Two Hands are Better Than None: Integrating Hand-Tracking into the Lumis inSight Platform

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Ryan Michael Torbic

Spring, 2021.

Technical Project Team Members

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Raymond Pettit, Department of Computer Science

Mark Floryan, Department of Computer Science

Two Hands are Better Than None

Integrating Hand-Tracking into the Lumis inSight Platform

Ryan Torbic
Department of Computer Science
University of Virginia
Charlottesville, Virginia, USA

ABSTRACT

Over the past fifteen months, I have been working with Lumis Corporation to provide medical simulation software to nursing and medical students. Access to quality lab resources is a difficult task; most medical simulations are expensive and/or require teacher supervision. Oftentimes, specialized equipment such as mannequins embedded with smart screens are necessary. Lumis's hardware bypasses these issues by utilizing a projector positioned above an ordinary mannequin -- such as one that could be found in any hospital -- and a simple Nintendo Wiimote, which registers light from a pen equipped with an infrared light to create an artificial "smart screen" without the needed for embedded technology.

However, this method can be improved upon. A Wiimote has limited field of view, and can only register infrared light, which poses problems when exposed to excessive sunlight and hinders future development. It is the goal of this project to replace the Wiimote with an Intel RealSense Depth Camera, by integrating it into Unity using skills and techniques developed in CS 4730. This will have numerous advantages; the RealSense Depth Camera has a wider field and depth of view, better precision, and is not limited by the need for IR light. This opens the door for new features such as using the camera to recognize if a student is interacting correctly with the mannequin without the constraint of a pen.

1 Introduction

The Lumis inSight Platform has a simple setup – a mannequin is placed on a table, with a projector on a stand facing downwards towards the mannequin. A Nintendo Wiimote is attached to the projector facing in a parallel direction (towards the mannequin), and a tablet PC is connected to the projector and running multiple displays (display one is the projector; display two is the PC).

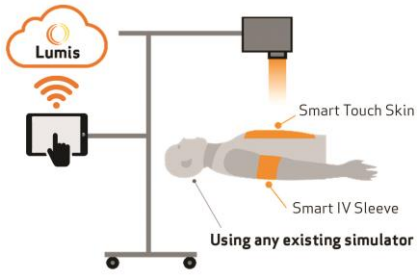

Figure 1: Basic setup of the Lumis inSight Platform

With this setup, users can interact with the tablet PC – which is mainly used for showing menu items, changing projector settings, and displaying instructions and information – using the touch screen, and can interact with the mannequin (the projected surface of display one) using a Bluetooth pen equipped with an infrared light. The Wiimote tracks the IR light and moves the mouse to the corresponding surface on the mannequin, and the Bluetooth registers mouse clicks.

When I first began working with Lumis in November 2019, the IR light tracking by the Wiimote was performed entirely by a Wiimote Whiteboard application developed by Johnny Chung Lee in 2007[1]. This was functional but not ideal, because the Wiimote Whiteboard software was independent of the Lumis software, forcing users to install two applications. To operate, they would first have to open the Wiimote Whiteboard, calibrate the remote, and only then could they open and engage with the Lumis platform. If the mannequin or projector got jostled, users would have to close the Lumis application, recalibrate their remote with the Wiimote Whiteboard, then open the Lumis application again.

One of the my first long-term projects with Lumis was to fully integrate the Wiimote Whiteboard software with our software. This was a daunting task, requiring connecting to

the Wiimote to the tablet PC via Bluetooth automatically upon platform initiation, creating a calibration interface on Unity that stores, remembers, and makes sense of calibration data, and reading and computing Wiimote IR data to send the mouse to the correct position on the projected display. Unfortunately, the Wiimote software library that Johnny Chung Lee used with the Wiimote Whiteboard was incompatible with Unity, so I was forced to implement this project using a Wiimote Library developed by Brian Peek of Microsoft. My take on the Unity-integrated Wiimote Whiteboard functioned well, but there were two problems:

1. Brian Peek's library resulted in slightly less accurate IR readings, which limited the range of movement with Bluetooth pen.
2. Because of the nature of IR light, readings experience significant interference outdoors or in well-sunlit rooms.

