints are modeled. The Santos avatar treats the skeleton as a series of links, where each pair of links contains at least one revolving joint. Expanding this method allows for modeling the entire human body as a series of several kinematic chains, as shown in Figure 1. The variable q_i is a joint angle and represents the rotation of a single revolute joint. There is one joint angle for each degree of freedom (DOF). The variable $\mathbf{q} = [q_1, \ldots, q_n]^T \in R^n$ is the vector of joint angles in an n-DOF model and represents a specific posture. The variable $\mathbf{x}(\mathbf{q}) \in R^3$ is the position vector in Cartesian space that describes the location of an end-effector with respect to the global coordinate system. For a given set of joint angles, $\mathbf{x}(\mathbf{q})$ is determined using the DH method (Denavit & Hartenberg, 1955). With this work, a 55-DOF model for the human torso, arms, legs, and neck, as well as six global DOFs, three for translation of the hip point and three for rotation about the hip point, is created.

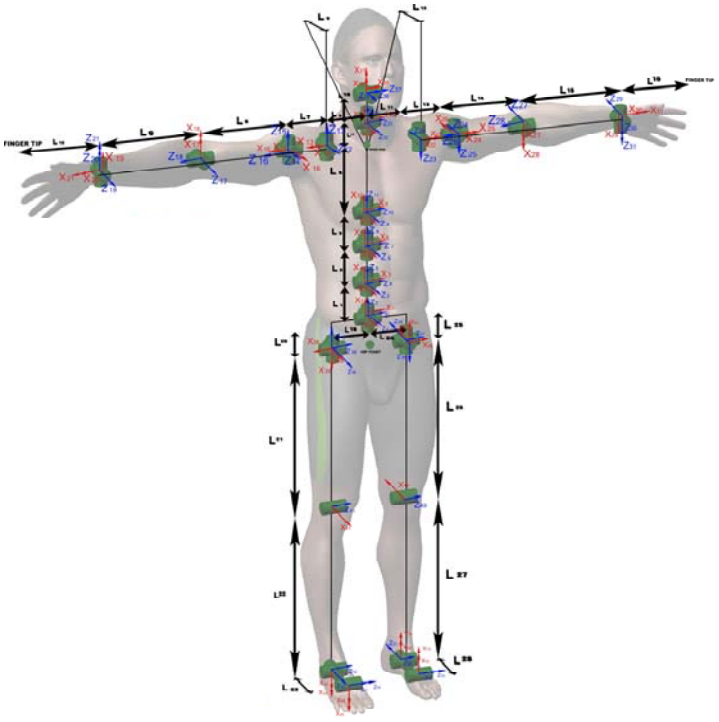

Figure 1: The underlying Santos digital human model

A fundamental element in the proposed motion simulation method is optimization-based posture prediction, as detailed in the literature (Marler et al. 2009). Given the human model described above, the design variables for the posture-prediction problem are q_i, measured in units of radians. In this case, the objective function is proportional to the deviation from the avatar's initial position q^I. Because some joints articulate more readily than others, a weight w_i is introduced to stress the relative stiffness of a joint. The first constraint, called the distance constraint, requires the end-effector to contact a target point. In addition, each joint angle is constrained to lie within predetermined limits. The variable q_i^U represents the upper limit, and q_i^L represents the lower limit. The optimum posture for the system is then determined by solving the following problem:

$$\textit{Find}: \ \mathbf{q} \in R^{DOF}$$

$$\textit{to minimize}: \ f_{Effort}(\mathbf{q}) = \sum_{i=1}^{n} w_i \left(q_i - q_i^I\right)^2$$

$$\textit{subject to}: \ \text{Distance} = \left\| \mathbf{x}(\mathbf{q})^{\text{end-effector}} - \mathbf{x}^{\text{target point}} \right\| \leq \varepsilon$$

$$q_i^L \leq q_i \leq q_i^U; \ i = 1, 2, \ldots, DOF$$

Equation 1: Posture optimization formulation in Santos

With ε being a small positive number that approximates zero. The equation above is solved using the optimization software SNOPT, discussed in Section 2.3.

2.2.2 Armor Performance Evaluation

Leveraging the posture prediction methods detailed in Section 2.2.1, motion can be simulated by defining tasks for the avatar (Mathai, et al., 2010). Given the simulation of a task using motion simulation, the performance of body armor systems is evaluated from an HSI perspective using a series of metrics described in this section. These metrics then provide the constraints and objective functions for the system optimization filter discussed in Chapter 3 and are laid out in detail in the literature (Mathai, et al., 2010). With the approach that is used for the armor system design optimization, it would be possible to use these metrics if a point cloud were created from their information; however, this would be future work.

Coverage measures the percentage of the body that is covered by the armor. This metric is estimated by calculating the area of the polygons on the avatar's skin that are covered by armor models. In the Santos environment, all three-dimensional models are made up of a mesh of triangular elements, so the coverage area can be determined by calculating the triangle area on Santos's body mesh that is overlapped by armor.

The balance metric is a measure of how stable the avatar is while performing the provided task. The stability parameter essentially compares the position of the zero moment point (ZMP) (Marler et al., 2011) to the center of the foot support region (polygon formed by connecting the heels and toes of the feet as seen in Figure 2). The center of the foot support region is assumed to be the most stable balance point. The ZMP is affected by the weight of armor as well as the task being performed.

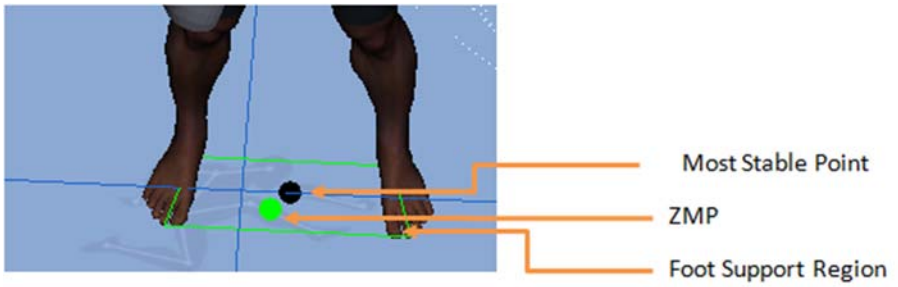

Figure 2: Diagram indicating the location of ZMP and foot support region

The weight of the armor is simply calculated by estimating the volume of each armor piece and considering user-defined density. The software is also capable of estimating the average density of the armor if the user provides the weight.

The restriction metric essentially indicates which armor would have to be removed in order to perform the full range of motion for the specified task. The restriction value is used to measure the intersection between each armor component as well as between the armor and the skin. In order to perform this estimation, the armor components are filled with voxels (three-dimensional polygons that represent unit volume), and the intersections between armor voxels are measured to estimate the overall restriction, as shown in Figure 3. Portions of armor can then be exported to CAD systems for redesign.

Range of motion is the measure of the extent to which joint rotation is restricted due to armor. Before the armor is parented to the avatar, each joint has a default range of motion. This range is reflective of the normal joint rotation for an average human. In order to determine the range of motion of a joint as affected by armor, the joint is rotated from the minimum value through the maximum value in small increments. For joints that have more than a single degree of freedom, the range of motion test is performed with each degree. At each increment along the default range of motion, collisions between two polygons representing skin on armor, skin on skin, or armor on armor are checked. If a collision occurs, then the test is terminated, and the last incremented value is set as the new range of motion.

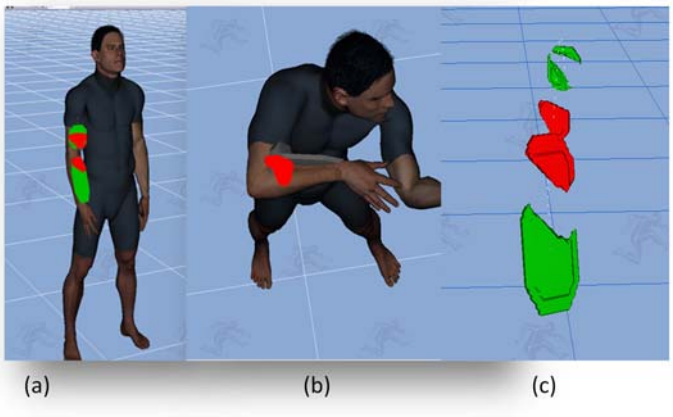

Figure 3: (a) Surrogate geometry representing armor is placed on the avatar. Armor geometry is analyzed for encumbrance. Encumbered volume is represented in red. (b) Volume of the armor that interferes with task performance is shown. (c) Exported OBJ version of armor showing encumbered volume (red) and unencumbered volume (green).

Task performance is measured by aggregating performance measures for each of the interpolated postures. These measures include discomfort, potential energy, and joint displacement (Marler et al, 2009). The performance measures are evaluated at discrete uniform intervals during the motion. The time intervals are specified by the user.

When various weights (i.e., body mass, armor mass, etc.) and forces are considered, necessary *joint-torques* are predicted, subject to strength limits and characteristics (Marler et al,, 2012). Relative values for these joint-torques are displayed graphically on the avatar during task completion in order to show which joints are exercising the highest torques. In addition, a joint-torque performance measure is calculated based on joint

torques generated in Santos while performing the task. This measure is a reflection of the armor weight on Santos as well as the task being performed.

These metrics will be the foundation upon which the current Armor System Optimization Filter method is based by providing the overall score for each armor system. Although the approach discussed in Chapter 3 uses only these metrics, it was designed in such a way as to allow for any additional calculations to be implemented in the filtering process.

2.3 SNOPT

The methods put forth in this thesis focus on the transformation and use of external data in order to create objective functions that satisfy certain constraints and trend towards design goals. The objective functions that will be discussed in Chapter 4 for the armor system design optimization are solved using the software SNOPT (Gill, Murray, & Saunders, 2002), which utilizes sequential quadratic programming (SQP). SQP has become the most successfully used method for solving nonlinearly constrained optimization problems. In order to use SQP, both the problem's objective function and its constraints need be continuously twice differentiable. The SQP method is based on gradients, meaning that it requires gradients for the objective function as well as the constraints. The gradients used in this method can be determined either analytically or computationally; however, providing analytical gradients will reduce computational time

and can increase accuracy. A drawback of using any gradient-based approach is that there is a tendency to find a local minimum as opposed to a global minimum. A local minimum is a point that is minimum only within a local area of feasible space, whereas a global minimum is the minimum over the entire feasible space. SQP methods work by solving a sequence of optimization sub-problems. Each of the sub-problems is solved by optimizing a quadratic model of the objective function subject to a linearization of the constraints. In the case that the problem is unconstrained, then the method will reduce to Newton's method. Although SNOPT is not the only optimization software package that implements SQP (NPSOL is another notable such package), SNOPT was designed with the motivation of being able to solve increasingly large models. To this end, SNOPT was developed on a new sparse SQP algorithm. The approach that SNOPT uses specifically exploits the sparseness of the constraint Jacobian and maintains a limited quasi-Newton. This is especially relevant to the problem of armor system design, as the constraint Jacobian for the armor overlap discussed in Section 4.2.2 is a sparse matrix.

CHAPTER 3

ARMOR SYSTEM OPTIMIZATION FILTER

This chapter provides the methodology and results of using the armor optimization filter. This chapter builds upon the Santos avatar and the newly developed metrics for evaluating armor systems discussed in the previous chapter. Previously, all armor systems needed to be evaluated independently, which could be time intensive. It was apparent that a useful tool could be developed by leveraging the metrics that existed in Santos and implementing them into an automated filtering method. The filtering method laid out in this chapter is intended to be a practical tool used to select the optimal armor system for a specific set of tasks, given specific criteria (e.g., minimal weight and maximum range of motion).

3.1 Introduction

Currently, with respect to human factors, PPE systems are evaluated on a manual basis, which can lead to sub-optimal designs or to an unclear and unquantifiable understanding of the shortcomings of a given armor system. For example, it may be extremely difficult for a warfighter to maintain a certain aiming posture over an extended period of time with a PPE system that was designed for someone who weighed 20 KG more and measured several centimeters taller. Recalling the first hypothesis proposed in Chapter 1, it was

theorized that it would be possible for existing armor systems to be down-selected based on performance during task simulation. The method developed in this chapter strives to automate the process that is now performed experimentally. For practical purposes, there is no single optimal armor for every situation, purpose, or warfighter, as different tasks require different objectives. The proposed armor system design optimization is intended to provide new designs based on the user-defined set of metrics, which are used to emphasize certain aspects. These metrics are calculated using the DHM software, which allows for strictly quantifiable results. With this in mind, there is an unlimited number of optimal designs that can be generated from this process. The proposed down-selection method can be used as an efficient way to select from these already approved optimal designs, without the need for extensive education on what went into the original design. This process can be used for quickly and efficiently selecting the armor that is most fitted for the task at hand.

3.2 Approach

The armor system optimization filter leverages already existing armor metrics; however, the metrics necessitated normalization in order to be compared. New methods and functionality were added to the software in order to implement the armor system optimization filter, and they are explained below.

The single requirement in order to run the system at all is that an armor system library must exist. After the library is selected, the user must define criteria in order to allow the method to determine the optimal armor system. From those pre-processing steps on, the approach can be thought of as a cyclical process with the following steps:

1. The armor system is imported onto the Santos avatar
2. Any tasks that have been defined are simulated
3. All armor system metrics are calculated
4. Results for the armor system are stored in a data structure
5. The library is searched for the next armor system

These steps are repeated until there are no longer any armor systems remaining in the library. The scores are then compared, and the optimal armor system is loaded onto the Santos avatar, with the corresponding scores displayed to the user. This process is illustrated in Figure 4 as well as outlined with psuedocode in Figure 5.

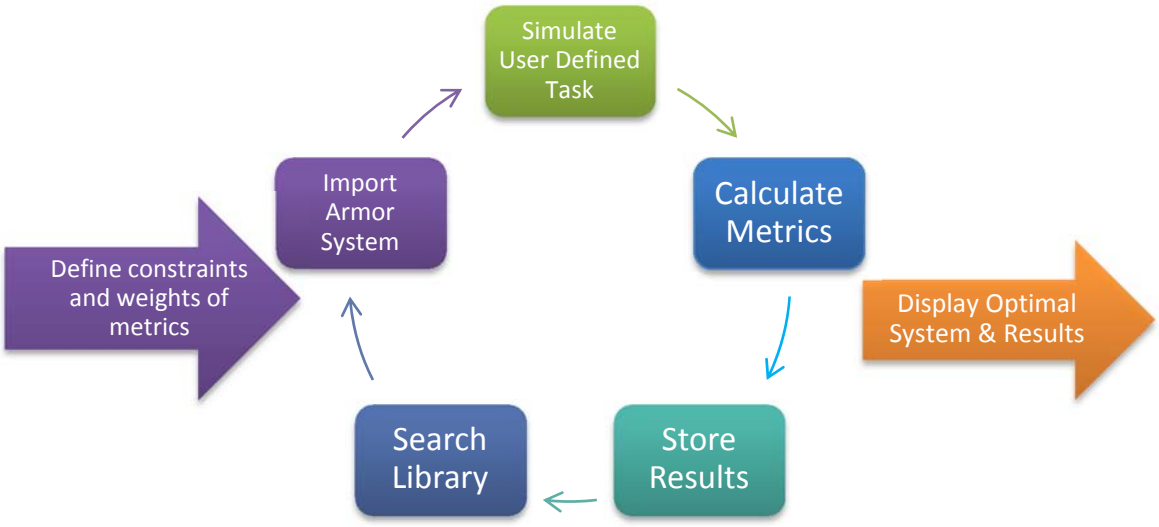

Figure 4: Flowchart of the armor system optimization filter process

```
foreach(Armor System){
    foreach(Metric){
        if(NoExceededConstraints  && MetricWeight>0){
            Val = CalculateMetric()
             if(Val <= MetricConstraint)
                 store Val*MetricWeight
        }
    }
}
```

Figure 5: Pseudocode for Armor System Optimization Filter

This process is performed by first allowing the user to define the constraints and weighting factors of each metric. This is the primary user input that accompanies the rendering of tasks, as well as the selection of an armor library. The method then imports an armor system, simulates the defined tasks and calculates the metrics, stores the results, and finds any other armor systems available in the library. This process is automatically

repeated until the library is entirely searched through, and the optimal armor system as well as results are returned to the user. Each of the steps shown above is explained in detail in the following sections.

3.2.1 Define Constraints and Weights of Metrics

The search filter provides the user with a set of definable weights and constraints for each metric as shown in Figure 6. The weighted slider shown beneath each metric allows the user to define the emphasis with which they would like to consider that score for the overall rating. If the weight is set to zero, then that metric is not considered at all. If all of the weights are set equally, then the filter searches for the optimal armor system with respect to all metrics.

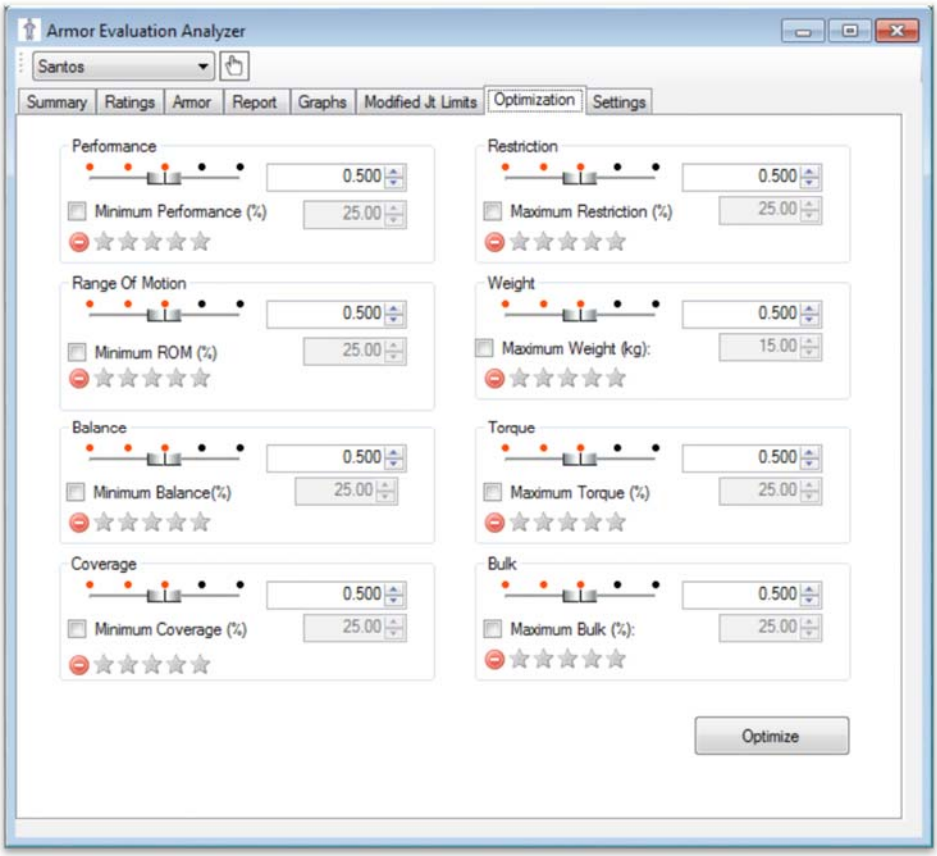

Figure 6: Graphical user interface to define metric weight-factors and constraints

3.2.2 Import Armor System

Armor systems can be created and stored on Santos using any modeling object set as an "armor" piece. When a model is set as an armor piece, it is then considered in all armor metric calculations. These armor pieces are parented to Santos' joints. When an armor piece is parented to a joint, it will move with respect to that joint throughout any task that is being performed in order to mimic real motion throughout a task. A special file

configuration was made in Santos to allow any set of armor pieces, with all relevant information to be saved and loaded back onto an avatar. Each armor system is loaded onto this avatar, with individual armor pieces being parented to their appropriately corresponding joints. These joints are determined either by name association in the file or by closest location. After an armor system's metrics have been evaluated, the armor pieces are removed from the avatar to allow for the next armor system.

3.2.3 Simulate User-Defined Task

Because some of the metrics, such as weight, coverage, and bulk, can be calculated without the use of any motion, the down-selection process can be performed without the need for any user-defined task. However, in order for all of the metrics to be calculated, a task must be rendered in the Santos software. These tasks can be linked together in order to form a complex task such as walking, then laying down, and then aiming. There is a pre-existing set of tasks available in the Santos software, but a user is also able to create any task that they deem necessary.

