

CONTACTLESS INTERFACING WITH DICOM  
USING LEAP MOTION

by

Brian Notarianni  
A Thesis  
Submitted to the  
Graduate Faculty  
of  
George Mason University  
In Partial fulfillment of  
The Requirements for the Degree  
of  
Master of Science  
Computer Science

Committee:

_____	Dr. Zoran Duric, Thesis Director
_____	Dr. Wilsaan Joiner, Committee Member
_____	Dr. Jana Kosecka, Committee Member
_____	Dr. Sanjeev Setia, Chairman, Department of Computer Science
_____	Dr. Kenneth Ball, Dean Volgenau School of Engineering
Date: _____	Summer 2016 George Mason University Fairfax, VA

Contactless Interfacing with DICOM Using Leap Motion

A thesis submitted in partial fulfillment of the requirements for the degree of  
Master of Science at George Mason University

By

Brian Notarianni  
Bachelor of Science  
George Mason University, 2014

Director: Dr. Zoran Duric, Associate Professor  
Department of Computer Science

Summer 2016  
George Mason University  
Fairfax, VA

Copyright © 2016 by Brian Notarianni  
All Rights Reserved

## Dedication

I dedicate this thesis to my mom and dad for teaching me the value of education and sending me to GMU.

## Acknowledgments

I would like to thank my adviser and mentor Dr. Zoran Duric, for advising me to work on this project with Dr. Mai. He guided me with his expertises and always insisted I keep it simple.

I would also like to thank Dr. Jeffrey Mai MD who proposed this problem and provided a professional insight and advice, and guided the design of the interface.

I would also like to thank Dr. Timothy Sauer, who advised me as a undergrad and taught me how to use SVD for dimensionality reduction.

I would also like to thank Kenneth Gould for his help with testing my interface and editing this thesis.

# Table of Contents

	Page
List of Tables . . . . .	vi
List of Figures . . . . .	vii
Abstract . . . . .	viii
1 Introduction . . . . .	1
2 Related Work . . . . .	3
3 Background . . . . .	5
3.1 Leap Motion . . . . .	5
3.2 DICOM Viewer . . . . .	7
3.3 Singular Value Decomposition . . . . .	9
4 Technical Approach . . . . .	12
4.1 DICOM interfacing . . . . .	12
4.2 Gesture Recognition . . . . .	14
4.2.1 Generating the <i>US</i> basis . . . . .	17
4.2.2 Runtime Recognition . . . . .	17
5 Experiments and Discussion . . . . .	21
5.1 Gesture Recognition Rates . . . . .	21
5.2 Usability with Gloves . . . . .	22
6 Conclusion . . . . .	24
Bibliography . . . . .	25

## List of Tables

Table	Page
3.1 A table of the data the Leap Motion provides via the SDK. . . . .	5
3.2 The icon and modes provided by RadiAnt while in 2D Mode. . . . .	8
3.3 The icon and modes provided by RadiAnt while in 3D mode. . . . .	9
4.1 An example basis retrieved from a $X$ . Each mode represents a scaled axis that goes through the Center for each bone. . . . .	18
4.2 Sample set of gestures and their approximation projection coefficients of the first three modes. . . . .	20
5.1 The recognition rates with 7 coefficients for two users. . . . .	21

## List of Figures

Figure	Page
1.1 A 3D view of a CT scan before and after adjusting the Contrast/Brightness settings. . . . .	1
3.1 Leap Motion hardware and palm. . . . .	6
3.2 Leap Motion physical hand vs hand structure. . . . .	6
3.3 RadiAnt DICOM Viewer with MANIX sample CT scan open. The 2D view (left) and a reconstruction of the same scan in the 3D viewer (right). . . . .	7
3.4 The toolbar from RadiAnt with 5 sections: Case Selection, Display Settings, Interaction Modes, Reconstruction and Views (including 3d Volume Rendering), Help . . . . .	8
3.5 The 3D toolbar from RadiAnt with 3 sections: Case Selection, Interaction Modes, Measure . . . . .	9
4.1 During runtime there is a simple distinction between the tasks each hand can perform. <i>Menu Hand</i> manages the settings, and <i>Control Hand</i> interacts with the DICOM based on those settings. . . . .	13
4.2 The menu before selection (left) and after it gets locked (right). . . . .	14
4.3 This diagram shows both the offline (training) and online (usage) of my approach. . . . .	15
4.4 At every time $t$ , I retrieve a hand structure from the Leap and stack the bones position on top of each other. These get statistically normalized after I am done collecting samples. . . . .	16
4.5 Scatter plot of coefficients $\mathbf{v}_{i,1:3}$ of the candidate gestures obtained by SVD. Red(bottom) is gesture 1; Lime(right) is gesture 2; Purple(Left) is gesture 3; Cyan(top) is gesture 1. . . . .	19
5.1 The error rates of gesture classification. . . . .	22
5.2 Screen capture of the Unity Tool, that uses both images from the leap, that I used for Timing Recognition. . . . .	23
5.3 Box-and-Whisker plots of hand detection times for different hand conditions. . . . .	23



# Abstract

## CONTACTLESS INTERFACING WITH DICOM USING LEAP MOTION

Brian Notarianni

George Mason University, 2016

Thesis Director: Dr. Zoran Duric

The goal of my work is to design and implement a gesture-based interface with a DICOM compliant image viewer. DICOM is a widely accepted standard for storing and viewing medical imagery. DICOM viewers require use of a mouse to interact with menus and to manipulate imagery. For a surgeon to use a DICOM viewer in an operating room, a method of covering the mouse or assistance from another person is required. This is usually not practical.

In this thesis, I have developed a gesture-based interface which maps mouse actions to hand gestures. The interface is intuitive and designed to add minimal cognitive load on the user. The system uses the Leap Motion to track hands. The output of the Leap Motion are “bones” of the hand, which are projected to low dimensional space using Singular Vector Decomposition to obtain a compact gesture representation. Gesture recognition is performed using  $k$  Nearest Neighbors in this low dimensional space.

I have implemented a system prototype with RadiAnt DICOM compliant viewer, Leap Motion and Unity. I ran a series of tests to evaluate my interface. First, I determined system usability with Dr. Jeffrey Mai [1]. Then I tested the gesture recognition with two users. Finally, I tested recognition times with dry and gloved hands covered with simulated blood.

## Chapter 1: Introduction

In this thesis, I will be presenting a methodology and prototype for interfacing with a DICOM viewer. DICOM (Digital Imaging and Communications in Medicine) viewers are used across the medical field for viewing important medical imagery.