With these issues in mind, I came up with the premise for this project – to develop an alternative to the IR Light – Wiimote model in a way that will allow users to still interact easily with our platform.

2 Related Work

Simulation in the medical profession is no recent phenomenon; the first commercial, mannequin-based medical simulator was available as early as 1994[2]. However, the Lumis inSight Platform offers clear advantages in that it can operate both as hardware and software. There are many companies that offer mannequins with smart software embedded into them; however, these are exorbitantly expensive, unwieldy, and difficult to transport. Other medical simulation companies do not rely on mannequins at all, and instead rely on haptic systems or hyper-realistic software to simulate medical experiences. The most advanced simulations are only available in laboratories under strict supervision of use, and access time is limited.

The Lumis inSight Platform excels in that it offers both the software and the hardware. It can be run projected onto a normal mannequin (with no embedded software needed) or onto a mere table. If the user wants, they can run it directly from their personal computer. It is available around the clock, offers personalized feedback, and since the hardware is uncomplicated, relatively inexpensive.

In terms of the aim of this project, there are not a surplus of systems that track a user's movements over a mannequin. One could embed a smartboard into the mannequin, but that verges into the territory of the far more expensive simulators mentioned above, and eliminates some of the accessibility of the platform. There is the Wiimote Whiteboard software, which while effective, has its fair share of flaws, which have already been covered. Other possible alternatives include finger tracking gloves or other wearable smart devices, but once again, those are expensive options. A user could also use a controller to manipulate the mouse, which would successfully avoid the problem of IR light interference presented with the Wiimote Whiteboard. However, it is not as simple, nor do the movements afford themselves as well to medical actions, as the hands-tracking solution proposed in this project.

3 System Design

The first major decision of this process was the camera source to replace the Wiimote. The Intel RealSense Depth Camera quickly emerged as an optimal choice. It outstripped the Wiimote in virtually every category – it has a wider field of view, functions both indoors and outdoors, can measure depth in addition to IR light, and comes equipped with several easily integrable software packages[3]. Moreover, the RealSense Depth Camera can be applied for usage in augmented or virtual reality, which was close enough to Lumis's field of expertise – medical simulations – that numerous potential capabilities could be immediately envisioned.

To become comfortably with the camera's capabilities, I downloaded the Intel RealSense SDK 2.0 from Intel's Github and imported the package into a new Unity project. A perusal of the sample scenes showed that the camera's abilities were very impressive – the RGB camera had multiple visual options; the depth camera could be manipulated to mask objects both nearer and farther than user-specified distances; and the IR camera was incredibly accurate and experienced little to no sunlight interference.

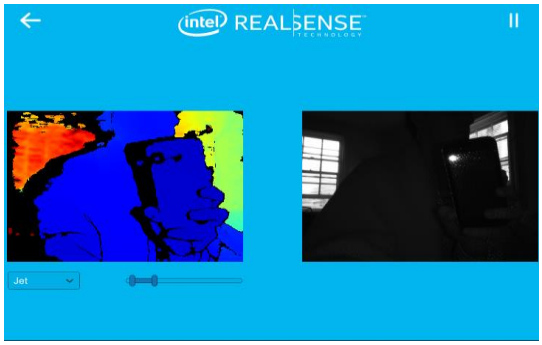

Figure 2: Using the Intel Depth (left) and IR (right) cameras. Note the IR light from my cell phone that is not present in the depth camera.

3.1 Deciding course of action

Keeping in mind the timeline for this project, I decided the best course of action would be to keep the existing structure of the platform, merely replacing the Wiimote with the RealSense camera. This may seem like a lateral move, but it would in actuality be a significant upgrade. The indoor/outdoor nature of the camera would limit the interference from sunlight, and the depth of the camera could also be set to omit unwanted light sources. Also, the wider field of view would be an added bonus.