3.2.4 Calculate Metrics

The usefulness of the down-selection method depends heavily on the metrics that it uses to determine the optimal armor system. In this case, the types of metrics have been laid out in Section 2.2.2. In order to make use of these metrics, they required normalization for easy comparison. Each score is normalized to be between 0 and 1. A five-star rating system (Figure 7) has been developed as a quick and easy way to compare different armor configurations. A five-star rating represents the highest score and a zero-star rating represents the worst score. The star ratings are based on the comparison parameters discussed in the previous section and generally indicate average performance during task completion. In general, each star represents 20% of the maximum outcome. The methodology to calculate the various star ratings is described as follows.

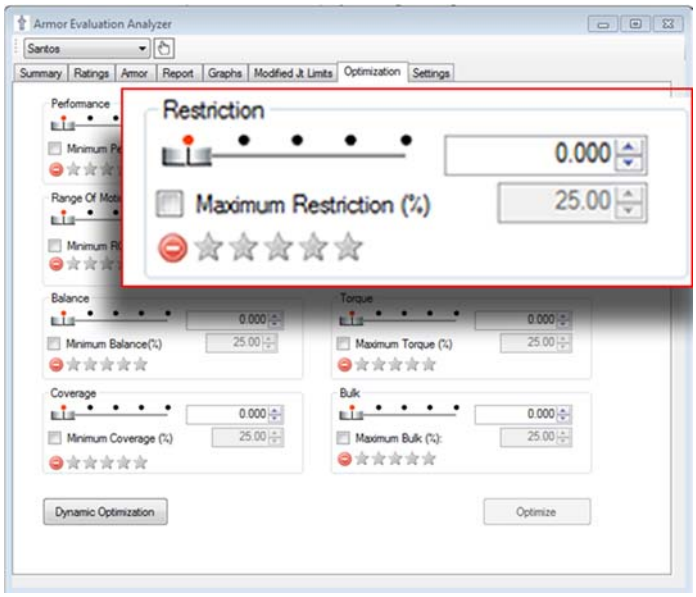

Figure 7: Star rating system and GUI for the Armor System Optimization Filter

3.2.4.1 Performance

For the performance rating, the performance measure values are converted to a ratio of the maximum possible performance measure value and then averaged. The average is converted to a percentage and assigned a rating. This gives us:

$$Performance\ Rating = \left(\frac{Performance\ with\ armor}{Performance\ without\ armor}\right) \qquad \in (0,1)$$

Equation 2: Normalized formulation of performance metric

3.2.4.2 Range of Motion

For the range of motion rating, the range of motion with armor is compared to the default (assumed maximum) range of motion for Santos. Each joint on the avatar has an existing default range of motion. The default range of motion is meant to mimic the normal joint rotation for an average human. For the range of motion with armor, each joint is rotated from its lowest range to its highest in small increments. At each increment, a collision detection is performed between the armor and the skin. If a collision occurs, the test is terminated and this new angle is determined as the range of motion. The ratio of the total range of motion with armor as compared to the default range of motion is then converted into a percentage and a star rating.

$$Range\ of\ Motion\ Rating = \left(\frac{Range\ of\ Motion\ with Armor}{Default\ Range\ of\ Motion}\right) \qquad \in (0,1)$$

Equation 3: Normalized formulation of range of motion metric

3.2.4.3 Balance

The balance metric uses the measures of the distance of the ZMP (Marler, Knake, & Johnson, Optimization-Based Posture Prediction for Analysis of Box-Lifting Tasks., 2011) from the center of the foot support region and the distance of the ZMP from the nearest edge of the foot support region. These two values are then converted into a ratio in order to estimate the stability metric. By evaluating the metric over a series of postures, a rating is determined from the average balance.

$$Balance\ Rating = \sum_{i=0}^{n} \frac{\left(\frac{Distance\ of\ the\ ZMP\ from\ Center_i}{Distance\ of\ ZMP\ from\ nearest\ edge_i}\right)}{n} \qquad \in (0,1)$$

where:

n = the number of postures in a motion

Equation 4: Normalized formulation of balance metric

3.2.4.4 Coverage

As discussed in the previous section, the coverage rating utilizes the polygon mesh of Santos. The coverage of the armor is calculated by summing the area of the polygons of Santos that are covered by the armor pieces. This is then compared with the total polygon area of Santos. The ratio of the armor coverage area to the total body surface area is used to determine the rating.

$$Coverage\ \ Rating = \left(\frac{Area\ of\ Santos\ Covered\ by\ Armor}{Total\ Area\ of\ Santos}\right) \quad \in (0,1)$$

Equation 5: Normalized formulation of coverage metric

3.2.4.5 Restriction

The restriction metric for each armor piece always exists between 0 and 1. Therefore, the rating for restriction is the average of the total armor restriction of each armor piece.

$$Restriction\ \ Rating = \sum_{i=0}^{n} \frac{(Restriction\ of\ armor\ component)_i}{n} \quad \in (0,1)$$

where:

n = the number of armor pieces

Equation 6: Normalized formulation of restriction metric

3.2.4.6 Weight

In order to calculate the weight rating, the weight of the armor has to be compared to a baseline standard. The baseline standard is derived from the current weight of the armor worn by Warfighters, as published by PEO Soldier (Fricker & Wilson, 2010). The standard armor weight is approximately 15 kg. The rating is based on the ratio of standard armor weight to the weight of the armor being tested. This is based on the assumption that any new armor configuration design has to be less than 15 kg for it to be a design improvement.

$$Weight\ \ Score = \left(\frac{Comparison\ Weight - \ Armor\ Weight}{Comparison\ Weight}\right)$$

$$Weight\ Rating = \ f(Weight\ \ Score) = \begin{cases} 0, & Weight\ \ Score < 0 \\ Weight\ \ Score, & Weight\ \ Score \leq 1 \\ 1, & Weight\ \ Score > 1 \end{cases}$$

where:

$Comparison\ Weight$ = The weight the armor is to be compared to. Default set 15KG.

Equation 7: Normalized formulation of weight metric

3.2.4.7 Torque

The torque rating is compared to the base rating of the avatar in motion without any armor present, giving the equation:

$$\text{Torque Rating} = 1 - \left(\frac{Torque\ with\ armor}{Torque\ without\ armor}\right) \qquad \in (0,1)$$

Equation 8: Normalized formulation of torque metric

The rating must be subtracted from 1 because the metric determines the decrease in torque due to the armor. This subtraction allows that a higher value is optimal, which is how the rest of the metrics operate.

3.2.4.8 Bulk

The bulk metric is based on the ratio of the volume of the body armor on the avatar, over what would be the volume of the entire avatar covered in one inch of armor. The one-inch armor allows for the rating to be between zero and one for all practical purposes. The maximum volume of the armor system was chosen empirically in order to ensure the range of the rating; however, it could be easily modified in the future if a new maximum is found to be more useful.

$$Bulk\ Rating = \frac{Volume\ of\ Armor}{Volume\ of\ entire\ body\ covered\ with\ 1\ inch\ armor} \qquad \in (0,1)$$

Equation 9: Bulk rating normalized

3.2.4.9 Overall Score

All of the metrics stated above are normalized between 0 and 1, and once they are normalized, they are multiplied by the user-defined weight to give a final value for each metric. All of these metrics are the summed to give an overall score. This formula can be described as

$$Overall\ Rating = \sum_{i=0}^{n} w_i * (s_i)$$

Equation 10: Overall rating of an armor system.

where w_i and s_i are the user-defined weight and the normalized score, respectively, of a metric i, and n is the number of armor metrics. The range of s_i is also from 0 to 1. After each metric calculation, the constraints are checked, and if any constraint is violated, the testing of that armor system ceases in order to save time, and all scores are reported as zero.

3.2.5 Store results

In order to compare armor systems, the scores must be stored and compared after the last armor system's metrics have been computed. However, in order to prevent redundant computations, each individual metric for all armor systems must be stored, so that when an armor system is determined optimal, the ratings of each metric can be displayed to the user. These metrics are placed in a data structure that allows for quick comparison, and are all deleted after the optimal armor system is determined and displayed to the user.

3.2.6 Search the Library

The software then iterates through all of the available armor configurations in the provided folder and performs the user-defined task with each. In order to perform in minimum time, only the configurations/evaluations necessary to obtain the metrics of interest are considered. For example, if a weighting factor of a metric is set to zero, that metric is not calculated. Additionally, an armor system's calculations are immediately halted if a constraint is found to be validated. After the software has iterated through all of the possible armor configurations and found the one that is optimal over the user-defined weight, it is placed back on the avatar with its ratings displayed.

3.3 Results

In order to test the system, four different examples are provided below. An example library of body armor systems is shown in Figure 8, which was used for all of the examples. For each example case, a series of three different tasks (acquiring a target while standing, grenade throwing, and acquiring a target while prone) is used (Figure 9).

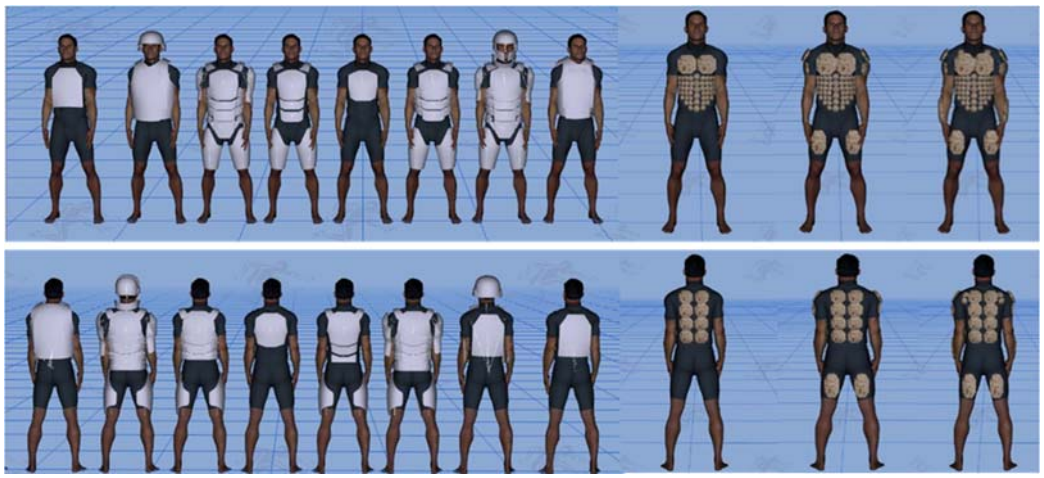

Figure 8: An example library of armor systems

Figure 9: The three tasks performed for an Armor System Optimization Filter example

For the first example (Figure 10), weighting factors for coverage, range of motion, restriction, bulk, performance, balance, and torque were set to zero, while the weighting factor for armor weight was set to 1. This implies that the only criteria in selecting an armor system would be the minimum weight. Since the example entails setting the weights of coverage, range of motion, restriction, bulk, performance, balance, and torque to zero, those calculations are not performed. Thus, the armor system with the lightest weight is selected.

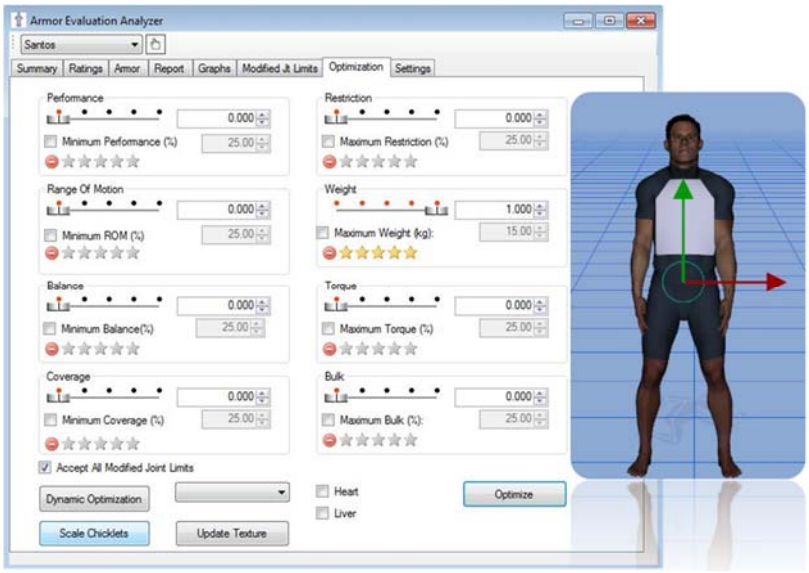

Figure 10: Armor System Optimization Filter metric results with optimal armor system for minimum weight

For the second example (Figure 11), weighting factors for armor weight, range of motion, restriction, bulk, performance, balance, and torque were set to zero, while the weighting factor for coverage was set to 1. This implies that the only criteria in selecting an armor system would be maximum coverage. Since the example entails setting the weights of armor weight, range of motion, restriction, bulk, performance, balance, and torque to zero, those calculations are not performed. From this, it is evident that the filter returns the armor system that covers the maximum area of the Santos avatar.

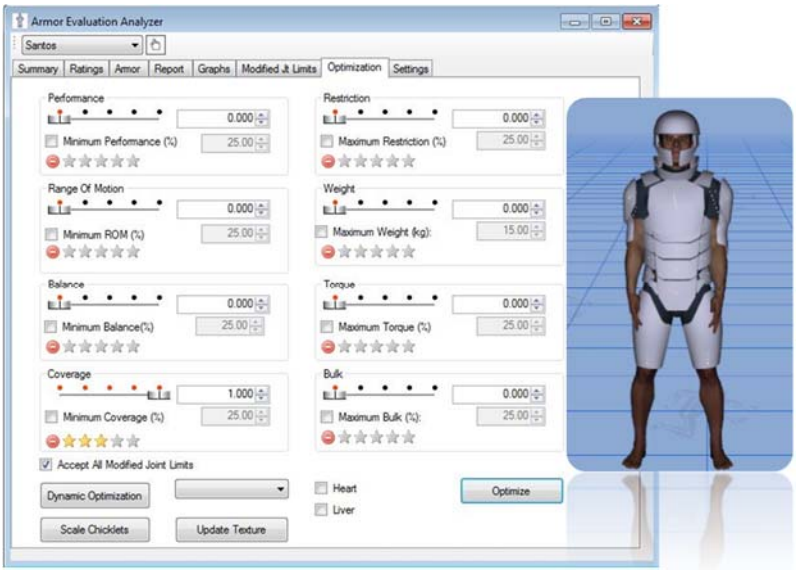

Figure 11: Armor System Optimization Filter metric results with optimal armor system for maximum coverage

For the third example (Figure 12), weighting factors for range of motion, restriction, bulk, performance, balance, and torque were set to zero, while the weighting factors for armor weight and coverage were set to 1. This implies that the armor system optimization filter should return the armor with the optimal balance of weight and coverage. Since the example entails setting the weights of coverage, range of motion, restriction, bulk, performance, balance, and torque to zero, those calculations are not performed. This result can only be subjectively validated, but the results fall within a plausible realm.

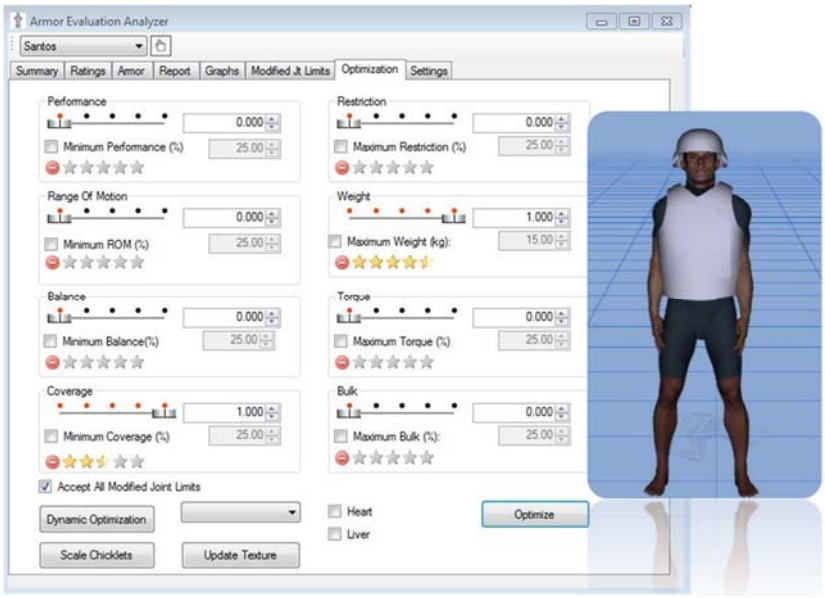

Figure 12: Armor System Optimization Filter metric results with optimal armor system for minimum weight, maximum coverage

For the final example (Figure 13), weighting factors for performance, balance, and torque were set to zero, while the weighting factors for armor weight, coverage, and bulk were set to 1. The restriction and range of motion weighting factors were set to 0.5. This means that weight and bulk are twice as important as range of motion and restriction. Since the example entails setting the weights of performance, balance, and torque to zero, those calculations are not performed. These results too can only be validated subjectively, as it is a more complex problem.

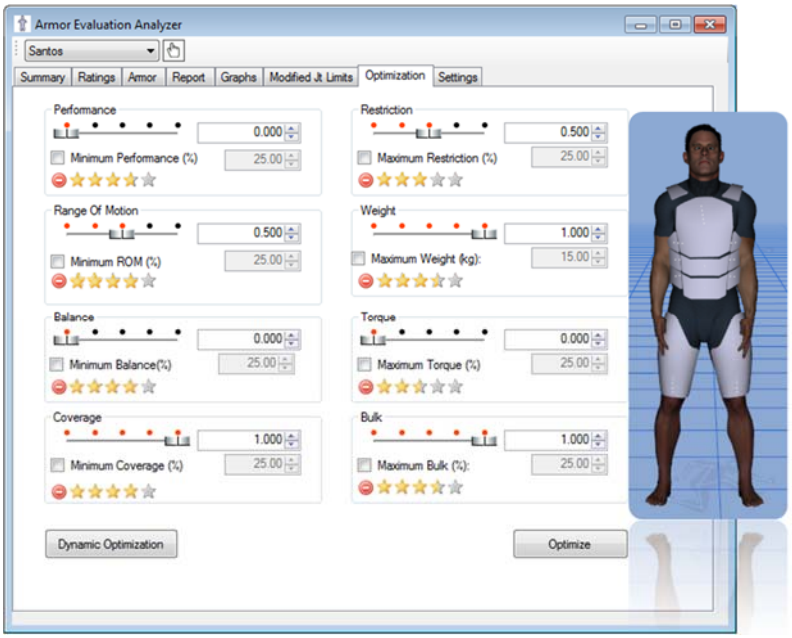

Figure 13: Armor System Optimization Filter metric results with optimal armor system of a balanced system

The results in each example provide an overview of the armor system selected, as well as the returned metric values. Due to the nature of some of the more complex problems, validation can only be performed subjectively. With that in mind, the results for the example do fall in line with what one would expect. This armor system had a lighter material than the user-created armor component designs, as well as a fair balance of coverage and light weight. As the types of problems become more difficult, the solutions become less intuitive and more difficult to validate. It is also of note that in the examples shown, the user-created armor component designs were not selected. This is due to the characteristics of the designs used in the library and the tests run, as they were neither light enough, nor covered enough of the avatar to be selected for the chosen examples.

CHAPTER 4

ARMOR SYSTEM DESIGN OPTIMIZATION

Chapter 3 detailed a method for selecting optimal armor based on metrics derived from digital human modeling software. Chapter 4 will expand upon these capabilities and provide the next step in the suite of armor system design tools by detailing a method for automatically designing optimal armor systems using continuous optimization. The armor system design optimization process detailed below can be used to create a series of armor systems that are optimally designed using third-party data. These new armor systems can then be used to create the type of armor system library that is described in Chapter 3.

4.1 Introduction

As material design continues to progress in terms of metrics such as weight, flexibility, bullet penetration, and blast impact, the technique for implementing wearable designs with regard to human factors has not been proportionately considered or advanced. It is important to acknowledge the human aspect of any designed material interaction, but especially so in the case of wearable PPE, as these materials are intended to be worn for long periods of time without hindering performance, burdening the user, or becoming uncomfortable. In order to properly account for all of the factors involved with wearable PPE, a human-centric design approach has been taken. This approach allows for quick, accurate, and exhaustive design of optimal armor systems based on real armor

metrics such as performance, range of motion, balance, torque, and coverage. It is a hypothesis of this thesis that, with this new approach, it is possible to design armor that offers both more protection and an over-all better experience for the user.