This imagery includes both 2D and 3D scans such as MRI (Magnetic resonance imaging) and CT (Computerized Tomography) scans. Medical scans are very information dense, and as such they cannot be viewed completely in a simple view.



Figure 1.1: A 3D view of a CT scan before and after adjusting the Contrast/Brightness settings.

There are tools in a viewer in order to facilitate the exploration of these scans in an office. My goal is to bring these tools into the Operating Room, so that a surgeon can manipulate the scans mid-surgery. An example of this would be a CT scan that has been reconstructed in to a 3D view, it can be adjusted so that it shows just the outermost skin to just the bones as in Figure 1.1. Currently, if a surgeon wanted to change this setting in the Operating Room, they would have to “break sterile” to use the mouse for these manipulations. Surgeons have been known to use cumbersome techniques, such as placing a towel over the mouse, or instructing a nurse to manipulate the scans. My goal is to make

an interface which is more convenient than their techniques in such a manner that does not increase cognitive load of the user.

My solution is a hand gesture based interface made specifically for use with DICOM viewers. The interface is designed to be intuitive, it uses simple hand motions that are familiar to users who already have experience with DICOM viewers. Using the Leap Motion, my method constructs a vector representing the “bones” of the tracked hand. During the training phase, these vectors form a matrix  $X$  that gets decomposed by Singular Vector Decomposition to obtain  $X = USV^T$ .  $US$  acts as a basis onto which the hand vectors are projected. This projected vector is then truncated and used by  $k$  Nearest Neighbors using  $V^T$  as gesture templates.

I have implemented a prototype of this system with the RadiAnt DICOM viewer, Leap Motion, and Unity. The prototype is made specifically for the RadiAnt DICOM viewer, although the design of the interface can easily be retooled for other DICOM viewers. Unity was used for data collection and testing.

The interface was designed with help from Dr. Jeffrey Mai [1]. His experience with DICOM viewers, as well as professional preferences, helped to shape the design. The gesture recognition, was tested with two different users, while the training was done by a simple user. Initially there was concern about the performance of the Leap Motion using gloves, especially while they are bloody. I tested the gesture recognition while wearing clean, dry Nitrile gloves, as well as with simulated blood on the same gloves.

The rest of this thesis will have the following layout: First, I will provide an overview of other work in the field from the literature in Chapter 2; next, in Chapter 3, I will present general information on the technology used; the Leap Motion, DICOM and SVD. Then, in Chapter 4, I will talk about my prototype software that uses this technology. To do this, I will start, in Section 4.1, with my interface’s layout and how my prototype interacts with the DICOM viewer and then, in Section 4.2, how the user’s gestures are recognized. In Chapter 5, I will cover the testing performance of the system and its ability to handle gloves. In Chapter 6, I present the conclusion and future work.

## Chapter 2: Related Work

This problem has been around since DICOM views have been used in the OR. While hardware, such as touch screens and even self sterilizing keyboards [2] help address the problem, they do not allow for full manipulation.

In the literature there are two main approaches, simulating the mouse and direct gesture control of the DICOM. Direct gesture control needs access to an API or SDK, while the simulating the mouse does not have contextual knowledge of the DICOM.

The actions that the surgeon wants to perform are mapped to the actions of a standard mouse [1]; *left click*, *right click*, *middle click*, and movement. As such there are several approaches [3–5] that just simulate the mouse directly. Mouse simulation is not content aware, as it is usually made for general purpose interface rather than specifically for the DICOM viewers.

Mouse simulation can also be performed with the Leap Motion. In [5] GameWave, which is an off the shelf software meant to use the Leap Motion as a mouse alternative for gaming, is used. They have even used this approach during actual surgeries, showing that gloves and blood do not impede the Leap Motion’s usage.

Depth camera capabilities of the Kinect, a commercially available device from Microsoft, have been used to perform gesture recognition [4]. These gestures are mapped to the mouse, not to the software, such as *left click* and *double click*.

Researchers in [4] used the location of the hand to position the mouse, and three imaginary surfaces next to the hand. They would use pointing at the forward imaginary surface with one finger to move the mouse, and pointing rightwards with the thumb to right click. If two fingers were used to move the mouse, their program would click and hold. The solution proposed by [3] also implemented an imaginary surface, which had a direct mapping of space to screen coordinates as if you were holding an imaginary mouse with one finger.

The mouse emulation approach suffers from how the DICOM viewer’s User Interface (UI) is meant for traditional mice that were designed and refined over decades. Users are used to moving the mouse across the screen to a tiny button. Performance test on the Leap as a mouse directly was performed by [6]; that is, tracking hand motion directly to mouse position. They found that the Leap took 3 times longer to travel to a button, and had three times the error compared to a standard mouse. This makes the approach impractical for the use in the OR.

The technique in [7] focused on a custom render instead of interfacing with a DICOM viewer. Their approach included tracking gestures with the Leap Motion to select a cutting plane, and other ways to select a region of interest. They tested their approach on non-medical subjects instead of trained surgeons. Their approach would be difficult to integrate with existing DICOM viewers, which are deployed in ORs already.

While [8] added additional functionality to an Ultra Sound Scanner with assistance from the manufacturer to allow for adjustment of the common settings, such as gain, zoom, and contrast using the Leap Motion. Their approach focused on usability with and without tools in the hand, making an interface that used one gesture that could be interpreted without dropping a medical tool such as a scalpel.

My approach differs from mouse simulation and direct control. It uses gestures to choose what actions to perform, then runs macros to set up the mouse, and finally emulates the mouse in the center of the view. This means a user would not need to click on the small WIMP, I.E. Windows Icons Menu Pointers, display, but still has the familiarity of a traditional mouse. Another way my approach differs from the other ones is that I put a lot of focus on avoiding accidental actions. With approaches like [4] it is easy to accidentally adjust one setting while intending to adjust another. My approach focuses on making adjusting any two settings independent of each other.

## Chapter 3: Background

In this design, I make use of several technologies. This chapter will describe the Leap Motion, DICOM viewers, and SVD in detail. First, I will talk about how I get hand pose information from the Leap Motion. Next, I will discuss how DICOM viewers work. Finally, I will talk about Singular Value Decomposition (SVD), and what it provides.

### 3.1 Leap Motion

The Leap Motion is a hand tracking device. It uses two infrared cameras with wide view lens and three 850 nanometers LEDS for illumination. This means it shares the limitations of many computer vision methods. Work area and shape, for instance, are an inverted pyramid above the device.