However, after some analysis into the scripts behind this functionality, I was discouraged by the idea's vitality. Intel's provided SDK made it very easy to see what the camera is seeing, but that is done by streaming data directly from the camera to a Unity texture – it is not so easy to access and respond to those images at the code level. Or in this case, detect what is the point of strongest IR light, and move the mouse correspondingly. In that sense, the Wiimote – which is designed to detect the location of IR light emitted from the Wii Sensor Bar – is actually a better choice. However, since that was not a viable option, I planned to do an extremely thorough review of the RealSense SDK's scripts to isolate how to best track IR light. This was not an appealing course of action – it was sure to be tedious, time-consuming, and with no guarantee of success within my time frame.

3.1.1 Seeing the light – and abandoning IR entirely

Before delving into a minefield of code, I conducted some research into alternative uses of the RealSense Depth Camera and quickly came across the Nuitrack SDK, which specializes in skeletal tracking[4]. Nuitrack offered several enticing benefits:

a. Is compatible with Intel cameras.
b. Is available for Unity.
c. Comes with an perpetual free trial (with a time cap per use, which is irrelevant for the purposes of this project).
d. Offers skeletal tracking and gesture recognition.

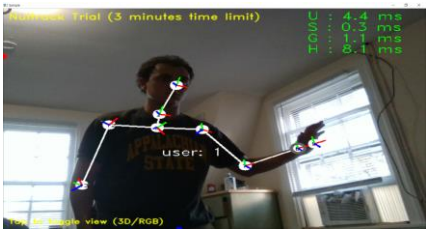

Figure 3: Skeletal tracking with Nuitrack application

I downloaded the Nuitrack application for testing, and the potential of the skeletal tracking quickly caught my eye. With the current system, a user needs a Bluetooth pen to manipulate the cursor. This was a fine option, but the goal of Lumis is to provide the most accurate simulation of medical services as possible. A nurse is not treating a patient with a Bluetooth pen, but with their hands. As such, an optimal simulation will not rely on hardware intermediaries, but simply on the user's hands. This realization led to me deciding to use the Nuitrack SDK and skeletal tracking as an alternative to relying on IR light to monitor user movement.

3.2 Nutriack: the hands-on approach

The Nuitrack Unity SDK was much more intuitive than the RealSense Unity SDK. Where RealSense merely offered example scenes that gave no explanations of the mechanisms behind the scenes, Nuitrack offered step-by-step tutorials on how to implement various aspects of their software's capabilities. Once I had imported the SDK into the Lumis Unity project, I attempted the first tutorial – skeletal tracking – and completed it without much difficulty. The skeletal tracking tutorial would automatically connect to the camera, detect the user, and display red ball objects in place of the user's skeleton. Initially, I planned on isolating just the red balls that corresponded to the user's hands, and using those to control the mouse.

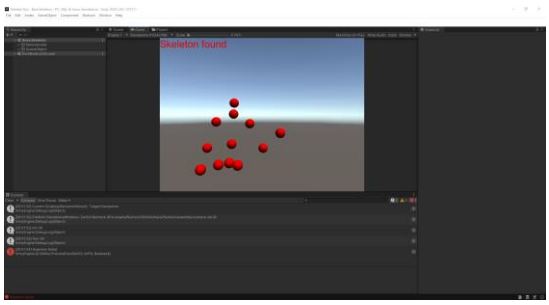

Figure 4: Skeletal tracking tutorial in Unity

However, as I performed a more exhaustive delve into Nuitrack's tutorials, I found one for hand tracking, which:

1. Automatically connected to the camera and detected the user.
2. Tracks both of the users hands and follows them in the Unity canvas-space with sprites.
3. Recognizes gestures such as a closed fist.

Thus, with a solid grasp of hand-tracking and gesture recognition, I set upon integrating Nuitrack into Lumis software.