This chapter provides details of all of the development of all of the approaches taken throughout the duration of this project, including initial approaches for the coverage objective function. All approaches are detailed as to why they were taken, and their benefits and shortcomings are described. The chapter concludes with a set of results using the most current methods.

4.2 Approach

The armor optimization is completed in two phases: creation of an objective function, followed by location/existence optimization of individual armor components. An overview of the optimization over a single segment can be visualized in Figure 14.

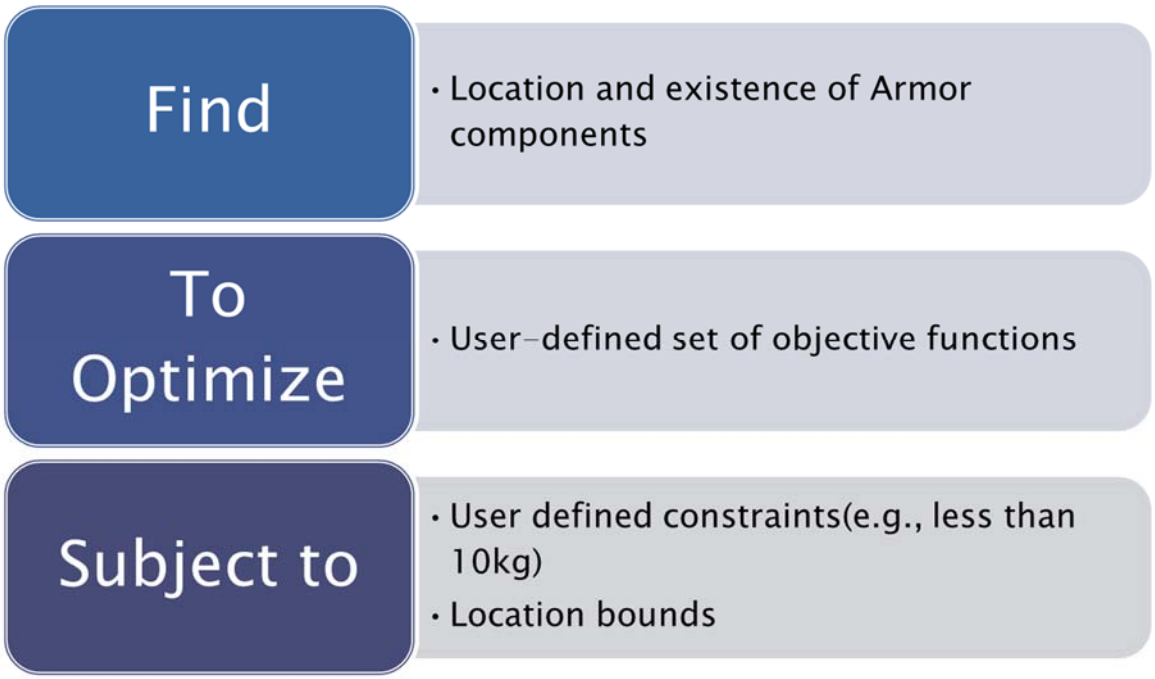

Figure 14: Overview of Armor System Design Optimization

For the first phase of the process, third-party data is used to create an objective function using regression analysis. This objective function can be formed by any kind of data, external or internal to the Santos software. For instance, it can be used to quantify the coverage at a given location on Santos. The user supplies point cloud data, which can represent organ location, blast pressure, or any other data that would be meaningful to optimize over. This point cloud data is composed of three-dimensional location data and a value at each location. The data is then used alongside a previously created, default point cloud that covers the entirety of the Santos avatar. This default point cloud ensures that an objective function can be formed even without the use of external data. An example of this default point cloud gradient is shown in Figure 15, where lighter colors ultimately reflect increasing importance. There are two reasons external data may not be

used directly to create the objective function. First, there is no guarantee that the point cloud data is dense enough to ensure a data point at every location. Secondly, there is no guarantee that, even if the data is dense enough, it encompasses the entire body. Trying to optimize over either of these scenarios leads to discontinuities and, therefore, failed results. In order to circumvent these problems, regression analysis is performed using both the user-supplied data and the default gradient, guaranteeing a single smooth, continuous function. Details of why and how the regression analysis is performed are provided later in Section 4.2.3.2. This process is repeated, and an objective function is formulated for each of the five segments.

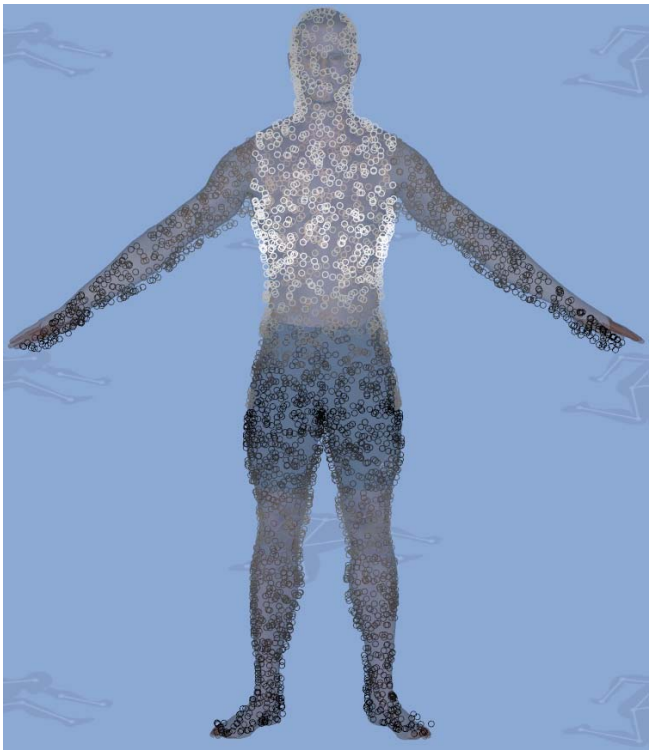

Figure 15: Example default point cloud

Once regression analysis has been performed and the objective function has been created, optimization is run for the location and existence of each armor component. The objective function, gradients, bounds, and constraints must all be defined prior to running the optimization, and the results are then returned and enforced in Santos. The optimization process is done using the SNOPT optimization software discussed in Section 2.3

4.2.1 Design Variables

The initial challenge with this approach was formulating the problem in such a way that the design variables would be continuous as well as bounded to the surface of the Santos avatar. This difficulty is due to the avatar's surface being defined in three dimensions but without any way to define the three-dimensional design variables that bind the armor component's location to that surface. Due to this complication, new methods were developed. Note that it was decided that a gradient-based optimization algorithm would be used, rather than a multi-start approach, for the sake of computational speed. As an initial step, the goal was to maximize a coverage score that was weighted with respect to the importance of the body segment being covered. However, the textures for Santos's skin are not continuous; they have seams (Figure 16). These seams indicate that if the u-v coordinates were used as design variables, there would be discontinuity in the data wherever a u-v coordinate did not map to the three-dimensional Santos mesh. Thus, it is

not possible to use traditional u-v coordinates (relative to the mesh) as design variables. In order to overcome this, the problem was translated from a three-dimensional mesh into a continuous two-dimensional problem. This was done by having two design variables, θ_i and z_i, for each armor-piece i. These design variables represent movement of an origin ray along the surface of a cylinder (Figure 17), with one cylinder for each body segment. The position on the cylinder acts as an origin point, from which a ray is projected towards Santos in order to determine a corresponding point on Santos's body mesh. This alleviates the problems of discontinuity in the design variables and provides a much easier way to formulate the problem. There is also a design variable for each armor component that is introduced for the weight objective function; it is known as the existence variable and is denoted as e_i for each armor piece i. This design variable is used to affect the weight of the armor piece, as well as allow for an armor component's removal if it is set to zero.

It should also be noted that using a Non-uniform rational basis spline (NURBS) was considered in order to provide a continuous function while also allowing for analytical gradients. NURBS is a mathematical model that is often used in computer graphics for generating and representing curves and surfaces. While a NURBS surface would be useful, it unfortunately is not feasible using the Santos avatar that exists in the DHM software used. The Santos avatar uses a polygonal mesh and was modeled in such a way that although the mesh could be converted into a NURBS surface, the surface would be segmented into a large number of fragmented sub-surfaces, which would not allow for whole body optimization. Another issue is the size of the NURBS surface that would be created. In order to create a detailed surface of the Santos avatar, the NURBS surface

would be a series of tremendously large files, which would require large processing times. With these considerations in mind, it is apparent that using a NURBS model is not a valid solution for this particular problem.

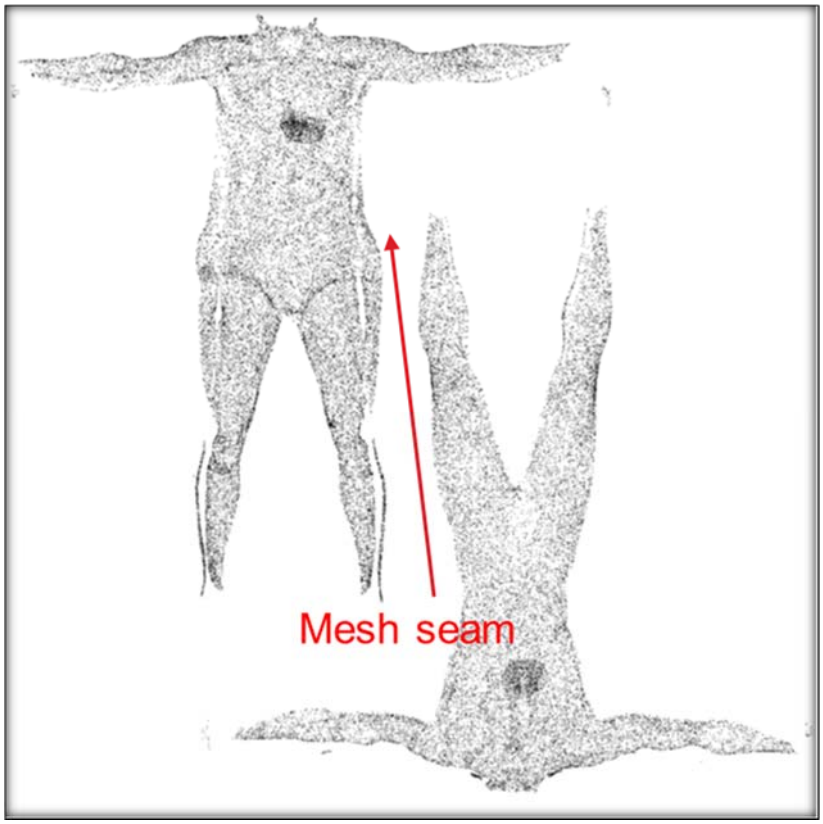

Figure 16: The UV mapping of the Santos avatar

4.2.2 Constraints

One of the many difficulties associated with the problem of armor system design is the prevention of overlapping of the different armor pieces. When a local optimum is found

for the objective function, all armor pieces will tend towards that value unless a constraint is placed that enforces a minimum distance between pieces. As the armor system design problem grows in the number of pieces, it also grows in complexity. At its core, the overlap problem is analogous to a common problem in three-dimensional modeling, which is collision detection. In collision detecting, a series of objects must be checked to see if an overlap occurs, a process which can be computationally expensive. This is a topic that has been well documented, and there are a variety of approaches (Jiménez, Thomas, & Torras, 2001), (Kockara, Halic, Iqbal, Bayrak, & Rowe, 2007), (Lin & Gottschalk, 1998). The initial methods for collision detection here are simplistic, but this is a topic that could be expanded on in future work in order to provide faster and more accurate optimal solutions. In order to overcome this, two different overlap constraints were formulated and tested.

The initial constraint formulation uses a distance approximation method to ensure that any two components, i and j, are separated by length r. This is done by considering a normalized approximation of the distance between the two points of origin for the rays that are drawn from the segment cylinder to the avatar mesh. An empirically found value to represent the distance of two armor component's radii is used as the lower limit for the constraint. The values are normalized in order to account for the different sizes of each cylinder segment's radius. Also, because this constraint acts as only a general approximation of the armor component's final position, the empirically determined lower limit for the constraint is sufficient for the purpose, and the actual value of the armor component's radius is not needed. Therefore, the constraint function is written as:

$$c_k \geq r_i + r_j$$

where:

$$c_k = \frac{(s*\theta_j - s*\theta_i)^2}{\pi^2} + \frac{(z_j - z_i)^2}{(z_{max} - z_{min})^2} = (\frac{\Delta\theta * s}{\pi})^2 + (\frac{\Delta Z}{z_{max} - z_{min}})^2$$

k = Index of constraint C

$z_i = z$ position of the origin ray on the cylinder for component i

$z_j = z$ position of the origin ray on the cylinder for component j

$\theta_i = \theta$ position of the origin ray on the cylinder for component i

$\theta_j = \theta$ position of the origin ray on the cylinder for component j

s = radius of the armor cylinder for a given segment

z_{max} = maximum z value of the cylinder

z_{min} = minimum z value of the cylinder

r_i = Empirically found normalized distance corresponding to radius of component i

r_j = Empirically found normalized distance corresponding to radius of component j

i = Index of an armor component

j = Index of an armor component

Equation 11: First attempt of overlap constraint and its gradients.

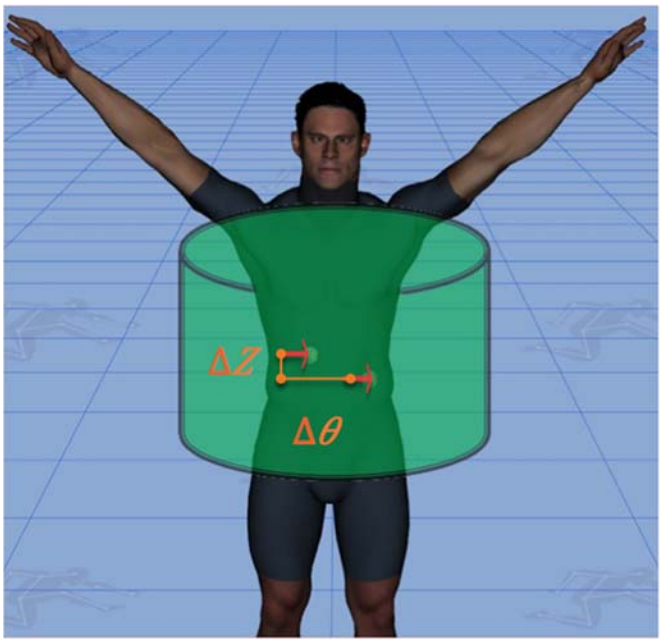

Figure 17: The components of the constraint function. The variables ΔZ and $\Delta\theta$ determine the difference approximation of the location of the origin point for the collision ray and have a lower bound to prevent overlap

A visual representation of this constraint can be seen in Figure 17. This constraint is then placed between every possible combination of armor components, as shown in Figure 18. In this picture, each line represents the distance constraint between the two armor components. Using this approach, the number of overlap constraints is as follows:

$$m = \frac{n(n-1)}{2}$$

where:

m = number of constraints

n = number of armor components

Equation 12: The number of constraints using the first overlap formulation

Figure 18: An example of the three constraints that are associated with a three-armor-piece problem

An example of this formulation with three armor pieces and therefore three constraints (Equation 12) can be seen in Figure 18. While this approach is guaranteed to prevent overlap, it also has its downside. Due to the need to check between every possible combination of armor components, the number of possible constraints grows as $O(n^2)$. This becomes extremely computationally intensive when the armor pieces grow to

numbers that begin to cover whole segments of the body. After testing this problem on a larger scale, SNOPT began to fail and abort the program. This was due to SNOPT's implementation in a section of unmanaged memory within the computer. When the problem grew to a large size, the memory that needed to be allocated in order to store the constraint variables was so large that it would cause the program to crash. This issue began to happen, although somewhat inconsistently, at approximately 80 armor components, which would equate to 3,160 constraints (Equation 12). As the number of components grew larger, to over 100 (4,950 constraints), the program would crash consistently.

In order to overcome this limitation, an alternative approach was devised in order to reduce the number of constraints to be equal to the number of armor pieces. This was done by formulating a step function that acted as a single constraint for all armor pieces. This formulation is then:

$$c \geq (2r) * p$$

where:

$$c = \sum_{i=0}^{n} \sum_{j=i+1}^{n} \begin{cases} (\frac{\Delta\theta * s}{\pi})^2 + (\frac{\Delta Z}{z_{max} - z_{min}})^2, & (\frac{\Delta\theta * s}{\pi})^2 + (\frac{\Delta Z}{z_{max} - z_{min}})^2 < 2r \\ 0, & (\frac{\Delta\theta * s}{\pi})^2 + (\frac{\Delta Z}{z_{max} - z_{min}})^2 \geq 2r \end{cases}$$

p = Number of sets armor pieces that are within 2 radiuses

Equation 13: The updated armor overlap constraint function

This formulation prevents overlap of all of the armor pieces that are within a given area of that piece by creating a single constraint as illustrated in Figure 19.

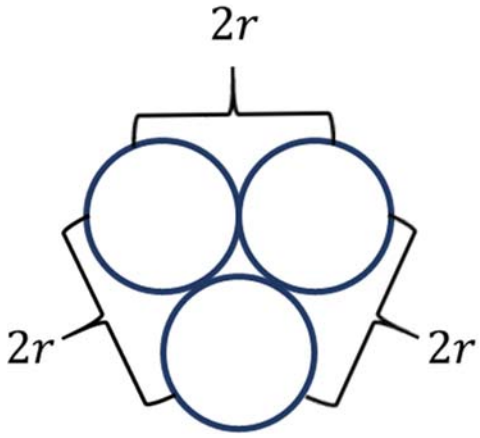

Figure 19: Representation of the overlap constraint, where there are three sets of armor pieces that are within $2r$ and therefore $p = 3$

In this example there are three armor pieces that are all within $2r$ of each other, giving $p = 3$. This means that the lower bound for the constraint is now $2r * 3$ or $6r$. From this, you can see that $c = 6r$ and therefore the constraint is active, and any decrease in distance would mean that the constraint would be violated as the summation of distances between pieces would be less than $6r$. The step function was chosen in order to allow armor pieces to move further away; it is shown in Figure 20.

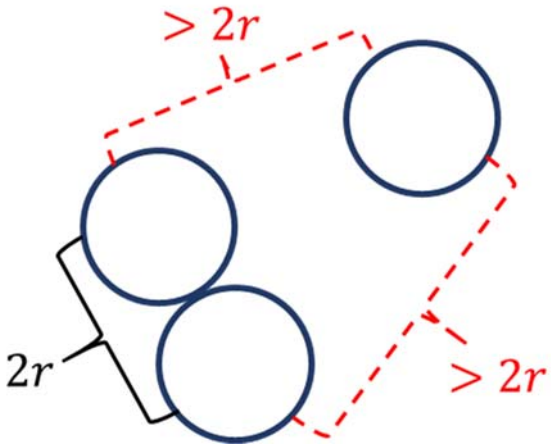

Figure 20: Representation of the overlap constraint, where there is only one set of armor pieces within $2r$ and therefore $p = 1$

In this example, the step function eliminates the summation of the two distances that are greater than $2r$ from other pieces, giving $p = 1$ and therefore a lower bound of the constraint of $2r * 1$ or $2r$. Again, it can be seen that the constraint $c = 2r$ and therefore is active; the pieces cannot move any closer together without the constraint being violated.

4.2.3 Objective Functions

In order to design an armor system using the aforementioned constraints and design variables, an objective function needs to be properly formulated. The goal was to be able to use the types of data and metrics available in the Santos software as the source as described in Section 2.2.2. In order to use this data, it needed to be transformed in such a way that a value could be associated with each location on the Santos avatar. To do this, a

method for transforming point cloud data to meaningful objective functions data was developed. This method can be applied to any user-supplied point cloud and can be executed on the fly. This allows for refinement of optimization gradients, in order to see how they would affect the overall armor system. For example, the user can quickly create a new objective function where coverage of the liver is a priority and the heart is a second or lesser priority. An objective function is calculated for each of the five segments of the body (i.e., torso, right leg, left leg, right arm, and left arm).