The leap SDK processes these images into data structures called *frames*. A *frame* is the state of the work area at a given time. *Frames* contain any number of *hands*. These *hand* data structures keep track of one hand in the work space. This includes handedness (left/right), position, and pose. The pose is stored in a collection of 5 *fingers*. *Fingers* contain *bones*, which has the raw positional data. Also included in the hand data is a palm direction, wrist direction, and a vector basis from these two.

Table 3.1: A table of the data the Leap Motion provides via the SDK.

Parent Data Structure	Child Data Structure	Other Variables
Frame	Hand (0+)	Time stamp
Hand	Finger (5)	Is Right, Palm Position, Basis
Finger	Bone (4)	Finger type
Bone		Center, Next Joint, Previous Joint

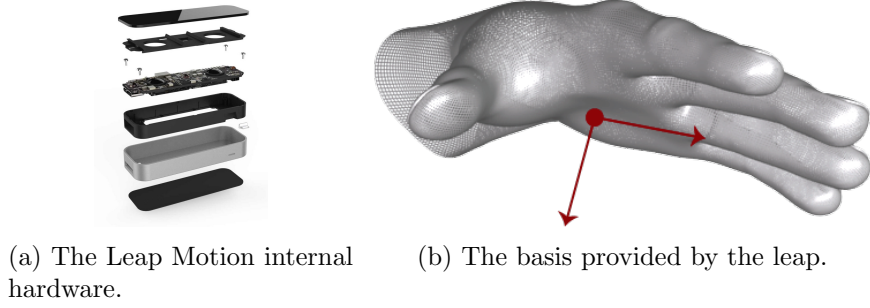


Figure 3.1: Leap Motion hardware and palm.

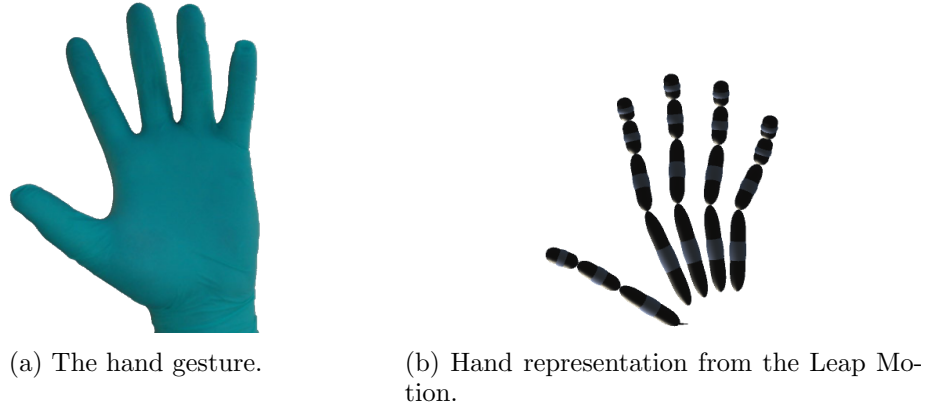


Figure 3.2: Leap Motion physical hand vs hand structure.

Before my numerical techniques can be used on the data, the data needs to be turned into a vector. I first retrieve a hand data structure from the Leap motion. This hand has 5 fingers, each of those fingers have 4 bones. This means that each hand has 20 bones with an  $[x, y, z]^T$ .

Each one of these bones has a set of joints. From this I take *next\_joint*. This is a point in 3 dimensional space of the Leap device. For gesture recognition purposes, the gestures should be independent of global position. For this, this point is converted to the reference frame of the hand. I use the palm transform provided by the Leap. It uses the center of the palm for the origin of the coordinate system. The basis is formed by the normal of the

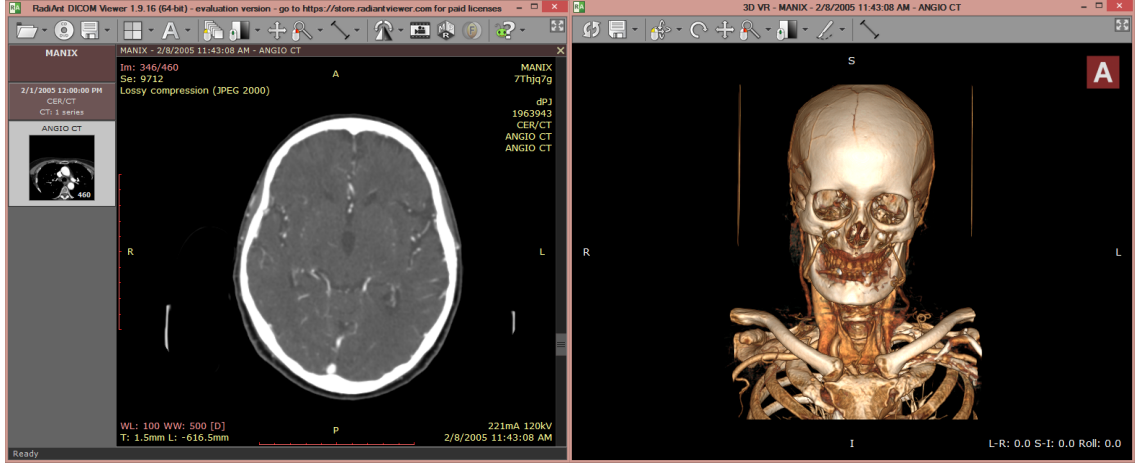


Figure 3.3: RadiAnt DICOM Viewer with MANIX sample CT scan open. The 2D view (left) and a reconstruction of the same scan in the 3D viewer (right).

palm and a vector in the direction of the wrist to the center of the palm.

After this transformation, I stack these bones upon each other into one  $m \times 1$  vector, where  $m = 60$ .

### 3.2 DICOM Viewer

Digital Imaging and Communications in Medicine (DICOM) is a standard defined by NEMA and ISO. When DICOM is referred to in this thesis, I am usually referring to a DICOM viewer such as Figure 3.3. When I say DICOM scan, I am referring to the scan open in the DICOM viewer; this can be a Magnetic Resonance Imaging scan (MRI), Functional MRI scan (fMRI), Computerized Axial Tomography scan (CAT), etc.

There are a large variety of DICOM viewers out there, ranging from simple ones for viewing personal records, to professional ones for the OR. Most DICOM software maps the important functionality to the mouse.

For my prototype, I primarily used RadiAnt. There are three different ways for a user to perform the same core set of instructions. The core set of actions is changing the zoom, contrast/brightness, panning, and the layer selection. The first way is to use each mouse





Figure 3.4: The toolbar from RadiAnt with 5 sections: Case Selection, Display Settings, Interaction Modes, Reconstruction and Views (including 3d Volume Rendering), Help

button as a different mode; holding down right click and moving the mouse will zoom the image in and out, which is a different action than holding middle mouse button and moving to adjust contrast/brightness.