3.2.1 Nuitrack and Lumis working hand-in-hand

Nuitrack's hand-tracking tutorial was a solid baseline, but it only moved sprites around the canvas, which was not nearly enough functionality. I needed to be able to move the actual cursor and trigger mouse clicks. Unity itself does not provide support for either of these actions; the only cursor-related operation it allows is changing cursor visibility. However, my work on the Wiimote Whiteboard exposed me to a number of Windows-specific functions that can manipulate the mouse, so, after making some mathematical adjustments to equate the canvas space to the screen space, I soon had a working function to move the mouse through hand movement alone. Likewise, I also made a closed fist correspond with a mouse click, at least initially.

3.2.2 Two hands are better than one

However, a blessing at first – the ability to track two hands simultaneously – soon turned into an interesting design question: that is, deciding how to manipulate the mouse when two hands are active. A study entitled "Left Handed Medical Professionals, in a Right Handed World: An ergonomic challenge" claimed that only 10-12% of the world's population is left-handed, and that, as unfortunate as it may seem, in the medical profession "left-handedness is considered as an inconvenience"[5].

Based on this, I made the right hand the default for the majority of mouse-related actions. If both the user's right and left hands are detected by the camera, the mouse will follow the right. The right hand also has clicking power; if a user closes their fist, the mouse will initiate a full click at that cursor location.

However, the left hand does retain some key functionality. The ability to drag and drop objects is a key component of the platform – for example, dragging and dropping defibrillation pads to the correct location on a patient's chest. The left hand retains the dragging and dropping functionality. When the user closes their fist, a mouse down function is initiated; when the user releases their fist, the mouse lifts up. Consequently, a user could also use their left hand to initiate a full mouse click, assuming they closed and opened their fist quickly enough.

However, in order to not favor right hand users and make the platform easily accessible for all, I added functionality in the main menu to swap dominant hands, which flips the operations of the right hand and left hand, or vice versa.

Having two hands operable also provided the benefit of more control. With the previous system – which only had the movement of the IR light and the clicking of a button – there were only two possible inputs. However, with two hands, in addition to each hand being able to form a fist on its own, there is also the additional input of both hands forming a fist at the same time. I associated that motion with switching between displays. On the original system, the user would only use the Wiimote to operate the cursor on the projected display over the mannequin, and they would use either a mouse or the touchscreen to operate the tablet PC. However, my solution allows easy flipping between the projected and tablet displays, which allows the user to operate oven the mannequin and jump back to manipulating menu items on the tablet without moving a step.

With the ability to move the cursor, initiate both full clicks and drag-and-drop, and switch the mouse across displays, my solution for user hand-tracking was finally implemented.

4 Results

The hand-tracking solution served as a functional alternative to the Wiimote Whiteboard. To verify, I had three test subjects run through a number of tasks to test the software's functionality. Subjects were instructed to:

1. Click through numerous menus to finally arrive at a scene designed to test defibrillation.
2. Switch modes so that they were operating the cursor on the body (projected) display.
3. Drag and drop defibrillation pads onto the correct parts of a patient's body.
4. Press the "Zap" button to test if the pads were placed correctly.
5. Repeat steps 3 and 4 as necessary.

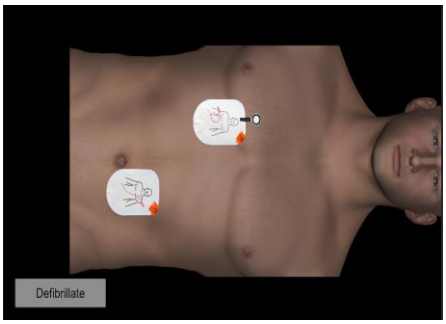

Figure 5: Users were instructed to use hand motions to drag and drop defibrillation pads onto patient.

On the whole, all participants had a positive experience with the platform. They completed their tasks without significant bugs or malfunctions. A particular strength was the drag-and-drop functionality; the test subjects were remarkably precise when moving the pads around the screen, performing much better than I originally expected. They also proved to be much more adept at navigating the menus than I thought; I anticipated errant clicks and shaky hands to throw off mouse positioning, but nothing of the sort emerged.