The point cloud that is supplied is a collection of three-dimensional points, each with an associated value. The associated values act as the objective function values, where the higher the number is, the more important that location is (e.g., an important coverage location). If multiple three-dimensional points map to a single two-dimensional point, the highest value is taken, as the objective function's goal is to maximize. In order to create a single gradient, a ray detection test is carried out to determine the location on the Santos mesh that is closest to the supplied point. This translates the three-dimensional point cloud data into two-dimensional UV coordinates. Regression analysis is then run on the UV coordinates and their corresponding values in order to provide a smooth and continuous objective function. Once the regression analysis has provided a function, it is used to create an image gradient for every available Santos UV coordinate.

4.2.3.1 Initial Approaches

For the initial approach to automated armor system design optimization, a modified version of coverage was chosen as the objective function. For the original coverage metric in Santos, all areas of the body were treated as the same value of coverage. If this was used as the objective function, the optimization process would simply fill the entire body with armor pieces, with no regard to the importance of the location being covered. In order to introduce meaningful coverage, it is necessary to differentiate between important parts of the body so that, for example, the heart could be valued higher than the elbow. In order to have an objective that varies depending on position relative to the avatar, a weighted coverage system has been implemented that allows one to give priority to the vital areas of the human body based on third-party data.

4.2.3.1.1 Joint Parenting

An initial attempt to obtain a value of the coverage score of an armor component entailed leveraging the existing capability in Santos to determine which joint the armor piece was parented to (as discussed in Section 3.2.2). In the Santos software, the armor pieces are parented to their nearest joint in order to simulate motion of the armor piece with the body (Figure 21). This approach was used as a proof of concept in order to determine if the design variables set in place could be used to move the armor pieces along the surface of the Santos avatar. While this initial attempt was able to serve its purpose of proving

the feasibility of the concept for armor design, there were severe limitations. The approach was tested by applying coverage score values along that spine that increased from the base of the spine until the neck. Although joint parenting has very little to do with actual coverage, this test allowed for quick and accurate validity of the optimization following a fairly straightforward objective function. This approach would ignore the lateral location of the armor component, as the true determining factor would be its latitudinal placement due to the changing of the spine joints. This approach had difficulty due to the gradient approach that is used by SNOPT. As can be seen in Figure 22, the coverage value remains constant over large ranges of Z. These areas where the coverage scores remain constant are locations at which the increase in Z value does not change the joint that the armor component is parented to. These constant ranges do not allow for SNOPT to calculate gradients, and therefore this problem could not be run unless the initial guess was close to a location that would change the parent joint.

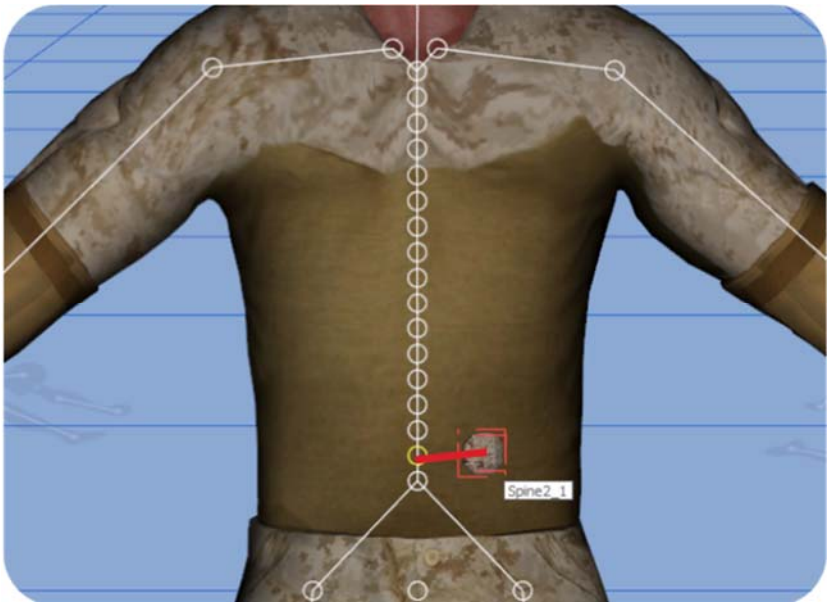

Figure 21: For the initial approach of determining a cover score value, the armor component found its nearest joint, with each joint having a pre-determined value

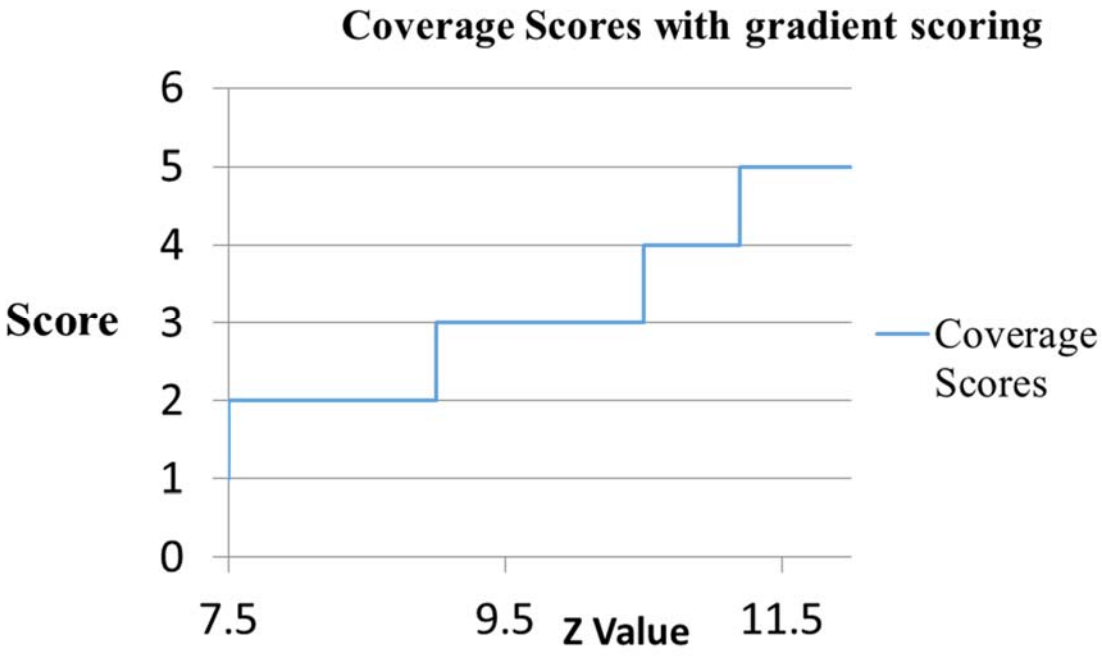

Figure 22: Coverage score with respect to Z variable. As the armor component moved higher up the spine, the coverage value increased

4.2.3.1.2 UV Gradient Mapping

In order to give the coverage function more practical meaning and usability within optimization, organ data provided as three-dimensional point clouds was used. A point cloud is a series of points that has a specific value associated with a three-dimensional position. An example of a point cloud would be blast pressure data of organs within Santos. By using these three-dimensional point clouds, coverage as an objective function implies a coverage of a specifically vital area. In order to use these point clouds, the data needed to be processed in order to create a continuous image gradient. This image gradient is mapped from a three-dimensional position on the body to a two-dimensional position on a texture map. This is done by shooting a collision ray normal to the front and back of the Santos avatar, originating from the three-dimensional position of a point in the cloud. This collision gives the three-dimensional location on the avatar that corresponds to the point in the cloud, and from this location, the two-dimensional mesh location can be computed. The value of the point is then represented as a color, with a higher color value indicating a more valuable location. There is now an associated objective function value for all two-dimensional points of the Santos avatar.

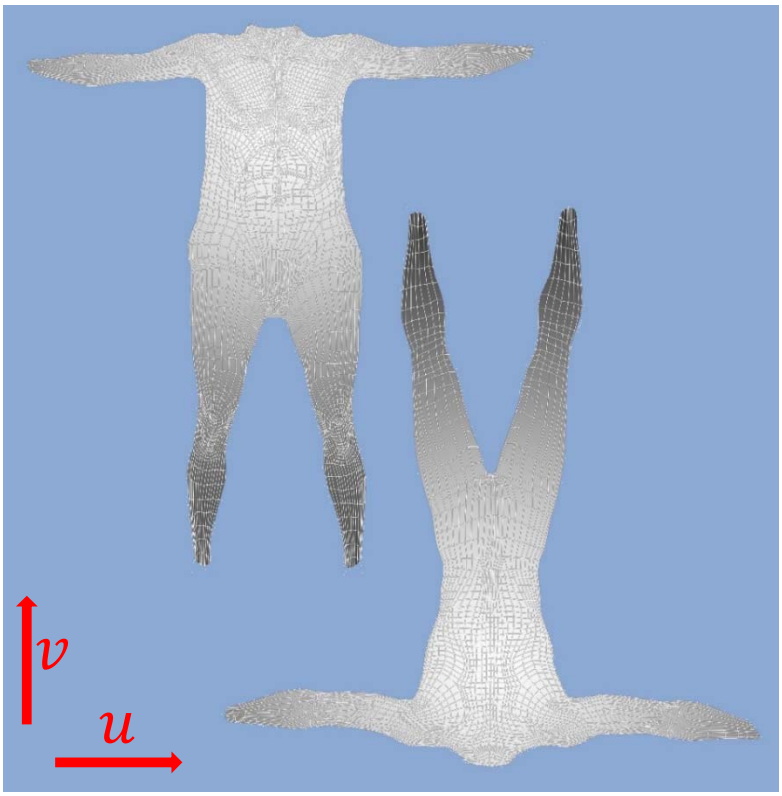

Figure 23: An example of the default gradient once it has been transformed into UV coordinates

4.2.3.1.3 Using a Default Mesh Gradient

In order to ensure that SNOPT can find changes in the objective function value, regardless of starting position, an underlying gradient is necessary. This default gradient (Figure 23) has point cloud data for the entire body and also has an increase in value as the design variables move from the extremities towards the center of the chest. This was used as a simple way to give direction to the optimization problem by implementing a constant change in values. The chest was chosen for the highest values as it is typically

considered to be of most importance. By implementing this underlying gradient in such a way, there are two major benefits. First, optimization will always be able to be performed, as there will never be a lack of data. This prevents the user from having to provide third-party data (e.g., coverage function values, such as organs, in order to run coverage optimization). The second benefit is the filling in of gaps between third-party data. If the situation in which we have three-dimensional point cloud data for only the heart is considered, then there would be the problem of optimizing over any area outside of the provided data, as well as creating a function that guarantees to provide smooth, continuous values that have at least a local maximum where the heart data is provided.

4.2.3.1.4 Creating a Mesh Gradient from a Point Cloud

One of the technical problems associated with this optimization formulation is to transform the external third-party data into a form that can be used as an objective function. The difficulty lies in creating a two-dimensional gradient function from a three-dimensional gradient function. This must be done in order to provide solutions that remain on the surface of the Santos avatar.

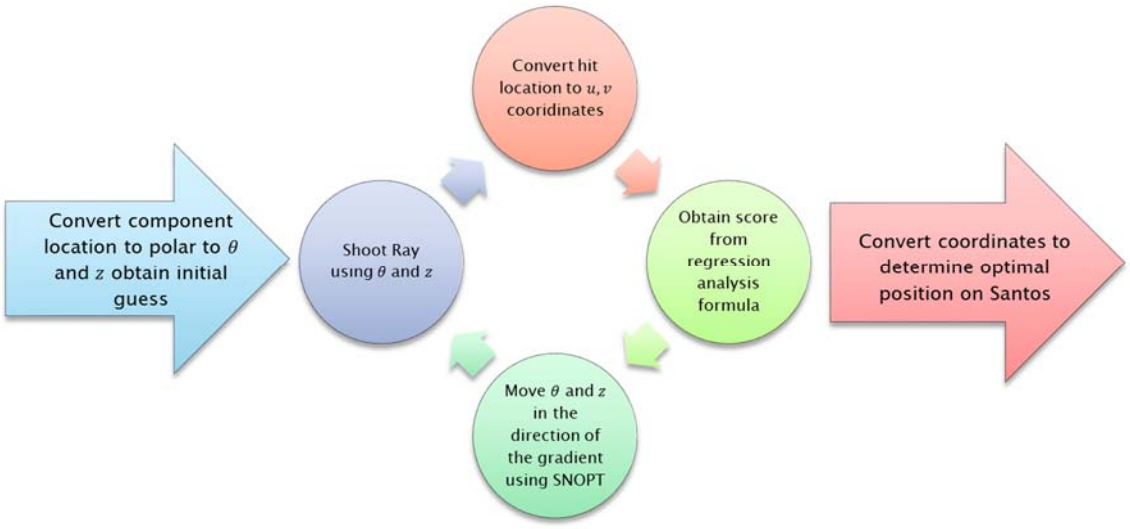

Figure 24: Flow chart of the process of transforming the design variables for the objective function

Creating a two-dimensional gradient from the three-dimensional point cloud data must be done in several steps, with an overview of the process shown in Figure 24. The first step is to locate the three-dimensional position of the avatar's surface that relates to the point cloud point. The three-dimensional surface points directly to the front and back of the avatar, corresponding to the location of a cloud's point. For example, if there is a point cloud for the heart, any individual point in the cloud would map directly to the chest and back of the avatar corresponding to that location. The avatar's three-dimensional surface point is determined by shooting a collision ray in the direction normal to the front as well as a collision ray normal to the back of the avatar. Each three-dimensional point on the avatar is then mapped to a 2D texture map and given the value originally associated to that point from the point cloud data structure. An ongoing list is maintained so that if a value is already associated with the texture map at a given location, only the highest value will be taken. By taking only the highest value, the system takes into account the

important coverage points without being diluted by the underlying default gradient. The process of creating a two-dimensional point and value from the associated three-dimensional point cloud is then performed for all of the data points available. This process will give a texture map as shown in Figure 25.

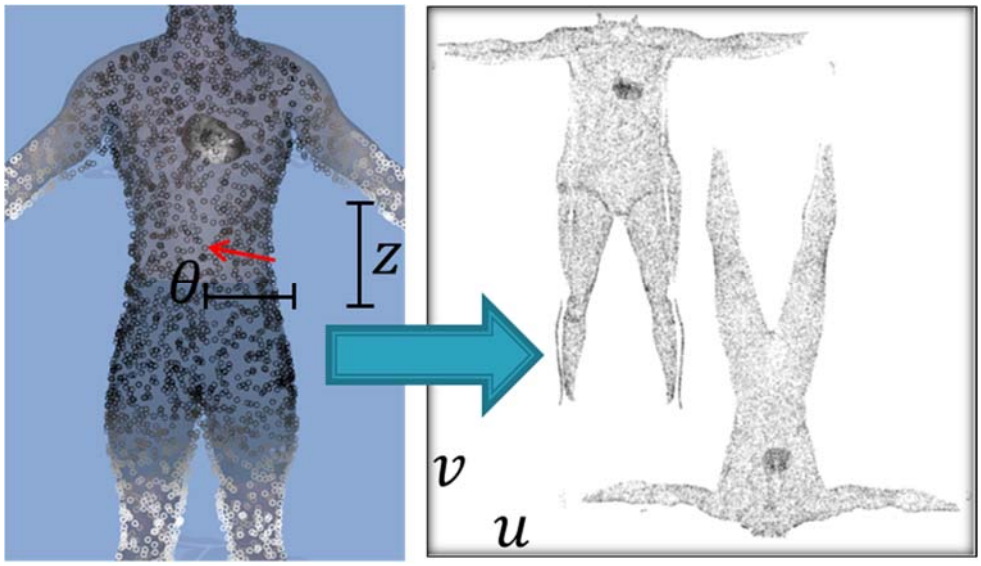

Figure 25: Shooting collision rays from the three-dimensional point position and mapping them to UV space

After the image gradient has been created, optimization can be performed using the design variables and constraints discussed earlier in the chapter. An overview of this process can be seen in Figure 24. First, initial positions are determined and transformed for each armor component i into their corresponding θ_i and z_i design variables. These variables are then used as initial positions for the SNOPT optimization process. SNOPT then performs optimization by systematically varying these design variables in order to

determine a maximum value. The value of the objective function here is determined by the texture mapping values that correspond to each UV position.

4.2.3.2 Regression Analysis

An inherent problem with the data that is being used to generate the objective function in Section 4.2.3.1.4 is that it is not only discrete, it is also likely to have gaps of information missing. This is especially true when the data goes through its necessary transformation into two-dimensional space. An example of this can be seen in Figure 26, where external three-dimensional point cloud data from the heart is transformed into two-dimensional space. The black pixels represent data points from the pressure data of the heart. Not only is the image littered with gaps, but the pixels represent the discrete nature of the problem. In order to overcome this problem, regression analysis was performed on the data. This is done by formulating a separate optimization problem and performing a least squares fit operation with regularization.

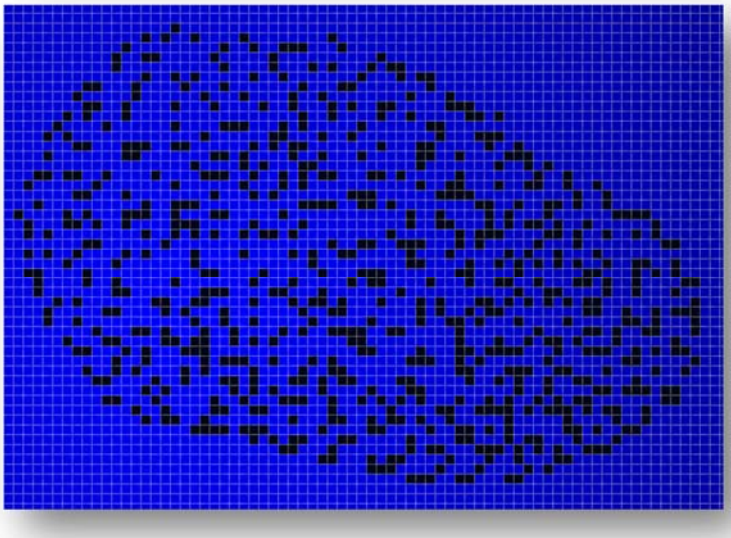

Figure 26: A visual representation of the point cloud data in two-dimensional UV space. When using an image gradient, the problem can no longer be considered continuous due to the discrete nature of the pixels. This also causes gaps where there is no valid information

In order to determine which formula would work best as a least squares approximation, the software package *TableCurve3D* by *Systat Software Inc*. was utilized. TableCurve3D can perform both linear and non-linear surface fitting and can be used to automate the surface fitting into a single processing step. The software fits and ranks 36,000 out of over 450 million built-in commonly used equations, allowing for the ideal model for the three-dimensional data that needed to be fit. Running the software on several different data sets in order to find the ideal equation to surface fit our data, the Order 10 Cosine Bivariate Series was chosen. The equation takes into consideration the range of any data set used and therefore is generalized enough to be used for the use of any third-party data, regardless of range. In order to use the surface fitting equation in the objective functions, it needs to be transformed into the two-dimensional space of the design variables. This

process requires taking the origin point θ and z, obtaining the corresponding u and v coordinate, and running a surface fit with the corresponding score at this location. This process can be visualized in Figure 27. Where the first picture shows the collision ray that is shot from the location of a given point in the cloud, towards Santos to get a three-dimensional position on the surface of the avatar. From this location, the corresponding u and v is known and mapped (shown in the second picture). The third picture shows this information being represented as a surface over which regression analysis is performed and a new objective function is formed. After all of the points have been iterated through, the list is then down-sampled based on a number defined by the user. The option to down-sample the points is to prevent the model from trying to over-fit the number of points in a small space, which leads to a poor regression model, and therefore poor optimization.