The second method is to use the icons at the top of the screen. These will set the left mouse button's functionality based on the button. By default, holding the left mouse button and moving it up or down will change the layer, but after clicking pan image the left mouse button will use that functionality. The third method, similarly, sets the functionality of the left mouse click. You can activate the icons at the top of the screen with keyboard shortcuts, such as pressing Z to set it to zoom. More such controls can be seen in Table 3.2.

Table 3.2: The icon and modes provided by RadiAnt while in 2D Mode.

Modes	Icon	Default Mouse Button	Key Combo
Layer		Left Mouse Button	B
Brightness/Contrast		Middle Click	W
Pan		Shift + Left Mouse Button	M
Zoom		Right Mouse Button	Z
Measure (Length)		N/A	L

With RadiAnt, the 3D mode is shown in a different window. To access this window you use the 3D View button from (Figure 3.4). This new window has a new toolbar Figure 3.5 and set of commands (Table 3.3).



Figure 3.5: The 3D toolbar from RadiAnt with 3 sections: Case Selection, Interaction Modes, Measure

Modes	Icon	Default Mouse Button	Key Combo
3D Rotation		3D Rotation	R
Roll		N/A	T
Pan		Shift + Left Mouse Button	M
Zoom		Right Mouse Button	Z
Brightness/Contrast		Middle Click	W
Scalpel		Alt + Left Mouse Button	S

Table 3.3: The icon and modes provided by RadiAnt while in 3D mode.

Not all DICOM viewers support all of these methods. As such, my prototype allows for a configuration file that defines what method to use as well as the key combos and icon locations. This allows my prototype to be used for multiple different versions rather than being specific to one particular DICOM viewer.

In a sterile environment, the user is not allowed to use a mouse, as they cannot be satisfactorily sterilized. This means the above actions cannot be performed without breaking “scrub”. In the course of looking in to this problem, I learned of multiple makeshift ways around breaking sterile. In some cases, doctors place a towel over the mouse, and others look over an assistant’s shoulder and dictate what to do.

### 3.3 Singular Value Decomposition

Singular Value Decomposition(SVD) factorizes a  $m \times n$  matrix  $A = USV^T$ . The matrix  $U$  is a  $m \times m$  matrix that forms an orthonormal basis.  $S$  is a  $m \times n$  diagonal matrix that

scales this basis,  $V$  is a  $n \times n$  matrix that can be used to recreate the original  $A$ .

I use the following notation,  $a_{i,j}$  is the element at  $i$ th row and  $j$ th column of matrix  $A \in \mathbb{R}^{m \times n}$ . I use  $\mathbf{a}_{:,j} \in \mathbb{R}^{m \times 1}$  and  $\mathbf{a}_{i,:} \in \mathbb{R}^{1 \times n}$  to represent the  $j$ th column and  $i$ th row of  $A$  as vectors.

SVD has many useful properties. When  $A$  is a set of scattered points, in a normally distributed ellipsoid around the origin,  $US$  will contain the semi-axes of the ellipsoid containing the points. Since the elements of  $S$  are in non-increasing order, the major semi-axis is first, followed by the next semi-axis.

SVD can be used for dimensionality reduction. Let  $[\mathbf{u}_{:,1} \mathbf{u}_{:,2} \dots \mathbf{u}_{:,m}] = U$  and  $[\mathbf{v}_{:,1} \mathbf{v}_{:,2} \dots \mathbf{v}_{:,n}] = V$  and  $s_i$  be the  $i$ th entry on the diagonal of  $S$ . Since the  $\mathbf{u}_{:,i}$  are ordered in non-increasing order of importance, later terms of  $U$  can be dropped.

$$A \approx A_p = \sum_{i=1}^p s_i \mathbf{u}_{:,i} \mathbf{v}_{:,i}^T = U_{:,1:p} \cdot S_{1:p,1:p} \cdot V_{:,1:p}^T \quad \text{where } p < n \quad (3.1)$$

It is easy to see that, even with  $p = n$ , that not all of  $V$  is used.  $V$  is complete orthonormal basis. Without losing accuracy during reconstruction, one can treat  $V^T$  as if it is  $m \times n$  matrix and  $S$  as a square matrix. Each one of the columns of  $V$  are a point in the  $US$  basis for which a the corresponding column of  $A$  is the same point in the basis given by the Leap. To retrieve the approximation for  $\mathbf{a}'_{:,j}$  we multiply both sides by  $\mathbf{e}_j \in \mathbb{R}^m$ , that is the vector that consists of all zeros except for the  $j$  th which is 1.

$$A \mathbf{e}_j \approx U_{:,1:p} \cdot S_{1:p,1:p} \cdot V_{:,1:p}^T \cdot \mathbf{e}_j \quad (3.2)$$

$$\mathbf{a}_{:,j} \approx U_{:,1:p} \cdot S_{1:p,1:p} \cdot \mathbf{v}_{j,1:p}^T \quad (3.3)$$

$$S_{1:p,1:p}^{-1} U_{:,1:p}^{-1} \mathbf{a}_{:,j} \approx S_{1:p,1:p}^{-1} \cdot U_{:,1:p}^{-1} U_{:,1:p} \cdot S_{1:p,1:p} \cdot \mathbf{v}_{j,1:p}^T \quad (3.4)$$

$$S_{1:p,1:p}^{-1} U_{:,1:p}^{-1} \mathbf{a}_{:,j} \approx \mathbf{v}_{j,1:p}^T \quad (3.5)$$

However,  $U$  is unary, so  $U^{-1} = U^T$  and  $S$  is diagonal, so just take the multiplicative inverse  $S^{-1} = S^*$  where  $s_{i,j}^* = 1/s_{i,j}$

$$\mathbf{v}_{j,:}^T \approx \mathbf{v}_{j,1:p}^T = S_{1:p,1:p}^* U_{:,1:p}^T \mathbf{a}_{:,j} \quad (3.6)$$

While this is for members of  $A$ , it is clear how  $\mathbf{v}_{j,:}^T$  can be treated as a function of  $\mathbf{a}_{:,j}$ . If the  $\mathbf{a}_{:,j}$  is replaced with any data point inset its corresponding  $\mathbf{v}_{j,:}$  is returned. Alternatively vectors not included in  $A$  can be used to calculate what their  $\mathbf{v}_{j,:}$  would be. I will refer to this as projecting upon the basis of  $US$ . That is, if the dot product of  $\mathbf{a}_{j,:}$  is taken with the columns of  $US$  the result is  $\mathbf{v}_{j,:}$ .