That is not to say that all three participants' experiences were flawless. Two of the three struggled with Step 2 – switching the cursor across displays. Although my code had taken measures to prevent this bug, oftentimes when the users closed both their hands to switch the cursor to the other display, when the cursor arrived at the other display, the system would register two closed hands and send the cursor right back to where it originated. As a result, display switching often took multiple attempts. Moreover, when both hands were being registered by the camera, the steadiness of both was reduced. As a result, I had to advise the participants to hide one hand out of the camera's view when doing high-accuracy tasks such as dragging the defibrillation pads. While these bugs are not desirable, they at least did not take away from the overall user experience, as all participants reported total satisfaction with the platform. Nevertheless, addressing these issues will certainly be a focus of future work.

5 Conclusion

I began this project with a simple goal: to try to find a reasonable alternative to the Wiimote Whiteboard method of tracking user movement. This was a necessary replacement – the Wiimote Whiteboard had limited range, lacked affordances to medical practice, and suffered interference when exposed to sunlight. As a solution, I circumvented the need for IR light or user-controlled hardware entirely, instead relying on skeletal tracking to permit the user to manipulate the platform with hand motions.

This was shown to be a functional solution; three subjects underwent a sequence of steps designed to test possible actions that a typical user of the Lumis inSight Platform would be called upon to perform. All three participants achieved the steps with minimal difficulty, experiencing some tiny bugs, but on the whole reporting complete satisfaction. Equally importantly, the hand-tracking solution removed the issues of the Wiimote Whiteboard – whether run in sunlight or indoors, user experience was the same, and the better quality of the Intel RealSense Depth Camera as compared to a Nintendo Wiimote led to a wider field of view and user recognition.

In short, this project can only be classified as a success; it expanded upon the functionality of the previous system while eliminating some of the existing drawbacks, all while creating an experience that better affords to typical medical practice.

6 Future Work

I intend – and likely will have opportunity to – expand on this project in the future. I would first like to fully integrate it with the Lumis hardware; I do not have access to the full projector, mannequin, tablet PC, and entire setup that goes with the inSight Platform. While I know my project works on multiple displays with the camera positioned on my desk, I would like to verify functionality in its final-product form.

I would also hope to include more gestures in addition to simply clutching of the fist. Medical staff perform various actions on a patient – it would be useful to recognize and react to common hand motions such as inserting a syringe, or administering the Heimlich maneuver.

Lastly, I would like to explore the capabilities of the Intel RealSense camera even further. The provided demos showed that it can be used for augmented and virtual

reality. While most times users will be working with a physical mannequin, it would be an enticing challenge to simulate virtual patients with which users can interact. In short, I am very excited for the possibilities that this project has opened, and look forward to revisiting it in the future.

ACKNOWLEDGMENTS

I would first like to thank Doug Nelson, founder and CEO of Lumis, for being a wonderful boss and mentor and providing me with many of the tools needed for this project. Also to anyone who tested out my work for the results section, especially since they gave me positive reviews. Lastly to my advisors, Prof. Pettit, Prof. Floryan, and program advisor Prof. Bloomfield, who all gave me valued feedback and, most importantly, preserved my eyes by giving me permission to use a larger font size.

REFERENCES

[1] Johnny Chung Lee. 2007. Low-Cost Multi-point Interactive Whiteboards Using the Wiimote. (2007). Retrieved April 23, 2021 from http://johnnylee.net/projects/wii/

[2] Anon. 2019. A brief history of the Center for Medical Simulation. (October 2019). Retrieved April 23, 2021 from https://harvardmedsim.org/history/

[3] Intel. 2021. Depth Camera D435. (March 2021). Retrieved April 23, 2021 from https://www.intelrealsense.com/depth-camera-d435/

[4] Nuitrack. 2021. Nuitrack Full Body Skeletal Tracking Software. (2021). Retrieved April 23, 2021 from https://nuitrack.com/

[5] Apurva Lunia. 2015. Left-handed medical professionals in a right handed world: An ergonomic challenge. (2015). Retrieved April 23, 2021 from https://www.longdom.org/proceedings/lefthanded-medical-professionals-in-a-right-handed-world-an-ergonomic-challenge-5167.html