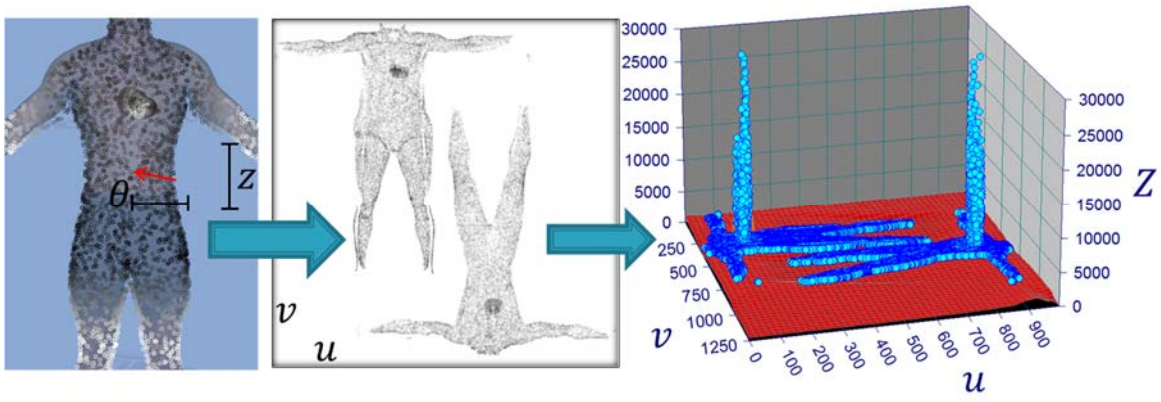

Figure 27: The three-step process of obtaining the objective function from the design variables. The arrow in the first picture indicates the collision ray that is shot towards Santos in order to determine the corresponding UV coordinate

$$
\begin{aligned}
CBS(u,v) \;=\; & a + \mathrm{b}(\cos(\mathrm{u}) \; + c(\cos(\mathrm{v})) + d(\cos(2\mathrm{u})) + e(\cos(\mathrm{u})\cos(\mathrm{v})) + f(\cos(2\mathrm{v})) \\
& + g(\cos(3\mathrm{u})) + h(\cos(2\mathrm{u})\cos(\mathrm{v})) + i(\cos(\mathrm{u})\cos(2\mathrm{v})) + j(\cos(3\mathrm{v})) \\
& + k(\cos(4\mathrm{u})) + l(\cos(3\mathrm{u})\cos(\mathrm{v})) + m(\cos(2\mathrm{u})\cos(2\mathrm{v})) \\
& + n(\cos(\mathrm{u})\cos(3\mathrm{v})) + o(\cos(4\mathrm{v})) + p(\cos(5\mathrm{u})) + q(\cos(4\mathrm{u})\cos(\mathrm{v})) \\
& + r(\cos(3\mathrm{u})\cos(2\mathrm{v})) + s(\cos(2\mathrm{u})\cos(3\mathrm{v})) + t(\cos(\mathrm{u})\cos(4\mathrm{v})) \\
& + v(\cos(5\mathrm{v})) + w(\cos(6\mathrm{u})) + aa(\cos(5\mathrm{u})\cos(\mathrm{v})) + ab(\cos(4\mathrm{u})\cos(2\mathrm{v})) \\
& + ac(\cos(3\mathrm{u})\cos(3\mathrm{v})) + ad(\cos(2\mathrm{u})\cos(4\mathrm{v})) + ae(\cos(\mathrm{u})\cos(5\mathrm{v})) \\
& + af(\cos(6\mathrm{v})) + ag(\cos(7\mathrm{u})) + ah(\cos(6\mathrm{u})\cos(\mathrm{v})) \\
& + ai(\cos(5\mathrm{u})\cos(2\mathrm{v})) + aj(\cos(4\mathrm{u})\cos(3\mathrm{v})) + ak(\cos(3\mathrm{u})\cos(4\mathrm{v})) \\
& + al(\cos(2\mathrm{u})\cos(5\mathrm{v})) + am(\cos(\mathrm{u})\cos(6\mathrm{v})) + an(\cos(7\mathrm{v})) \\
& + ao(\cos(8\mathrm{u})) + ap(\cos(7\mathrm{u})\cos(\mathrm{v})) + aq(\cos(6\mathrm{u})\cos(\mathrm{v}2)) \\
& + ar(\cos(5\mathrm{u})\cos(3\mathrm{v})) + as(\cos(4\mathrm{u})\cos(4\mathrm{v})) + at(\cos(3\mathrm{u})\cos(5\mathrm{v})) \\
& + av(\cos(2\mathrm{u})\cos(6\mathrm{v})) + aw(\cos(\mathrm{u})\cos(7\mathrm{v})) + ba(\cos(8\mathrm{v})) \\
& + bb(\cos(9\mathrm{u})) + bc(\cos(8\mathrm{u})\cos(\mathrm{v})) + bd(\cos(7\mathrm{u})\cos(2\mathrm{v})) \\
& + be(\cos(6\mathrm{u})\cos(3\mathrm{v})) + bf(\cos(5\mathrm{u})\cos(4\mathrm{v})) + bg(\cos(4\mathrm{u})\cos(5\mathrm{v})) \\
& + bh(\cos(3\mathrm{u})\cos(6\mathrm{v})) + bi(\cos(2\mathrm{u})\cos(7\mathrm{v})) + bj(\cos(\mathrm{u})\cos(8\mathrm{v})) \\
& + bk(\cos(9\mathrm{v})) + bl(\cos(10\mathrm{u})) + bm(\cos(9\mathrm{u})\cos(\mathrm{v})) \\
& + bn(\cos(8\mathrm{u})\cos(2\mathrm{v})) + bo(\cos(7\mathrm{u})\cos(3\mathrm{v})) + bp(\cos(6\mathrm{u})\cos(4\mathrm{v})) \\
& + bq(\cos(5\mathrm{u})\cos(5\mathrm{v})) + br(\cos(4\mathrm{u})\cos(6\mathrm{v})) + bs(\cos(3\mathrm{u})\cos(7\mathrm{v})) \\
& + bt(\cos(2\mathrm{u})\cos(8\mathrm{v})) + bv(\cos(\mathrm{u})\cos(9\mathrm{v})) + bw(\cos(10\mathrm{v}))
\end{aligned}
$$

Equation 14: Order 10 Cosine Bivariate series used in regression analysis

The Order 10 Cosine Bivariate Series (CBS) in Equation 14 is set into a least squares fit shown in Equation 15. This equation becomes a sub-problem to the overall optimization problem that must be solved in order to properly formulate the main objective function. In this problem, we are trying minimize the relative error between the known value at a given u-v point and the result of the formula at that same u-v point. This is done by using a vector of the coefficients $\boldsymbol{w}$ (i.e., $a, b, c, d, \ldots bw$) in Equation 14 as the design variables in Equation 15.

Find: $\boldsymbol{w}$

to minimize: ε

where:

$$\varepsilon = \sqrt{\sum_{n=1}^{N} (CBS(u_n, v_n) - \ val_n)^2}$$

CBS = Equation 14

$\boldsymbol{w}$ = A vector of coefficients used in the CBS function

N = Number of data points over which to fit

u_n = u coordinate value of point n

v_n = v coordinate value of point n

val_n = Score value of point n

Equation 15: Regression Analysis formula

Initially, there was a problem with over-fitting of the function, which resulted in poor optimal solutions, so regularization was necessary. Regularization is a technique used to control over-fitting by adding a penalty term in order to discourage coefficients from growing too large. This is a technique that is discussed extensively in the book *Pattern Recognition and Machine Learning Vol. 1* (Bishop, 2006). The simplest of such techniques is performed by summing the squares of all coefficients, leading to the generalized regression formula in Equation 16.

$$\sum_{n=1}^{N} (f(x_n, w) - t_n)^2 - \lambda * \boldsymbol{w}^2$$

where:

x_n = a data point n

$\boldsymbol{w}$ = a vector of adjustable parameters

t_n = value at a given point n

λ = regularization coefficient

Equation 16: Least squares with regularization in order to prevent over-fitting

The coefficient λ is used to define the relative importance of the regularization term compared to the sum-of-squares term. Substituting Equation 15 into Equation 16, we get Equation 17.

Find: $\boldsymbol{w}$

to minimize: ε

where:

$$f(u_n, v_n) = \sqrt{\sum_{n=1}^{N} (CBS(u_n, v_n) - val_n)^2} - \lambda * \boldsymbol{w}^2$$

$\boldsymbol{w}$ = a vector of coefficients used in the CBS function

N = number of data points over which to fit

u_n = u coordinate value of point n

v_n = v coordinate value of point n

val_n = score value of point n

λ = regularization coefficient

Equation 17: Regression Analysis performed with regularization

In this equation, λ is set to a default of 0.5 but can be changed in the software. This sub-problem is solved using SNOPT, with the design variables being the coefficients of Equation 14. The optimization runs very quickly, as analytical gradients are available for the problem, with an example shown in Equation 18.

$$\frac{df}{dc} = \frac{\cos(v)\sqrt{\sum_{i=0}^{n}\frac{(CBS(u_i, v_i) - val)^2}{n}}}{\sum_{i=0}^{n}\frac{(CBS(u_i, v_i) - val)^2}{n}}$$

where:

f = the function in Equation 17

c = a coefficient in Equation 14, designated above as the vector $\boldsymbol{w}$

Equation 18: Example gradient for the coefficient c in the regression analysis formulation

4.2.3.3 Coverage Objective Function

After finalizing which approach would be taken for the optimization process, the final objective function was formed for coverage. The final formulation takes the form:

Find: z_n, θ_n

to maximize: $Val = f(\theta_i, z_i) = \sum_{i=0}^{n}(CoverageValue(Hit\ Detection(\theta_i, z_i))$

subject to:

$$0 \leq z_i \leq l_i$$

$$0 \leq \theta_i \leq 2\pi$$

$$c_i \geq (r_i + r_j) * p$$

where:

$$c_i = \sum_{i=0}^{n}\sum_{j=i+1}^{n}\begin{cases}(\theta_i - \theta_j)^2 + (z_i - z_j)^2, & (\theta_i - \theta_j)^2 + (z_i - z_j)^2 < r * 6 \\ 0, & (\theta_i - \theta_j)^2 + (z_i - z_j)^2 \geq r * 6\end{cases}$$

$$Hit\ detection\ (\theta_i\ ,\ z_i) = u_i, v_i$$

$$CoverageValue(u_i, v_i) = Cosine\ Bivariate\ Order\ 10\ Series(u_i, v_i)$$

l_i = the length of the cylinder around Santos

n = the number of armor components

z_i = the height of the origin location of the ray

θ_i = the angle that determines the lateral location of the origin ray

r = the radius of an armor component

Equation 19: Armor coverage objective function

In this formulation the function known as $Hit\ detection$ is the ray collision that is performed using the initial θ and z position to obtain the corresponding u and v values on the Santos mesh.

4.2.3.4 Weight Function

In order to provide a method with which to remove pieces and thus yields 1) a method for integrating weight criteria and 2) more accurate design results, an existence variable was created. The variable was added in conjunction with the weight function, as it is a metric that can be validated subjectively. If the existence variable were used with only the coverage function, the optimization process would attempt to add as many armor components as possible until the body was covered, as this would maximize the objective function value. By adding the existence variable along with the weight objective function,

there is now a system in place that will minimize the number of armor pieces. This is done by placing a maximum limit of the total weight of the armor system, and allowing the optimization process to determine the weight of each armor component in order to satisfy this constraint. This new existence variable was added to the original coverage function value, as the weight of an armor piece will directly affect the coverage of a given armor component. This gives the formulation:

Find: z_n, θ_n, e_n

to maximize: $Val = f(\theta_i, z_i) = \sum_{i=0}^{n}(CoverageValue(Hit\ detection\ (z_i,\ \theta_i)) * e_i$

subject to:

$$0 \leq z_i \leq l_i$$

$$0 \leq \theta_i \leq 2\pi$$

$$c_i \geq (r_i + r_j) * p$$

$$0 \leq e_i \leq 1.5$$

$$\sum_{i=0}^{n} e_{i * w_i} \leq w_{max}$$

where:

$$Hit\ detection\ (z_i,\ \theta_i) = u_i, v_i$$

$$CoverageValue(u_i, v_i) = Cosine\ Bivariate\ Order\ 10\ Series(u_i, v_i)$$

l_i: the length of the cylinder around Santos

n: the number of armor components

z_i: the height of the origin location of the ray

θ_i: the angle that determines the lateral location of the origin ray

r : the radius of an armor component

$$c_i = \sum_{i=0}^{n} \sum_{j=i+1}^{n} \begin{cases} (\theta_i - \theta_j)^2 + (z_i - z_j)^2, & (\theta_i - \theta_j)^2 + (z_i - z_j)^2 < r * 6 \\ 0, & (\theta_i - \theta_j)^2 + (z_i - z_j)^2 \geq r * 6 \end{cases}$$

e_i: the existence variable of an armor component

w_i: the weight of an armor component

w_{max}: the maximum weight as defined by the user

Equation 20: Coverage/weight objective function

The existence variable e_i has a subjectively determined upper bound of 1.5, allowing the objective function to determine if the armor piece should be increased in weight. Various other numbers were tested, but if the number was too large, armor pieces would become unreasonably heavy at the optimal point, and all other armor pieces would be removed. Thus, the number 1.5 was chosen to allow the pieces to increase in size, but cap their growth in order to ensure the existence of other armor pieces. This would occur in the situation where increasing the weight would not exceed the overall maximum weight, and also if the specific armor component's particular coverage score is high enough to warrant the trade-off. The final value of each armor component's existence variable multiplied by its initial weight is returned to Santos, and the new weight is applied. If the existence variable of any armor component reaches zero, that armor piece is removed from the system.

4.2.3.5 Mobility Function

One of the benefits, and therefore the reason for choosing the point-cloud method, is that it allows for different data types to be incorporated into a single objective function. This can be done by following the same process that was performed for the coverage objective function (i.e., transforming the point cloud into the design variables and running regression analysis to create a surface fit equation from the data) and including the new surface fitted objective function in addition to any other previously computed objective functions. This method was tested using a simplistic form of the mobility metric. The mobility metric can be thought of as how much the armor inhibits the motion of a task that Santos is attempting to perform. These motion restrictions are most often caused by armor pieces being too near joints that need to move in a certain direction. In order to simulate this, a point cloud was created for the arms that focused on the mobility of the elbow. The point cloud was intended to recreate the fact that if range of motion was to be maximized while trying to add armor components to the arm, it would be optimal to restrict pieces of armor from interfering with the elbow motion. The equation then for the mobility objective function can be found by substituting the range of motion function for the coverage function in Equation 19, giving:

Find: $z_n,\ \theta_n, e_n$

To maximize:

$$f(\theta_i, z_i) = \sum_{i=0}^{n} ROMValue\big(Hit\ detection\ (z_i,\ \theta_i)\big)$$

Equation 21: Mobility objective function

In order to use this objective function alongside the coverage function, all that needs to be done is to include the additional surface fit to any other objective functions that exist. Using the example of combining the coverage function, the weight function, and the newly formed mobility function, we can expand

Equation 20 to be:

Find: $\mathrm{z_n},\ \theta_\mathrm{n}, e_n$

to maximize:

$$f(\theta_i, z_i) = \sum_{i=0}^{n} \left[\left(CoverageValue\left(Hit\ detection\ (z_i,\ \theta_i)\right)\right) * e_i\right] + ROMValue\left(Hit\ detection\ (z_i,\ \theta_i)\right)$$

Equation 22: Objective function that takes into consideration coverage, weight, and mobility

This formulation requires no new additional design variables and is subject to all of the previous bounds and constraints. The new function $ROMValue\left(Hit\ detection\ (z_i,\ \theta_i)\right)$ is the newly formulated surface fit of the point cloud data representing the range-of-motion results. Both $ROMValue$ and $CoverageValue$ functions are determined through the regression analysis performed with their respective data sets and using the formulation in Equation 17. In the data that was used throughout the testing process, the magnitude of the data was consistent. However, in the future it may be necessary to

normalize data sets in order to prevent high-magnitude data sets dominating the objective function.

4.2.4 Whole-Body Optimization

The optimization process detailed above is for a single-segment optimization. Due to the nature of the cylinder approach being used, five cylinders must be created for the torso, right leg, left leg, right arm, and left arm. In order to perform whole-body optimization, the single-segment optimization is applied to each of the five different body segments. These results are then optimized in order to obtain the overall optimal whole-body armor solution. This is done by finding the n maximum objective function values and locations regardless of the segment, where n is the number of armor components. For example, if there are four armor components, and the maximum four possible objective function values of each component are a combination of three being on the chest and one being on the arm, this will be the resulting armor design. Thus, the whole-body armor optimization process is the result of running each segment individually and then selecting the optimal combination of their results. An overview of the psuedocode that was implemented is as follows:

1. Run optimization on all n components on a single segment and return an array of scores
2. Run all n components on any other segments, again returning each individual score

3. Find the highest n scores of all arrays (e.g., if there are four components, and the highest score on the right arm segment is higher than the fourth-highest score on the torso segment, that component is added to the list)
4. Return the components to their highest possible positions

This method allows for little modification of the original segment optimization in order to provide whole-body optimization. This method does, however, have the con of requiring five separate optimization problems to be performed in order to obtain a solution. This could be addressed in the future by using parallel processing to perform these optimizations concurrently, which would greatly reduce computational time. The other major drawback of this method is that it provides for very poor initial guesses. For example, when an optimization process is performed on the arm, and all of the armor pieces originated on the torso, the initial guess will essentially be random.

4.2.5 Sensitivity Analysis

The method put forth in this chapter is also useful to supply the user with information that may allow for specific design decisions and trade-off analysis. For this reason, the added capability of sensitivity analysis was implemented and can be used to increase the understanding of relationships between the input and output variables of the system or model (Prannel, 1996). For this thesis, the following three types of sensitivity analysis are extracted and discussed:

1. Objective function value
2. Lagrange multipliers
3. Derivative values of the design variables with respect to the objective function

4.2.5.1 Armor Component Objective Function Value

By determining which pieces contributed least to the objective function score, one can decide which pieces to remove if such a case were necessary. With respect to the highest objective function score, it is possible to identify which pieces are the most critical to the design. In order to make use of this information, each individual armor piece's component of the overall objective function must be maintained and returned to the user. The individual objective function score for a given armor component can be derived from Equation 19 by examining a component's score from the summation of the overall objective function and would be defined as:

$$Val = f(\theta, z) = (CoverageValue(Hit\ Detection(\theta, z))$$

Equation 23: An individual armor component's score that contributes to the overall objective function

where the $CoverageValue$ function can be replaced by any of the other metrics' objective function (e.g., mobility objective function formulated in Equation 21).

Initial results of the sensitivity analysis were positive and useful. To showcase this, Figure 28 shows the data set used to create the coverage objective function (Equation 19) that was used for an example. This point cloud was transformed into an objective function using the approach and methodology laid out in the previous sections of the chapter. The values of the points in the cloud are highest for the white points and lowest for the black. The points here are meant to represent a blast pressure value at a given point. This data is an example of the type that could be supplied by the user.

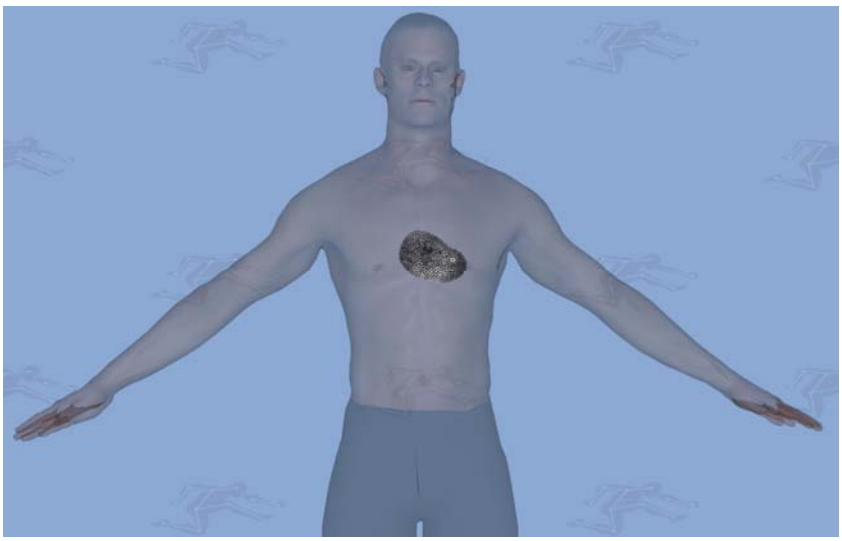

Figure 28: The point cloud of the heart was used as the basis for the objective function

Coverage optimization was then performed, and the software was used to highlight the three pieces with the highest objective function value. These results can be seen in Figure 29, with the numerical results found in Table 1. Although the objective function values were derived from example data and therefore have no specified units, the results would be similar for data that the user provided. Table 1 shows a range of 299 to 385 for the

objective function value scores. The range here reported to the designer would indicate how important it is to maintain all of the armor components in the system. This information also tells the designer how much each component contributes to the overall design of the armor system. For example, the armor component of index 0 can be considered 28% more important to coverage than the component of index 11.