SVD will find patterns in the columns of  $A$ . So the columns of  $A$  need to be built such that the patterns can be detected. When dealing with a more complicated data structure, the data must be converted into a collection of vectors. I will show how this works with hand in Figure . The basis  $U$  is shown in Table 3.1.

## Chapter 4: Technical Approach

The first part of my approach is a flow and communication protocol which facilitates interaction with DICOM in such a manner that it is intuitive and reliable. This protocol receives gestures from the user, and uses them to navigate the menu and sends commands to the DICOM viewer. In the second part, I recognize gestures using SVD and  $k$  Nearest Neighbors.

### 4.1 DICOM interfacing

The two aspects of DICOM that need to be controllable are:

**Command Selection:** I make a menu in which a user can select a mode from Table 3.2.

This is performed by moving the menu hand to the desired option. This selects the mode.

**Image Manipulation:** I use the other hand to perform the action the mouse would take.

The user pinches to emulate holding down the mouse button and moves their hand to move the cursor.

Since only one aspect of the view should be manipulated at a time, there are two more defined actions: command hand enter and menu hand enter. The command hand sends mouse input to the DICOM viewer; this is the hand that will pan the view, adjust the contrast or zoom. The other hand, the menu hand, is the focus of my two stage design. It controls what the settings for the command hand are. The menu hand is the only way to change from one mode to another; that is, if the user is adjusting contrast with the command hand, he needs to use the menu hand to change the command hand's action to panning the view.

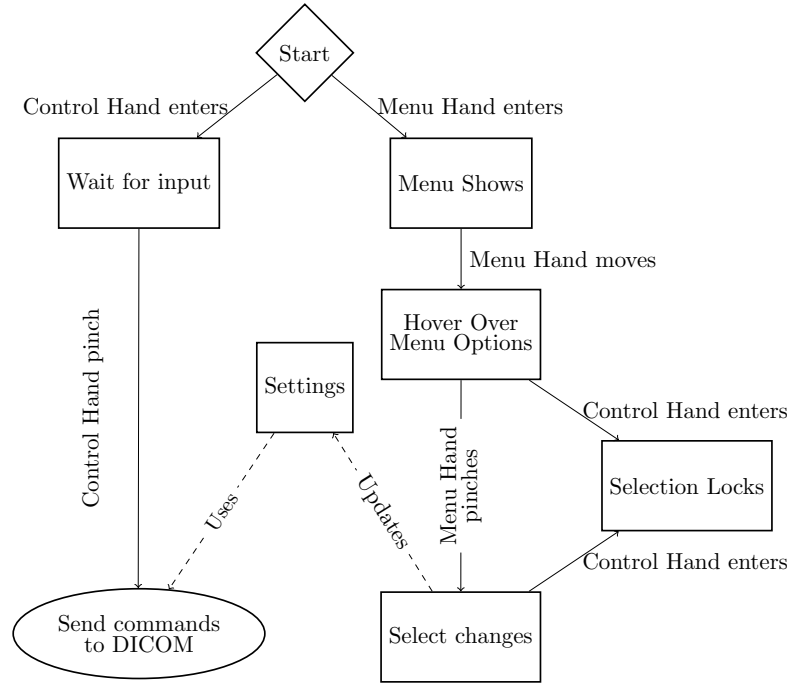


Figure 4.1: During runtime there is a simple distinction between the tasks each hand can perform. *Menu Hand* manages the settings, and *Control Hand* interacts with the DICOM based on those settings.

Upon the menu hand entering the Leap’s area, it shows a small menu that has the modes that can be used to manipulate the images. These modes correspond to the functionality of the DICOM viewer, such as brightness, contrast, pan, and rotate. When the user moves their hand, they hover over the mode and then pinch to lock in.

As an example, the prototype menu starts hidden. When the user enters with the menu hand, a menu pops up, Figure 4.2, that contains Table 3.2’s items. He then can select *contrast/brightness* if he wants to change the contrast and brightness. He then enters his command hand. While the command hand is present, the menu hand becomes inactive, and the menu turns a color to indicate its mode Figure 4.2b. He can then adjust the view with the command hand to his liking.

A mode defines two things, what motions to look for in the hand and what mouse button to use, the latter comes from Table 3.2. For example, the pan feature in the DICOM viewer

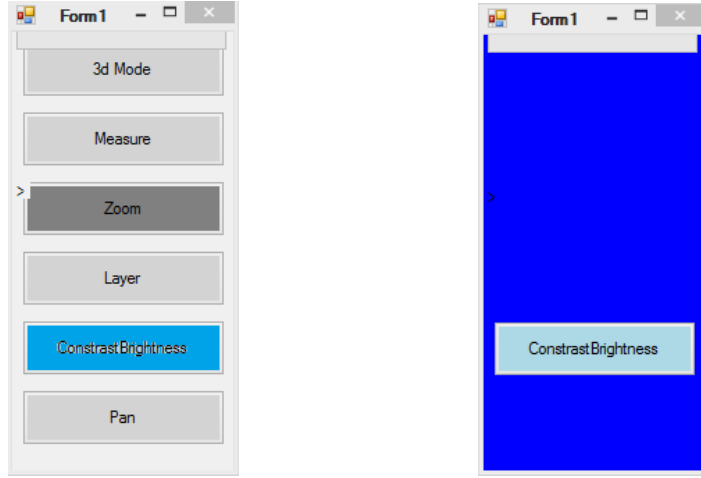


Figure 4.2: The menu before selection (left) and after it gets locked (right).

uses the left mouse click as well as mouse movement. The pan mode in my software will send xy movement and left mouse button down when the command hand pinches. When the user is no longer pinching, the mouse is moved to inside the image, so they do not click away.

## 4.2 Gesture Recognition

My method for recognition uses SVD to project hands into a low dimensional space in which I use  $k$  Nearest Neighbor to recognize gestures. The full approach is shown schematically in Figure 4.3. In the diagram there are 4 parts that need to be described.

For smooth location vector, I use simple geometric smoothing on the *palm position* given by the Leap. That is

$$\mathbf{x}'_t = \alpha \cdot \mathbf{x}'_{t-1} + (1 - \alpha) \cdot \mathbf{x}_t \quad (4.1)$$

where  $\mathbf{x}_t$  is position at time  $t$  and  $\mathbf{x}'_t$  smoothed position at time  $t$ , and  $\alpha \in [0, 1]$ .

The next step is retrieving the hand vector from the Leap done in *Bones Relative location converted into a Vector*. I first compute the center of all the 20 bones. These bones are

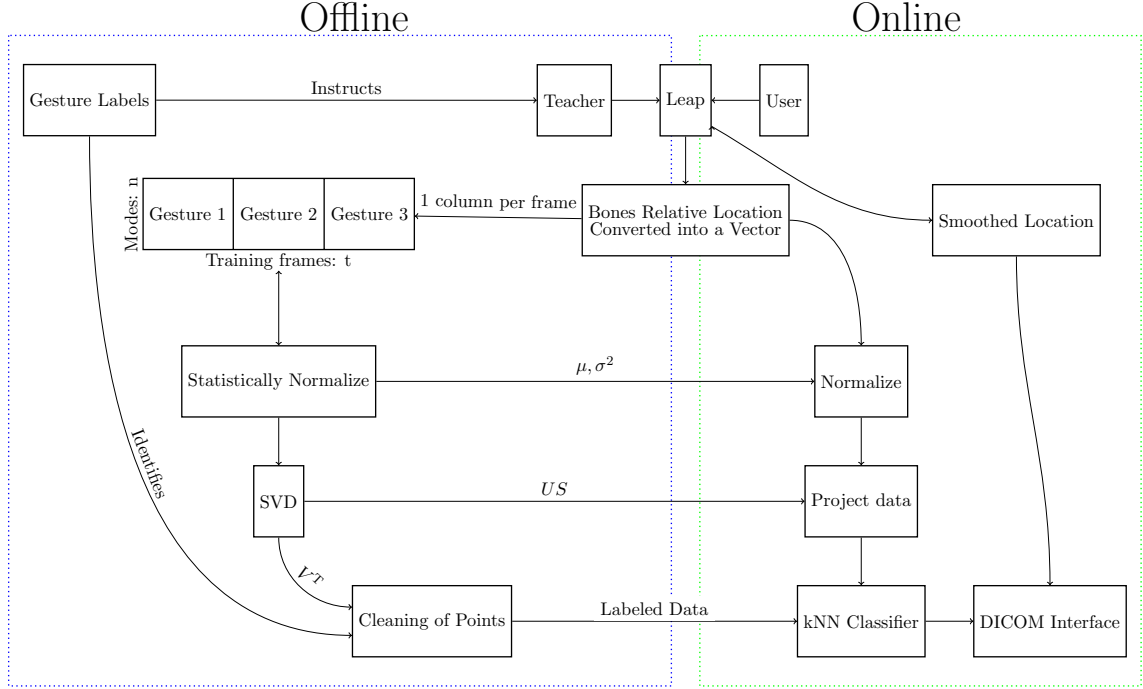


Figure 4.3: This diagram shows both the offline (training) and online (usage) of my approach.

given relative to the Leap Motion. I use them in the space relative to the palm transform. The Leap Motion provides a *palm position* (a position vector) and *palm basis* (a set of three vectors) to facilitate this. I stack these bones on top of each other to make one  $60 \times 1$  vector named  $\mathbf{x}'_{:,j}$ . This vector is shown in Figure 4.4.

Before I construct the input to SVD,  $X$ , I normalize the entries. I calculate  $\mu \in \mathbb{R}^{m \times 1}$  and  $\sigma^2 \in \mathbb{R}^{m \times 1}$  by using the standard formulas

$$\mu_i = \frac{1}{n} \sum_{j=1}^n x'_{i,j} \quad i = 1 : m \quad (4.2)$$

$$\sigma_i^2 = \max \left\{ \frac{1}{n-1} \sum_{j=1}^n (x'_{i,j} - \mu_i)^2, 1 \right\}, \quad i = 1 : m \quad (4.3)$$



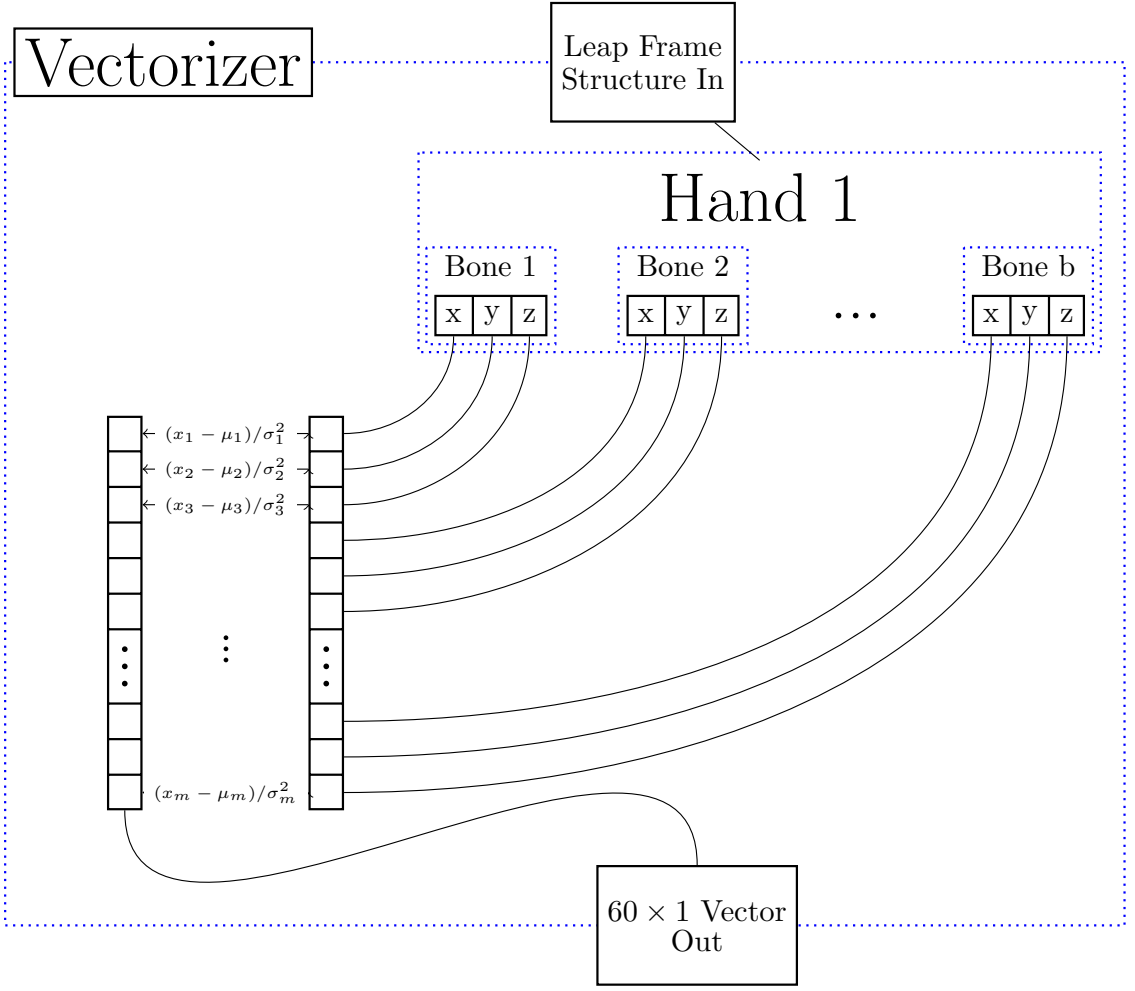


Figure 4.4: At every time  $t$ , I retrieve a hand structure from the Leap and stack the bones position on top of each other. These get statistically normalized after I am done collecting samples.

$$x_{i,j} = \frac{x'_{i,j} - \mu_i}{\sigma_i^2} \quad i = 1, \dots, m \quad (4.4)$$

If the max is not used when defining  $\sigma^2$ , a machine precision error for the bones that are very close to the origin of the hand and have very small values of  $\sigma^2$ .