Comparing the returned and highlighted pieces in Figure 29 to the basis for the objective function in Figure 28, one can see that the pieces line up nicely with the section of the point cloud with the highest area of white points, which you will recall indicate the highest objective function values. These results indicate to the designer that the three armor pieces highlighted are of the highest importance to the design.

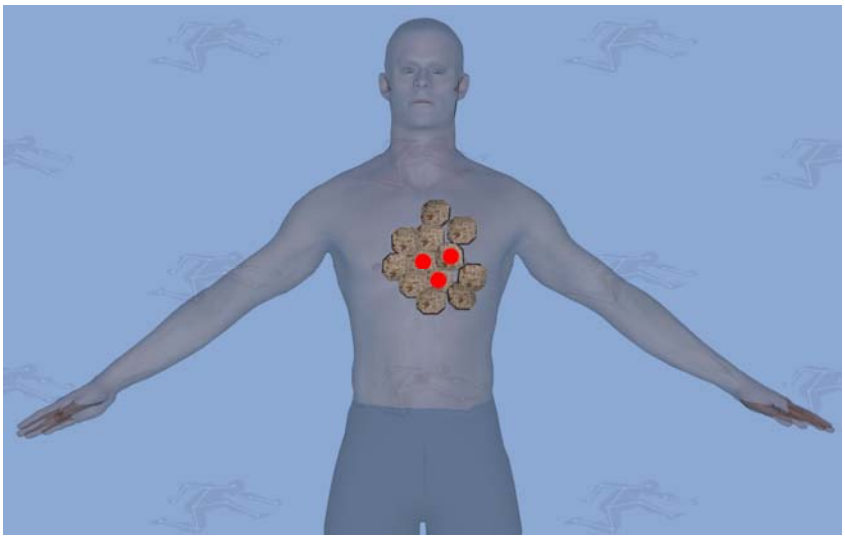

Figure 29: The sensitivity analysis with the highest three armor component objective score values being highlighted

Armor ID	Coverage Score
[0]	385.0281982
[1]	369.5085144
[2]	354.5787354
[3]	350.3672485
[4]	326.8458557
[5]	322.7375183
[6]	316.9371643
[7]	313.1019287
[8]	312.1017456
[9]	305.940155
[10]	304.3232422
[11]	299.6908264

Table 1: Numerical results for each armor component's averaged derivative score

4.2.5.2 Derivative of Objective Function with Respect to Design Variables

The other piece of information that is useful to a designer is the pieces with the highest sensitivity values. The higher value of the derivative implies that being able to change the position of the piece even slightly would result in the highest variation in the

objective function score. This could either mean an increase or a decrease (indicated by the sign of the score) in the objective function score, which can indicate how one might want to move components. In order to give this information simplicity and still retain its meaning, the two design variables were combined into a single value. This is important for indicating which armor components we could move to get the greatest effect. Since the location of the armor component is determined by both design variables, we can combine them and simplify the information to indicate which piece needs to be moved. Although the design variables are continuous and smooth, the objective function does not have analytical gradients due to the process of transforming the variables from three-dimensional space into two-dimensional space. The gradients of each design variable must be obtained from SNOPT, which uses a finite difference approximation. The average of the derivatives of the objective function with respect to each design variable would come from the objective function in Equation 23 and would be:

$$Averaged\ Derivative\ Score = \frac{\frac{\partial val}{\partial \theta} + \frac{\partial val}{\partial z}}{2}$$

where:

$$Val = CoverageValue(Hit\ Detection(\theta, z)$$

$$\frac{\partial val}{\partial \theta} = \frac{\partial CoverageValue}{\partial Hit\ Detection(\theta, z)} \frac{\partial HitDetection(\theta, z)}{\partial \theta}$$

$$\frac{\partial val}{\partial z} = \frac{\partial CoverageValue}{\partial Hit\ Detection(\theta, z)} \frac{\partial HitDetection(\theta, z)}{\partial z}$$

Equation 24: The averaged derivative score used for sensitivity analysis

A similar process was performed as in the two previous sections in order to highlight the armor pieces that correspond to the highest design variable derivatives with respect to the coverage objective function (Equation 19) and is illustrated in Figure 30. Comparing Figure 30 and Figure 28, one can see that the pieces with the highest derivative values lay on the edge of the point cloud values. This is due to the sharp decrease in values from the edge of the point cloud to the surrounding defaulting point cloud gradient. The reasons that these components specifically had the highest gradients can be explained by looking at the results of Figure 29. In this figure, the highest objective function values are located on the left side (looking at it from the picture's perspective). This would mean that the armor pieces would be going from the highest values of the heart point cloud pressure data to the lower values of the default gradient values. This indicates to the designer that if they were able to move these armor pieces (this may be done by changing the amount of overlap allowed), you would see the highest change in your overall objective function score. The armor pieces and their corresponding scores are also shown below in Table 2. In this table, the sign of the value indicates the direction of the change in the objective function due to the change in the design variables. This means that by modifying a given design variable, you can note the magnitude and direction of its effect on the objective function value. This would allow a designer to know the significance of a change in design variable (e.g., an armor component moved slightly vertically) on the objective function value.

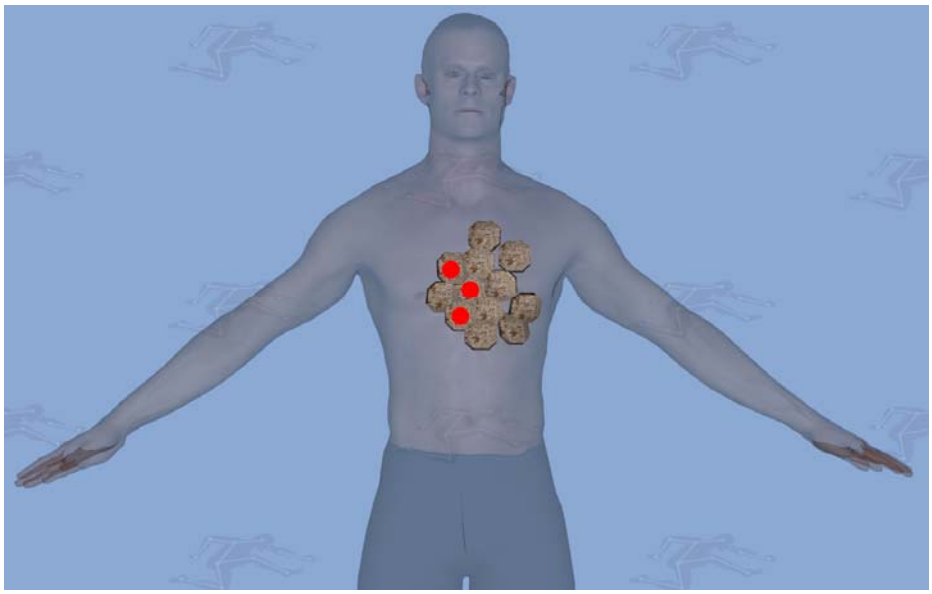

Figure 30: The sensitivity analysis with the highest three armor component derivative values being highlighted

Armor ID	Averaged Derivative Value
[0]	14.59407139
[1]	35.23088074
[2]	-60.46815109
[3]	-30.32169724
[4]	84.32984924
[5]	-27.18102074
[6]	35.0343399
[7]	51.26932144
[8]	37.26494217
[9]	188.4612885
[10]	20.94153976
[11]	-8.592450142

Table 2: Numerical results for each armor component's averaged derivative score

4.2.5.3 Lagrange Multipliers

Another valuable piece of information that is typically considered in sensitivity analysis is the Lagrange multiplier for each constraint. There is a single Lagrange multiplier for each of the constraints in the problem. The value of the Lagrange multiplier at the solution is equal to the rate of change in the optimal value of the objective function as the constraint is relaxed. Lagrange multipliers in the case at hand would tell the designer

which constraints to relax in order to see the largest increase in the objective function value. SNOPT is able to output the Lagrange multipliers for every problem that it runs. An output from a problem again using the coverage objective function (Equation 19) with 12 armor pieces can be seen in Figure 31.

Nonlin constrnt	State	Value	Lower bound	Upper bound	Lagr multiplier	Slack
nlncon 1	FR	4.068991	.	6.021956	.	1.953
nlncon 2	I ++	5.698981	.	5.687245	21.94565	-.1174E-01
nlncon 3	FR	3.050319	.	5.338847	.	2.289
nlncon 4	FR	2.021939	.	4.974575	.	2.022
nlncon 5	FR	4.544199	.	4.591565	.	.4737E-01
nlncon 6	FR	3.318494	.	4.185937	.	.8674
nlncon 7	FR	2.026959	.	3.752183	.	1.725
nlncon 8	FR	3.231309	.	3.281995	.	.5069E-01
nlncon 9	I ++	2.761710	.	2.761689	13.99296	-.2172E-04
nlncon 10	D FR	.0000000+000	.	2.165309	.	.
nlncon 11	D FR	.0000000+000	.	1.428571	.	.
nlncon 12	FR	.1284227E-02	.	.7000000E-01	.	.1284E-02

Figure 31: Example output from SNOPT output file with the Lagrange multiplier for each constraint being highlighted

From this output, it can be seen that both constraint 2 and constraint 9 are active. This means that this armor overlap constraint (as defined in Section 4.2.2) is being enforced and that no armor components involved in this constraint can be moved any closer together. Here, the larger the Lagrange multiplier, the larger that constraint's effect on the overall objective function value, and in this case, the coverage and survivability of the armor system. Specifically, in the example above, constraint 2 has a larger effect on the system than constraint 9. Since the constraints being shown here are with regards to overlap (i.e., the distance between armor components and therefore their location in the final solution), what the high value of the Lagrange multiplier for constraint 2 means is that by allowing for more overlap along this constraint, a greater objective function could

be achieved. Another way to interpret this result is to say that if the designer were to add more material over a certain spot (i.e., by either allowing for more overlap or by increasing the material density at this location), the greatest positive increase of the optimal value would occur.

4.2.6 Graphical User Interface (GUI)

In order for a user to import point cloud data to use as an objective function, manipulate data, select optimization settings, and perform the optimization, a visualizer was created within the Santos software and can be seen in Figure 32.

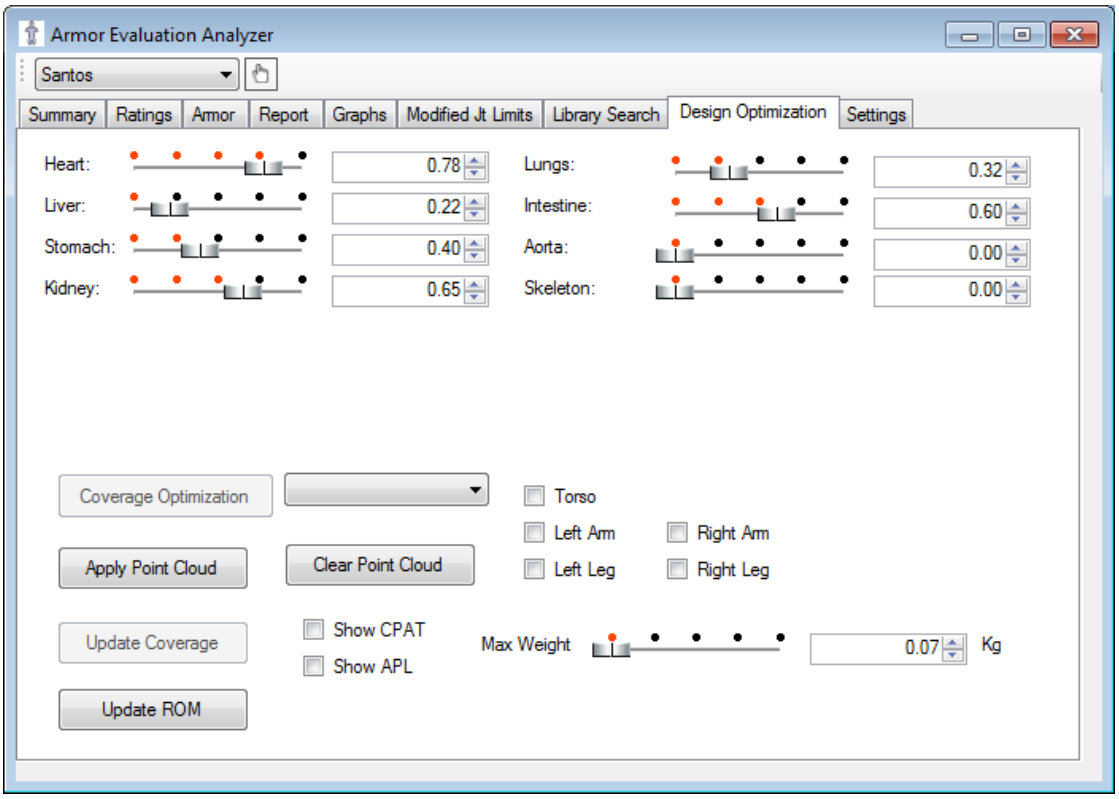

Figure 32: The GUI used to formulate the objective functions from the point-cloud data, as well as to run the system design optimization

The user can then use a series of numerical sliders in order to add/remove emphasis to certain parts of the point cloud. If the user increases the value of the slider, the values of the points in the cloud associated with that section are increased. When regression analysis is performed, any changes made to the point cloud values will be considered, with a greater value implying greater importance. These weights are then part of the equation that is used for the objective function. From this, the value for any point in a cloud is defined as:

$$v_w = v_i * w_c$$

where:

c = A point cloud group (e.g., heart, lungs, liver, etc.)

v_w = weighted value of a point in a point cloud c

v_i = initial value of a point in a point cloud c

w_c = weight of the point cloud c

Equation 25: Equation to determine the weighted value of a point in a point cloud using the GUI

The GUI has buttons to load or clear point cloud data on the Santos avatar, along with checkboxes to enable/disable the visibility of the clouds. The visibility for individual organs can also be toggled by setting their respective weights to zero, as this would imply that they are not used in the objective function.

In the example shown below, the user is able to increase/decrease the weight of a series of organs for which blast pressure point cloud was available. The user can also select over which body segments the design optimization should take place (i.e., torso, right arm, left arm, right leg, and left leg). The final element of the GUI is the numerical slider that can be used by the user in order set a maximum weight of the overall armor system. A suggested maximum weight for a whole-body system of 15 KG was discussed in Section 2.2.2. This weight is then set as the upper boundary constraint for the weight objective function.

4.3 Results

This section lays out select results from the three objective functions (i.e., coverage, weight, and mobility) that were discussed earlier in this chapter. These results provide the proof of concept and highlight the limitations of the current approach

.

4.3.1 Coverage

The first objective function that was formulated and therefore tested was the coverage objective function discussed in Section 4.2.3.1. Referring back to the GUI (Figure 32) that was implemented and discussed in Section 4.2.6, one can recall that the user has the ability to add emphasis to certain organs by applying a weighting factor during the pre-processing stage of the regression analysis step (Equation 25). This allows for any desired variation of an objective function using a single data set. A set of examples that use the formulation in Equation 19 as the objective function can be seen below.

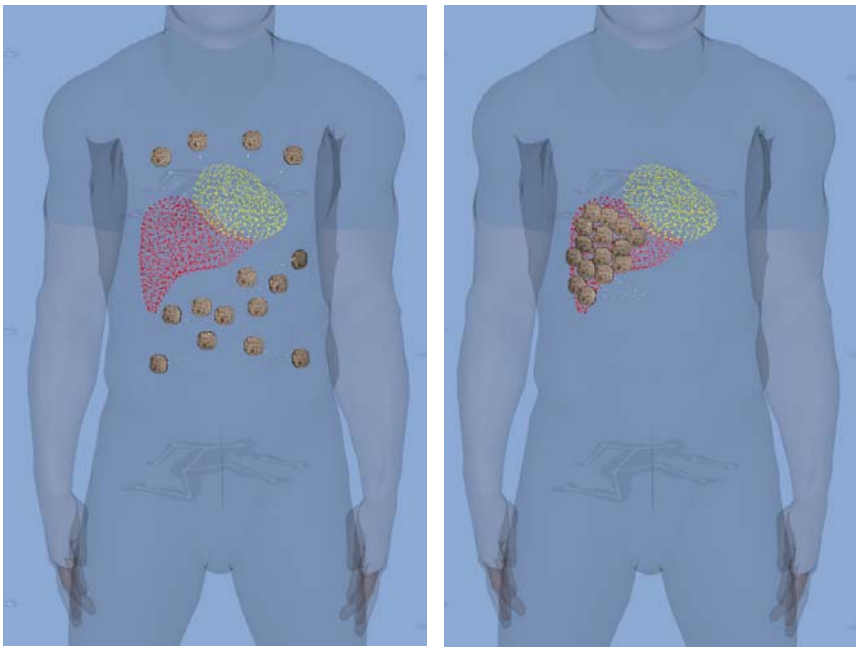

Figure 33: The liver (red) is set as a higher priority than the heart (yellow), and the armor pieces are optimized to cover the maximum score

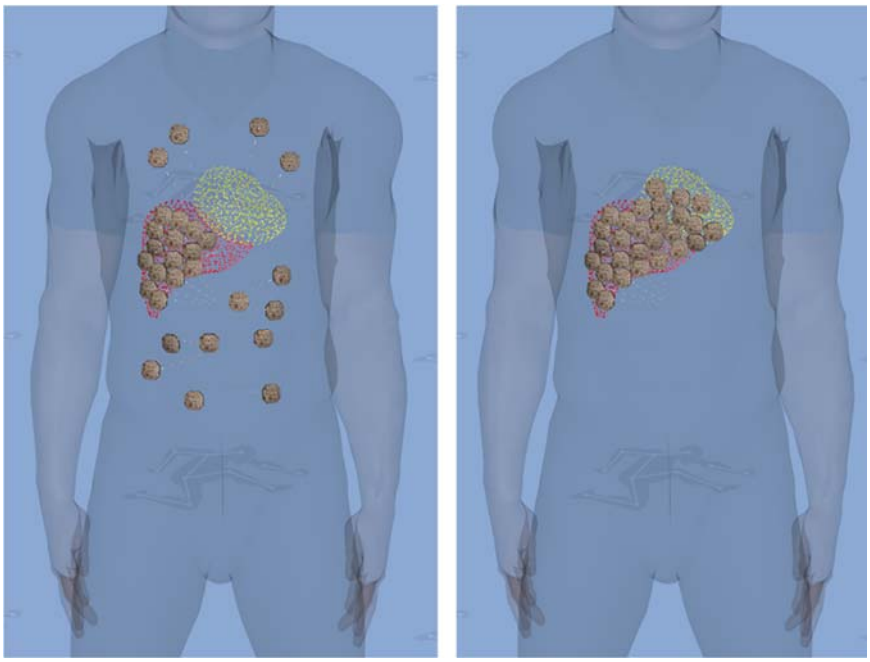

Figure 34: Adding additional armor components to the design allows for coverage of the heart after the liver has been covered

In Figure 33, the point clouds of the organs were colored in order to clarify the different organs. For this example, the heart and liver were the only two organs that were considered. The weightings were set such that the liver was of more significance. The figure shows an initial guess, with the resultant optimization solution. Since there were not enough armor components to cover both organs, they clustered at the more important liver. Adding additional armor components (shown in Figure 34) allows for the components to move to the heart after fully covering the liver.

There is also an example using the same number of components with two differently weighted objective functions. For this example, four organs were weighted differently for two different test cases. The organ point clouds for the heart, liver, stomach, and kidneys can be seen in Figure 35. For each example, the weights were set, regression analysis performed to create an objective function, and the armor components' locations used as design variables in order to optimize that objective function. The results of these two examples, along with the weights used for the objective function, can be seen in Figure 36 and Figure 37. Both examples consist of 28 armor components, which equates to 56 design variables and the single overlap constraint (Equation 13).