### 4.2.1 Generating the $US$ basis

I need to generate the basis  $US$  used to find the  $v_{:,t}$  from the real time data. I used SVD as described in Section 3.3. I construct the matrix  $X$  which has  $\mathbf{x}_{:,j}$  as columns. The  $\mathbf{x}_{:,j}$  are the gestures representatives. During this training phase, I collect samples for  $g$  gestures. I collect the first  $k$  as the potential examples for the first gesture, in my prototype  $k = 1000$ . Then I collect  $k$  examples of the second gesture. This continues until  $n = gk$  meaning that  $X$  is  $m \times n$ .

Next, I compute SVD on this matrix to obtain  $X = USV^T$ . Here the columns of  $U \in \mathbb{R}^{m \times m}$  represent the modes of  $X$ .  $S \in \mathbb{R}^{m \times n}$  contains the scaling factors  $s_{i,i}$ , where  $i \in 1, \dots, m$ , along the main diagonal and zeros everywhere else. I form a scaled orthogonal basis using the columns of  $US$ . Finally,  $V^T \in \mathbb{R}^{n \times n}$  is another unary matrix. I use the first  $p < m$  rows of  $V$  (or columns of  $V^T$ ). In this thesis, I have used  $p = 3$  for visualizing the space and  $p = 7$  for best recognition rates.








I visualize the result in Table 4.1. The first row of this table shows a representation of  $\mu$ . Since  $\mu$  gets added to every hand when reconstructing them, it will act as an origin through which modes pass when they change from positive to negative. The first 3 hand poses from  $US$  are shown in the rest of the table.

While Table 4.1 relates to  $\mu$  and  $US$ ,  $V$  can also be viewed as scatter points. This results in Figure 4.5, which shows gesture templates from 4 gestures. Each row of  $V_{:,1:p}$  represents a  $p$ -dimensional point corresponding to an approximation of a column of  $X$  using the mode in  $US$ .  $v_{i,j}$  represents the projection of  $\mathbf{x}_{:,i}$  on  $U_{:,j}s_j, j$ . The appearances of 4 gestures are shown in Table 4.2.

### 4.2.2 Runtime Recognition

Using the mean  $\mu$ , standard deviation  $\sigma^2$ , basis  $US$ , and gesture prototype hands, gestures can now be identified. It should be noted that the  $\mu$  and  $\sigma^2$  are set from the training phase and do not take the new data into account. To keep this clear I will refer to new frames as

Table 4.1: An example basis retrieved from a  $X$ . Each mode represents a scaled axis that goes through the Center for each bone.

Mode	Negative	Positive	Description
$\mu$			Zero Position means of bones
First			General Tightness
Second			Thumb and Pointer Move in and out Together
Third			Thumb and Pointer Move in and out Alternating

$\mathbf{y}_{:,t}$  and still have:

$$\mathbf{y}_{j,t} = \frac{\mathbf{y}'_{j,t} - \mu_j}{\sigma_j^2} \quad j = 1, \dots, m \quad (4.5)$$

Next, I need to project this vector into the basis defined by  $U$  and scale it by  $s_{i,i}$ . Let

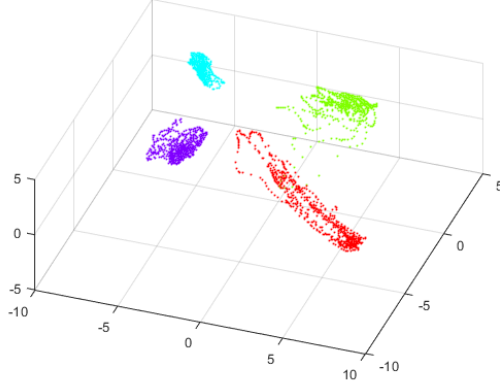










Figure 4.5: Scatter plot of coefficients  $\mathbf{v}_{i,1:3}$  of the candidate gestures obtained by SVD. Red(bottom) is gesture 1; Lime(right) is gesture 2; Purple(Left) is gesture 3; Cyan(top) is gesture 1.

$\mathbf{z}_{:,t} \in \mathbb{R}^{p \times 1}$  be the projection of  $\mathbf{y}_{:,t}$  onto this basis, and  $\mathbf{z}_{j,t}$  be the  $j$ th entry of the vector  $\mathbf{z}_{:,t}$ . This will correspond to the weight of  $u_{:,j}$  needed to recreate it.

$$\mathbf{z}_{j,t} = \frac{\mathbf{u}_{:,j}^T \mathbf{y}_{:,t}}{s_{j,j}} \quad (4.6)$$

Now that I have the point in basis of  $U$  and scaled by  $S$  I can run  $k$ NN. I selected the closest  $k$  points using euclidean distance from a labeled examples. Then I just labeled the gesture with whatever had the most matches.

Table 4.2: Sample set of gestures and their approximation projection coefficients of the first three modes.

Gesture	Example	Bone View	1	2	3
1			5.14	-5.16	0.80
2			2.60	2.90	1.12
3			-5.52	1.89	3.55
4			-5.99	0.11	-1.98

## Chapter 5: Experiments and Discussion

I have tested in both the gesture recognition and the interface usability. To test usability, I have talked to Dr. Jeffrey Mai [1]. Finally, I test the interface with bare, gloved and wet gloved hands.

### 5.1 Gesture Recognition Rates

I tested gesture recognition rates of users. User 1, myself, trained the system providing 1000 samples of 4 gestures of Table 4.2 resulting in 4000 samples. These samples were used in SVD to obtain a  $US$  basis and labeled  $V$ . I selected 1/10th of the  $V$  to use as gesture templates.