Figure 35: The organ point clouds that were used and weighted for the results in Figure 36 and Figure 37

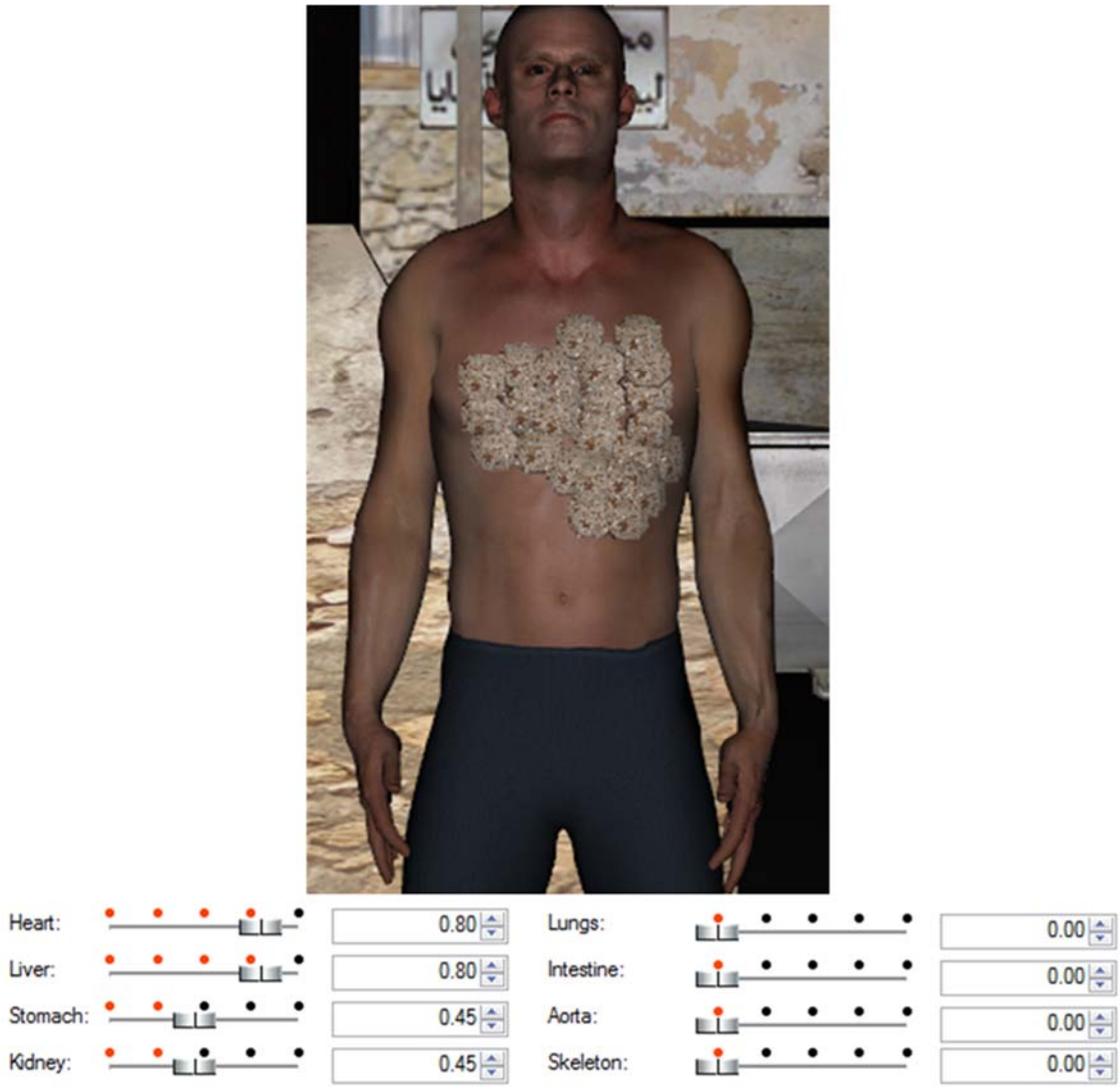

Figure 36: Examples of the coverage objective function with point cloud data being more heavily weighted on the stomach and kidney than the heart and liver

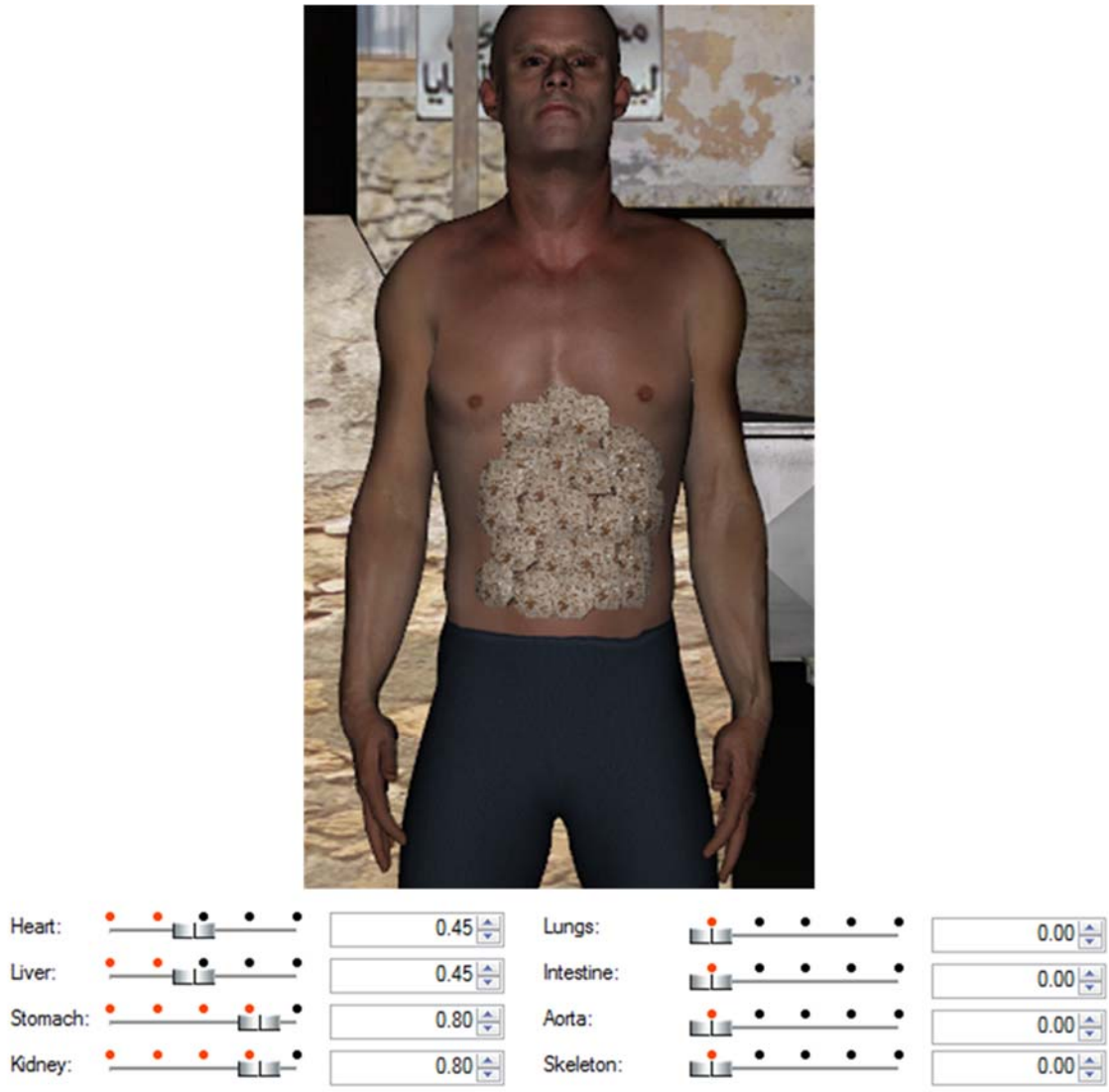

Figure 37: Example of the coverage objective function with point cloud data being more heavily weighted on stomach and kidney than the heart and liver

In Figure 36, the weightings for the heart and liver were set to almost twice the weightings of the stomach and kidney. This was done in order to emphasize protection of the heart and liver, while still formulating the objective function such that armor components would trend towards the stomach and kidneys. The results show that the

armor system is focused mainly over the two primary organs, with pieces continuing to the secondary organs.

In Figure 37, the same number of pieces was used, but the weighing factors were varied to emphasize the importance of the stomach and kidneys over the heart and lungs. From this, it can be seen that the armor system stays intact, but the armor components obtain a higher objective function value by covering the higher weighted stomach and kidney organs.

In both examples, notice that the armors touch, but no overlap of any concern occurs. This overlap prevention remains enforced and consistent in all examples.

4.3.2 Existence

As was discussed in Section 4.2.3.4, the weight function and constraint are implemented in addition to the coverage function. This is due to the fact that without a competing objective function, to optimize weight would be to remove all armor pieces from the system. Since this result would provide no new information, it is performed in conjunction with the coverage function. These two competing functions attempt to have as many armor components as possible in order to maximize coverage, while at the same time minimizing weight in order to satisfy any weight. An example of armor coverage

and weight optimization can be seen in Figure 38. This example has 70 armor components and, with the new design variable along with the two location design variables, the problem now has 210 design variables. There is now an additional constraint for the weight, along with the single overlap constraint, which equates to 2 constraints. All of the examples shown below use the formulation in Equation 20.

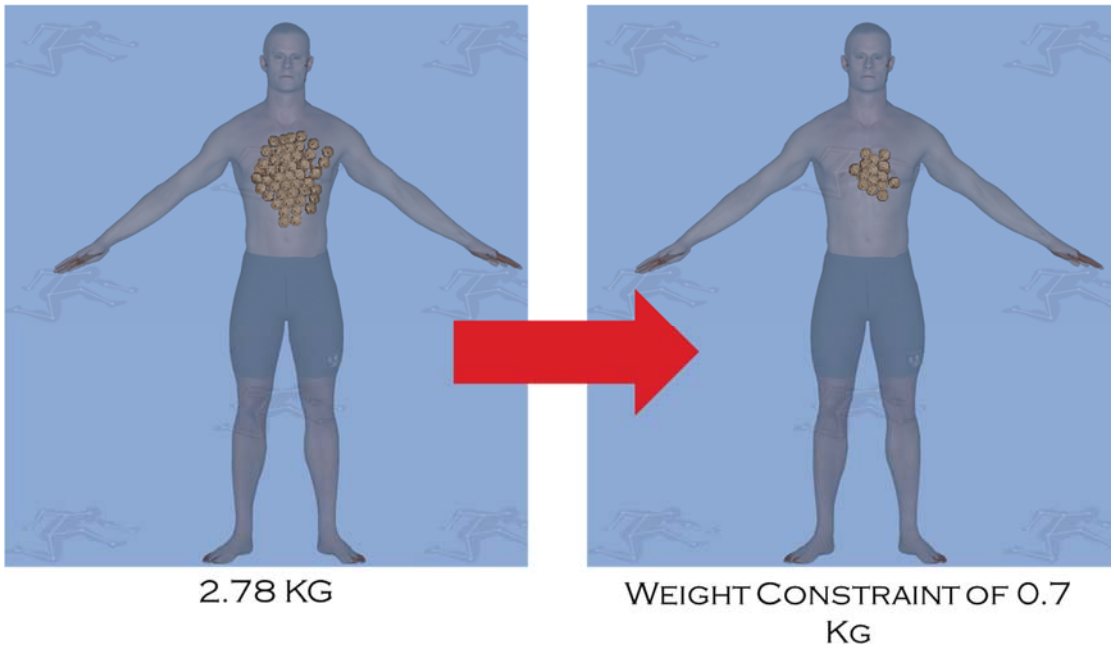

Figure 38: Example of the weight constraint being applied during optimization. The coverage function in this function was focused around the heart

In this example, the armor coverage function was focused around the heart. In order to satisfy the weight constraint, the 2.78 KG of armor were reduced to 0.7 KG, with the remaining armor components being placed on the heart. Additionally, an example using

whole-body optimization is shown in Figure 39. As noted in Section 3.2.4.6, an average weight for a whole-body armor system is approximately 15 KG.

Figure 39: Full-body coverage with the picture on the right showing a decreased weight constraint and therefore armor components on the shoulders

In this example, the same initial guess was used, but the weight constraint was changed just slightly (from 7 KG to 6.5 KG), such that only a small portion of the armor components were removed. For the case shown, the arms had the lowest coverage value, and therefore the components were removed only from there, keeping the rest of the

design intact. Coverage optimization still occurs when the weight constraint is reduced, but the armor components tend to stay in their previous positions due to SNOPT having found a local maximum in the previous solution. The problem consisted of 100 armor components, which equates to 300 design variables and 2 constraints (overlap and weight). This formulation, which has 2 constraints, is using the new constraint formulation in Equation 13; when using the previous formulation of Equation 11, it equates to 4,951 constraints using Equation 12.

4.3.3 Mobility

This set of examples showcases the addition of the range-of-motion objective function discussed in Section 4.2.3.5. For this example, a point cloud was created in such a way as to provide an academic example of the results that could be produced by a range-of-motion metric calculation. The point cloud was created to maximize the mobility over the arms. This was done by giving the points around the shoulder, elbow, and wrist the lowest objective function values. The reasoning behind this is that armor pieces around the joints can cause the biggest restriction in motion. The initial point cloud and results are shown below in Figure 40. Equation 21 is the objective function for this example.

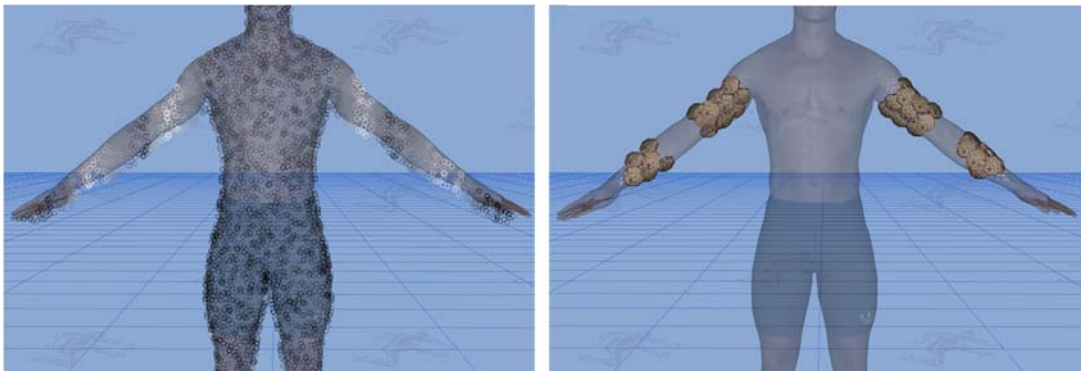

Figure 40: Initial point cloud and results of mobility of arms as an objective function

The initial point cloud from which the objective function is created is shown, with the lighter color points indicating a higher objective score. For this example, only the arm segments were considered due to the primary concern being the ability to place armor pieces on the arms, but in such a way as to minimize the restriction of mobility. The results shown place the armor pieces in such a way as to cover all available areas of the arm, while still avoiding placement over the mobility restriction areas of the joints. Although the weight constraint is set to the default 15 KG and therefore has no effect on the solution shown, it is still a constraint, and the existence variables are still included. With the 44 armor pieces used in this problem, there are 132 design variables and 2 constraints (again using the second constraint formulation of Equation 13).

.

4.3.4 Multi-Objective Optimization (MOO)

The final set of examples showcases the use of both coverage and mobility as objective functions as discussed in Equation 22. By including the mobility in addition to the coverage function, these results showcase an academic use of multi-objective optimization. For this, an example point cloud was created to mimic the type of data that could be representative of future work that would automatically generate point clouds from armor system metrics. These clouds would allow for less intuitive design problems to be addressed (e.g., more complex data of range of motion, balance, torque, etc.). The cloud that was created was done so in order to represent the decrease in range of motion that would occur when armor components were located too close to either the elbows or the shoulders. This mobility objective function was combined with the coverage function that focused on the heart, as well as covering the upper areas of the arms.

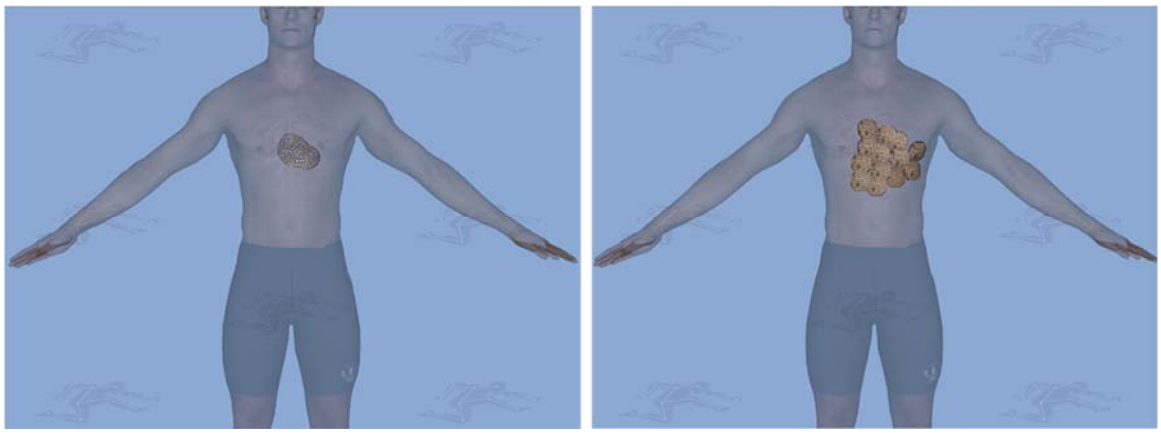

Figure 41: Initial result with only coverage of the heart as the objective function

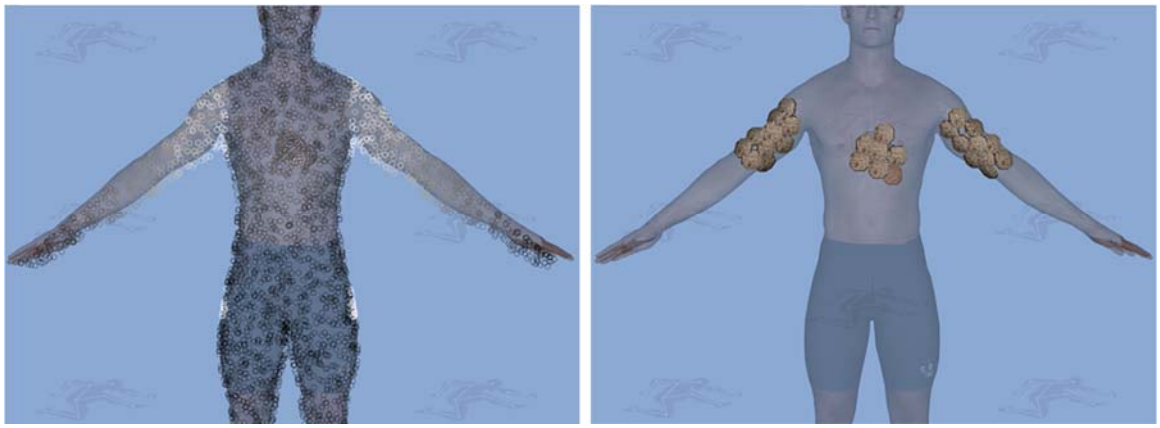

Figure 42: Whole-body results with coverage as an objective function. The armor components move to the arms

Figure 43: Including the range-of-motion objective function in addition to the coverage function pushes the armor components away from the elbow and further down the shoulder

For initial results, armor components were optimized solely over the torso, with the solution placing the components over the heart, as expected (Figure 41). In this example, there were 14 armor components, 42 design variables, and 2 constraints. More armor components were then added, and the arms were included in the objective function, but range of motion is still not being considered (Figure 42). This example now has 28 armor pieces, 84 design variables, and 2 constraints.

This example also showcases that, once the arms are included, some of the armor components leave the torso and move to the shoulders. This is due to the high values being set for the arm coverage; after the heart has been covered, the optimal placement for components is on the arm.

Finally, the range-of-motion objective function was included (Figure 43). In this result, you can see the armor pieces being pushed away from the elbow and shoulder, in order to maximize the range of motion of these joints. The final example adds no additional armor pieces, so the number of design variables and constraints remains 84 and 2, respectively.

CHAPTER 5

CONCLUSION

5.1 Summary

The work and approaches discussed and shown in this thesis address a gap that exists in the way that armor systems are evaluated, chosen, and designed. The primary goal was to leverage existing digital human modeling capabilities within Santos in order to provide a systematic, efficient, and measurably improved way to design, as well as to compare, armor systems.

It should be noted that the continuous optimization approach focused on method development, and the examples provided are meant to be academic. It is intended to provide one piece of the puzzle in order to perform more intensive, whole-body armor system design. Besides the usefulness of the tools used, there are two significant contributions. The first contribution was an approach for taking the three-dimensional surface of Santos and transforming it into a two-dimensional, bounded, continuous space over which the surface could be optimized. When this problem was originally formulated, there were no methods for formulating a problem where the variables would only allow for placement on a location along Santos's body. This method of transforming variables alleviates this problem in a non-intuitive, yet simple and effective way. The main problem that existed with using this approach was the lack of analytical

gradients for the gradient-based optimization. Although using the cylinder approach allows for continuous and smooth design variables, there is still no mathematical function for this transformation due to the nature of the hit detection method. This is a problem that is discussed further in Section 5.3 with possible solutions.

Another novel contribution was an alternative approach to topology optimization, as mentioned in Section 1.2.5. In a typical topology optimization problem, the design variables directly affect the existence of a number of finite pieces of materials in order to create a structural design and abide by load and space constraints. This leads to the problem of checkerboarding whereby enough pieces remain to satisfy the formulated problems constraints, but the design is either infeasible from a manufacturing standpoint or provides results that are observably not optimal. This problem is caused by errors in the finite element formulation. The similarity to this problem that can be drawn to the armor system design optimization would be an area of extremely small armor components that are meant to cover an area of interest (e.g., the heart), but has a significant amount of pieces that were removed, creating an armor component system that is not truly optimal for coverage and may not even be possible from a manufacturing and materials standpoint. In order to overcome this, the addition of location was included to the design variables in the form of θ and z, which allowed for the armor pieces to be located strictly on the surface of the avatar. By allowing the finite number of materials to both determine existence and change location, optimal design tends to relocate the pieces into areas of local optima.

5.2 Discussion

This work presented progress in all areas intended to be addressed at the onset of the work. Nonetheless, there were several challenges, especially with regards to the system design optimization. There is a clear gap in the current state of the art for optimal armor design with respect to human factors, and there is room for improvement in this area.

The work of the armor system optimization filter provided a tool that could be used to make decisions more effectively. This tool allows for a wide variety of armor systems to be designed and selected as needed, based on situational criteria, instead of a single, all-purpose design. The tool is easy to use and is effective for its intended purpose. In order for the system to be as effective as possible, it is important that the armor system library is thoroughly populated. If the system is able to select from a very large number of options, it will increase the chances of finding a globally optimal solution. The approach was also coded in such a way as to allow for the easy insertion of newly developed metrics. This allows the Santos software either to improve the computational time of the existing metrics or possibly to implement any newly devised metrics that may be useful. The length and complexity of tasks that are rendered on the avatar also increases the computational time. As the software continues to develop and these times decrease, so will the time for the armor system optimization filter.

Although the armor system design optimization was shown to be feasible, advances still must be made in order to provide full functionality and usefulness. The biggest challenge in this area was formulation of the design variables. It was clear that defining the problem in such a way that it would be a continuous, differentiable, smooth problem that was also bounded to the Santos avatar was not going to be a simple task. While the approach taken did allow for the necessary transformation from three-dimensional space into two-dimensional space, there is still room for improvement in the method that was discussed in this thesis. The design variables were able to be formulated in such a way that they were continuous but no longer differentiable due to the hit detection transformation not being able to be formulated.

One of the main problems is that the transformation through hit detection does not allow for analytical gradients. The lack of analytical gradients causes two problems. The first problem is that this requires SNOPT to perform finite difference approximation after each iteration, in order to determine the gradient of the objective function. With each armor component having three design variables, as the number of armor pieces grows, this process becomes computationally expensive. This causes the run-time for optimization to become unreasonably long, in the magnitude of hours, for large problems. A large problem here would be defined as whole-body optimization starting around approximately 100 armor components.

The other issue that stems from the lack of analytical gradients is that, as problems grow large, there are situations where SNOPT fails to find an optimal solution, especially with

poor initial guesses. In these cases, SNOPT exits and gives an error message describing that "the current point cannot be improved." Presently, initial guesses are determined by the current location of the armor components on the Santos avatar. This can lead to a very poor initial guess for the torso segment if all armor components were previously placed on, for example, the leg segment. There may be benefits from attempting a method for improved initial guesses, which would help alleviate both the time issue and the failed solutions. An example of improving the initial guess would be to move the armor pieces to cover a segment without overlapping and then begin the optimization. However, both of these problems could be greatly improved by the use of analytical gradients. The analytical gradients would be able to be obtained from further research into a smooth, continuously differentiable two-dimensional representation of the three-dimensional points along the surface of the Santos avatar.

The first hypothesis was that it was possible for existing armor systems to be down-selected based on performance during task simulation; this was proven true. The results were positive, and the tool that was created has immediate potential to be used for optimal armor selection. The method is especially useful due to its generalized nature. Since any task that can be rendered can be considered, the armor selection has the potential to be used to determine the optimal armor for either a specific task or a very wide range of tasks. The armor systems library can also be continually populated, which allows for future expansion. The tool is most useful if the library is to serve as an ever-expanding database of successful armor designs. By providing more potential solutions, there is a greater chance of finding a globally optimal solution. However, even with a

smaller library size, it was easy to see the potential usefulness of this tool, and it worked as was predicted.

The second hypothesis was that new armor systems can be automatically designed based on user-specified criteria and third-party objectives. This hypothesis, although being more technically challenging to prove than initially anticipated, was also found to be true. The difficulties were not in the process of optimization of the objective function formed from the data, as a method for this was successfully developed. Rather, the difficulty was in formulating the design variables and objective function in a way that would be continuous along the Santos avatar. This complication led to an increase in both complexity and computational time. That being said, the hypothesis was proven true, and the concept was executed and found to be usable. The results showcased real potential for the use of this tool and also provided informative and accurate results. While the original hypothesis was proven true, the method that was used in the approach could be revisited. Possible new approaches will be discussed in Section 5.3.

The third hypothesis was that predictive DHM capabilities and product performance could be integrated and embedded within an overarching optimization process. Stating the hypothesis in another way, it is possible to use the Santos software to perform continuous optimization of a human systems integration design problem. This hypothesis was proven true, as a method for performing continuous optimization was formulated and tested with initial results.

Currently, HSI processes that use DHM capabilities focus on optimizing over a discrete set of options. The armor system optimization design filter is an example of this process. When discrete optimization is used, the quality of the solutions depends primarily on the available solutions that can be selected. While increasing the number of choices will increase the possibility for a globally optimal solution, continuous optimization is the preferred approach. In continuous optimization, the number of possible solutions no longer matters because solutions can now be determined on the fly. Instead of optimizing an HSI process using a discrete number of choices, this thesis has shown that is possible to leverage DHM software not just to determine between a finite set of options, but to design options within the realm of the human factors measured in the software. This idea can be extended to not just determining, for example, the optimal placement of a drive shaft in a car between three different predetermined choices, but also to use DHM capabilities to design the exact placement and height of a drive shaft through continuous optimal design. By leveraging the DHM capabilities within an overarching optimization loop, it has been shown that designs can be improved into the scope of globally optimal as opposed to locally optimal.

5.3 Future Work

One of the primary goals that would be addressed in future work is to re-evaluate the transformation of design variables in the objective function for design optimization. The

method discussed in this paper provided a different approach to the discontinuity of variables problem, and this approach can still be leveraged and approved. While some sort of transformation is necessary in order to provide a continuously bounded function, it may be possible to formulate the problem such that analytical gradients of the design variables can be used. This would be done by creating a UV mapping of a texture in such a way that all UV coordinates map to a location on an avatar. In the current method of UV mapping, there are many holes and seams where a given UV coordinate does not map to any location on the avatar. This current discontinuity prevents the UV coordinates from being used as design variables.

One of the biggest factors in the high computational time is the lack of analytical gradients, as SNOPT is forced to perform finite difference approximation with respect to each design variable. As the number of armor pieces approaches a large amount, this becomes extremely time intensive. Analytical gradients could not be used with the two-dimensional UV coordinate mapping, as there were "seams" in the function, causing discontinuity. It may be possible to use a different three-dimensional to two-dimensional mapping system that allows the two-dimensional space version of the problem to be entirely continuous and differentiable. If this were possible, the problem could be formulated such that the design variables are within the two-dimensional space, and analytical gradients could be used. This would allow for much faster optimization times.

Another area of work would be to automate the process of transforming the armor system metrics into usable point cloud data. An example of this would be to have Santos

perform a task and record the areas that go through range of motion throughout the task. This data would then be formulated into a point-cloud and that point-cloud used as the objective function. This process could be used for all of the available armor metrics, allowing for quick and meaningful objective functions.

A last area of potential work would be to use real-time armor optimization. A recent addition to Santos performs a series of hit detections with a detailed penetration model. This hit detection is done in real time and maintains information as to whether a hit collides with an armor piece, an organ, or the body. An example of this can be seen in Figure 44. If this data could be transformed quickly enough, it would be possible to try to optimize the armor pieces in time with the data that is being provided from hit detection.

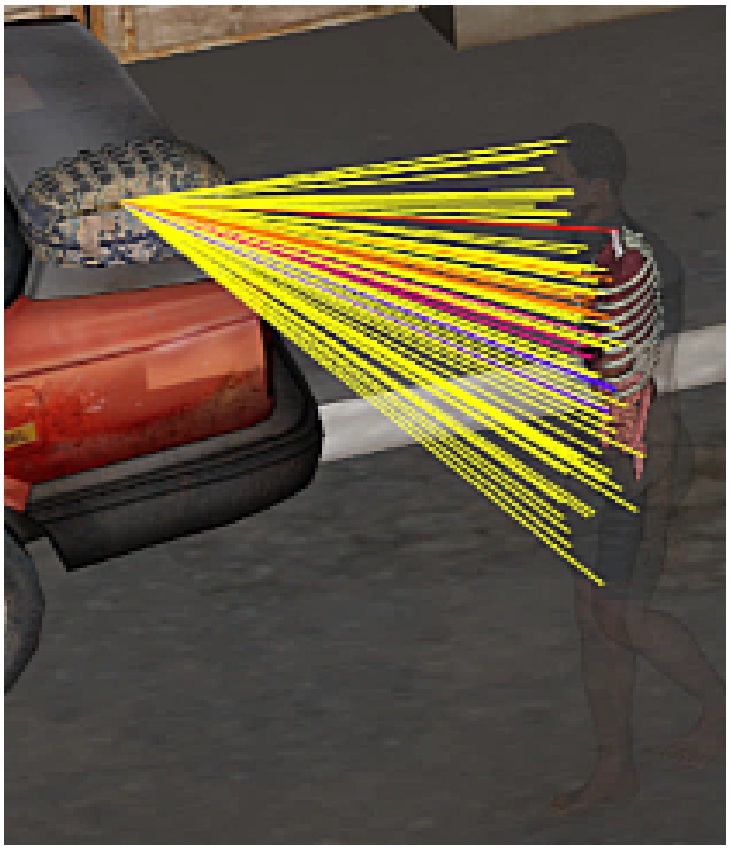

Figure 44: Real time hit-detection and damage score in Santos with each color of the ray indicating a hit of a different body part

REFERENCES

Adams, P. S., Slocum, A. C., & Keyserling, W. M. (1994). A model for protective clothing effects on performance. *International Journal of Clothing Science and Technology*, 6-16.

Bendsoe, M. P., & Sigmund, O. (2003). *Topology optimization: theory, methods and applications.* Springer.

Bialas, W. F., & Karwan, M. H. (1982). On Two-Level Optimization. *IEEE Transactions On Automatic Control, Vol AC-27*, 211-14.

Bishop, C. M. (2006). *Pattern recognition and machine learning. Vol. 1*. New York: Springer.

Booher, H. R. (2003). *Handbook of human systems integration* (Vol. 23). John Wiley & Sons.

Chan, Y. (2011). *Location Theory and Decision Analysis: Analytics of Spatial Information Technology*.

Chang, M., Low, S. H., Calderbank, A. R., & Doyle, J. (2007). *Layering as Optimization Decomposition: A Mathmetical Theory of Network Architectures.*

Chateauroux, E., & Xuguang, W. (2010). *Car egress analysis of younger and older drivers for motion simulation.* Applied ergonomics.

Denavit, J., & Hartenberg, R. S. (1955). A Kinematic Notation for Lower-pair Mechanisms Based on Matrices. *Journal of Applied Mechanics,, 77*, 215-221.

Drezner, Z., Klamroth, K., Schobe, A., & Wesolowsky, G. (2002). *The Weber Problem, Facility Location: Application and Theory.*

Farahani, R. Z., Steadieselfi, M., & Asgari, N. (2010, July). Multiple Crteria Facilty Logation Problems: A Survey. *Applied Mathematical Modelling, 34*(7), 1689–1709.

Faraway, J. (2000). Modeling reach motions using functional regression analysis. *Digital Human Modeling for Design and Engineering Conference and Exposition.* Michigan.

Francis, R. L., Jr., F. M., & White, J. A. (1991). *Facility Layout and Location: An Analytical Approach.*

Fricker, R. D., & Wilson, A. G. (2010). Assessing the Methodology for Testing Body Armor.

Gama, B. A., Bogetti, T. A., Fink, B. K., Yu, C.-J., Claar, T. D., Eifert, H. H., & Jr., J. W. (2001). *Aluminum Foam Integral Armor: A New Dimension in Armor Design.*

Gill, P., Murray, W., & Saunders, A. (2002). SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal of Optimization, 12*(4), 97-1006.

Goldfarb, M. A., Ciurej, T. F., Weinstein, M. A., & Metker, L. W. (1975). *A Method for Soft Body Armor Evaluation: Medical Assessment.* NTIS.

Gu, Z. (2006). Study on the principle of soft complex stab-resistant body armor. *Journal of Textile Research.*

Gwynne, S. M., & Kuligowski, E. D. (2010). *Application modes of egress simulation.* Pedestrian and evacuation dynamics.

Harman, E., Frykman, P., Pandorf, C., Tharion, W., & Mello, R. (1999). *Physiological, Biomechanical, and Maximal Performance Comparisons of Soldiers Carrying Loads Using U.S. Marine Corps Modular Lightweight Load- Carrying Equipment (MOLLE), and U.S. Army Modular Load System (MLS).* Technical Report, Army Research Institute of Environmental Medicine, Military Performance Division, Natick.

Hasselquist, L., Bensel, C. K., Corner, B., Gregorczyk, K. N., & Schiffman, J. M. (2008). *Understanding The Physiological, Biomechanical, And Performance Effects Of Body Armor Use.* Natick Soldier Research,, Development and Engineering Center,, Natick.

Hennessy, E. R., & Zielinski, M. R. (2006). *Results of Benchmarking Ergonomics Evaluation of Explosive Ordanance Disposal (EOD) Personal Protective Equipment (PPE) Standard Program.* NATICK.

Jiménez, P., Thomas, F., & Torras, C. (2001). 3D collision detection: a survey. *Computers & Graphics*, 269-285.

Johnson, R., Smith, B. L., Penmatsa, R., Marler, T., & Abdel-Malek, K. (2009). Real-Time Obstacle Avoidance for Posture Prediction. *3rd International Conference on Applied Human Factors and Ergonomics.* Miami: AHFE International.

Kaliraj, M., Narayanasamy, P., Rajkumar, M., Mohaideen, M. M., & Manickam, I. N. (2014). Deisgn, Fabrication and Analysis of Advanced Polymer Based Kevlar-49. *Applied Mechanics and Materials*, 122-127.

Knapik, J. J., Reynolds, K. L., & Harman, E. (2004). Soldier load carriage: historical, physiological, biomechanical, and medical aspects. *Military Medicine*, 45-56.

Kockara, S., Halic, T., Iqbal, K., Bayrak, C., & Rowe, R. (2007). Collision detection: A survey. *Systems, Man and Cybernetics*, 4046-4051.

Li, T.-T., Wang, R., Lou, C.-W., & Lin, J.-H. (2013). Static and dynamic puncture behaviors of compound fabrics. *Composites: Part B*, 60-66.

Li, X., Ding, L., Hedge, A., Hu, H., Qin, Z., & Zhou, Q. (2013). Optimizing the physical ergonomics indices for the use of partial pressure suits. *Applied Ergonomics*, 393-403.

Lin, M., & Gottschalk, S. (1998). Collision detection between geometric models: A survey. *Proc. of IMA conference on mathematics of surfaces*, 602-608.

Lou, C. W., Hsing, W. H., Hsieh, C. T., & Lin, H. L. (2013). Mechanical and Physical Property Evaluations of Kevlar/Polyester Complex Nonwoven Fabrics. *Applied Mechanics and Materials*, 61-64.

Mahbub, R., Wang, L., & Arnold, L. (2014). Design of knitted three-dimensional seamless female body armour vests. *International Journal of Fashion Design, Technology and Education*, 198-207.

Marler, R. T., Rahmatalla, S., Shanahan, M., & Abdel-Malek, K. (2005). A New Discomfort Function for Optimization-Based Posture Prediction. *SAE Human Modeling for Design and Engineering Conference.*

Marler, T. R., Arora, J. S., Yang, J., Kim, J., & Abdel-Malek, K. (2009). Use of Multi-objective Optimization for Digital Human Posture Prediction. *Engineering Optimization 10, no. 41*.

Marler, T., Arora, J., Beck, S., Lu, J., Mathai, A., Patrick, A., & Swan, C. (2008). Computational Approaches in DHM. *In Handbook of Digital Human Modleing for Human Factors and Ergonomics*.

Marler, T., Knake, L., & Johnson, R. (2011). Optimization-Based Posture Prediction for Analysis of Box-Lifting Tasks. *3rd International Conference on Digital Human Modeling*.

Marler, T., Mathai, A., Johnson, R., & Taylor, A. (2012). *New Methods for Modeling the Biomechanical Effects of Body Armor Systems.* Personal Armor Systems Symposium.

Mathai, A., Marler, T., Farrell, K., Meusch, J., Taylor, A., Beck, S., . . . MacKiewicz, J. (2010). A New Armor Simulation and Evaluation Toolkit. *Personal Armor Systems Symposium*.

Metker, L. W., Prather, R. N., Coon, P. A., Swann, C. L., & Hopkins, C. E. (1978). *A Method of Soft Body Armor Evaluation: Cardiac Testing.* ARMY ARMAMENT RESEARCH AND DEVELOPMENT COMMAND ABERDEEN PROVING GROUND MD CHE MICAL SYSTEMS LAB.

Migdalas, A., Pardalos, P. M., & Värbrand, P. (1998). *Multilevel Optimization: Algorithms and Applications.* Springer.

Moujahed, S., Simonin, O., & Koukam, A. (2009). *Location Problems Optimization by a Self-Organizing Multiagent Approach.*

Mukai, T., Kuriyama, S., & Kaneko, T. (2006). Keyframe Animation of Virtual Humans via Motion Data Learning. *Systems and Computers in Japan, 37*(5).

Murray, S. L., Simon, Y. L., & Sheng, H. (2011). The effects of chemical protective suits on human performance. *Journal of Loss Prevention in the Process Industries*, 774-779.

Prannel, D. J. (1996). Sensitivity analysis of normative economic models: Theoretical framework and practical strategies, Agricultural Economics. *Agricultural Economics*, 139-152.

Reza, F. Z., Abedian, M., & Sharahi, S. (2009). Dynamic Facility Location Problem. In *Facility Location Concepts, Models, Algorithms and Case Studies* (pp. 347-372). Springer.

Salvendy, G. (2012). *Handbook of human factors and ergonomics.* John Wiley & Sons.

Sigmund, O., & Petersson, J. (1998). Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural optimization*, 68-75.

Valença, S., Griza, S., Oliveira, V., Sussuchi, E., & Cunha, F. (2014). Evaluation of the mechanical behavior of epoxy composite reinforced with kevlar plain fabric and glass/kevlar hybrid fabric. *Composites: Part B.* doi:http://dx.doi.org/10.1016/j.compositesb.2014.09.040

Wilkins, M., Cline, C., & Honodel, C. (1969). *Fourth Progress Report Of Light Armour Programs.*

Wilkins, M., Cline, C., & Honodel, C. (1978). *Mechanics Of Penetration and Perforation.*

Wilkins, M., Cline, C., & Honodel, C. (1980). *Computer Simulation of Penetration Phenomeon.*

Zhou, M., Shyy, Y. K., & Thomas, H. L. (2001). Checkerboard and minimum member size control in topology optimization. *Structural and Multidisciplinary Optimization*, 152-158.