One month later, user 1 and user 2 (a different user) gathered another 4000 gestures using the same procedure each. These points used the original  $US$  basis to generate the projection  $V$ .

I then tested the accuracy of this data at 4 class classification and 1 vs all classification. I tested classification with different number of gestures modes. The best results were obtained with  $p = 7$ .

Table 5.1: The recognition rates with 7 coefficients for two users.

Subject	4 class	1 vs all
User 1	87.44%	98.5%
User 2	94.27%	95.45%

Figure 5.1 shows the results obtained with different values of  $p = 1, \dots, 20$  for both users.

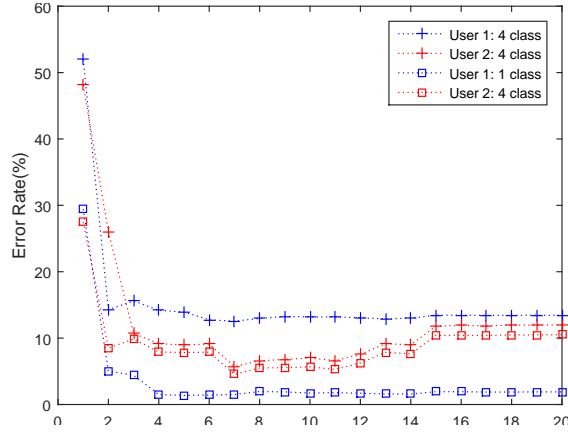


Figure 5.1: The error rates of gesture classification.

## 5.2 Usability with Gloves

For usability I relied on anecdotal evidence. This software is aimed solely at professionals in the OR. This means that it does not need to be easy to learn. My approach will take advantage of the muscle memory they have built up from years in the field.

One of the big concerns of usability was a physical one; can the Leap work in the OR. The specific problems are gloves, the sterile tarp, and blood. The literature does contain an example of Leap Motion being used in the OR during autopsies, but I tested the use myself.

I tested the interface with wet and dry gloves. I set up a testing tool that times how long it takes for a hand to be recognized by the Leap when a hand enters the view. I use the Leap Motion IR image and the detected hand count for this.

The tool I made is shown in Figure 5.2. It uses half an undistorted image from each lens to make it more reliable. I check the middle row of the image for brightness to start the timer. When the Leap detects a hand I stop the timer. I repeat the experiment 30 times for each set-up. I started with normal, gloved and glove with “blood” hands. To simulate blood, I used chocolate syrup.

I entered my hand from out of view right side, and moved my hand in the center of the



Figure 5.2: Screen capture of the Unity Tool, that uses both images from the leap, that I used for Timing Recognition.

view until detection. I then moved out of the Leap's view to the left. That way the Leap could not use previous data to speed up recognition time.

The result of the tests show that the glove does not make a difference for recognition times. The simulated blood made a small difference.

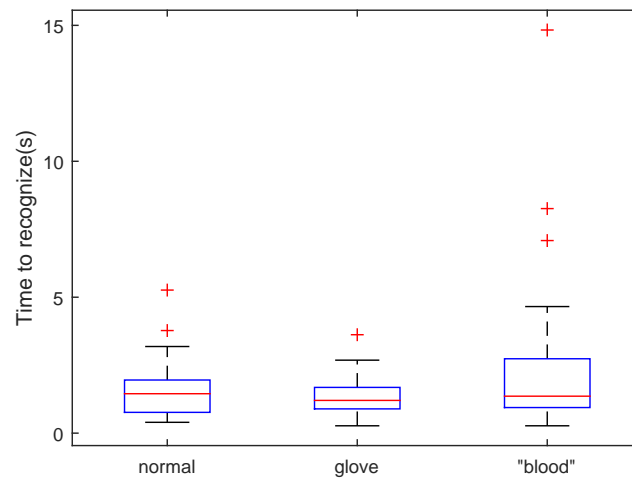


Figure 5.3: Box-and-Whisker plots of hand detection times for different hand conditions.



## Chapter 6: Conclusion

I have designed and implemented a contactless interface for the DICOM compliant viewers, such as RadiAnt, using the Leap Motion. This design uses the surgeon's existing knowledge to be as intuitive as possible and add minimal cognitive load.

The work comes in two parts, the interface and gesture recognition. This interface uses a two hands work flow to minimize user error by having the user select an option from a menu and act upon it separately. The systems use the Leap Motion to track hands. The Leap Motion's native representation gets transformed and projected into a lower dimensional space through the use of SVD. The gestures are recognized using  $k$ NN in this new space.

The system was tested in three ways. The usability was tested by Dr. Jeffrey Mai. The gesture recognition rates were tested with a second user on the original data. Finally the ability for the system to handle gloves was tested through recognition times.

The next step in this work would be to test it in a operating room and ensure that it meets the needs of medical professionals.

## Bibliography

## Bibliography

- [1] J. Mai, personal communication, 2016.
- [2] “Fda clears vioguard self-sanitizing keyboard,” <http://www.infectioncontrolday.com/news/2012/01/fda-clears-vioguard-selfsanitizing-keyboard.aspx>, accessed: 2016-07-10.
- [3] A. Manolova, “System for touchless interaction with medical images in surgery using leap motion.”
- [4] A. Tuntakurn, S. S. Thongvigitmanee, V. Sa-Ing, S. S. Makhanov, and S. Hasegawa, “Natural interaction on 3d medical image viewer software,” in *Biomedical Engineering International Conference (BMEiCON), 2012*, Dec 2012, pp. 1–5.
- [5] M. Nicola Bizzotto, M. Alessandro Costanzo, and M. Leonardo Bizzotto, “Leap motion gesture control with osirix in the operating room to control imaging: first experiences during live surgery,” *Surgical innovation*, vol. 21, no. 6, pp. 655–656, 2014.
- [6] M. C. B. Seixas, J. C. S. Cardoso, and M. T. G. Dias, “The leap motion movement for 2d pointing tasks: Characterisation and comparison to other devices,” in *2015 International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS)*, Feb 2015, pp. 15–24.
- [7] J. Shen, Y. Luo, X. Wang, Z. Wu, and M. Zhou, “Gpu-based realtime hand gesture interaction and rendering for volume datasets using leap motion,” in *Cyberworlds (CW), 2014 International Conference on*, Oct 2014, pp. 85–92.
- [8] N. Rossol, I. Cheng, R. Shen, and A. Basu, “Touchfree medical interfaces,” in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2014, pp. 6597–6600.

## Biography

Brian M. Notarianni graduated from Franklen W. Cox High School, Virginia Beach, Virginia, in 1983. He received his Bachelor of Science from George Mason University in 2014.