Novel Methods for Robust Real-time Hand Gesture Interfaces

by

Nathaniel Sean Rossol

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

Real-time control of visual display systems via mid-air hand gestures offers many advantages over traditional interaction modalities. In medicine, for example, it allows a practitioner to adjust display values, e.g. contrast or zoom, on a medical visualization interface without the need to re-sterilize the interface. However, there are many practical challenges that make such interfaces non-robust including poor tracking due to frequent occlusion of fingers, interference from hand-held objects, and complex interfaces that are difficult for users to learn to use efficiently.

In this work, various techniques are explored for improving the robustness of computer interfaces that use hand gestures. This work is focused predominately on real-time markerless Computer Vision (CV) based tracking methods with an emphasis on systems with high sampling rates.

First, we explore a novel approach to increase hand pose estimation accuracy from multiple sensors at high sampling rates in real-time. This approach is achieved through an intelligent analysis of pose estimations from multiple sensors in a way that is highly scalable because raw image data is not transmitted between devices. Experimental results demonstrate that our proposed technique significantly improves the pose estimation accuracy while still maintaining the ability to capture individual hand poses at over 120 frames per second. Next, we explore techniques for improving pose estimation for the purposes of gesture recognition in situations where only a single sensor is used at high sampling rates without image data. In this situation, we demonstrate an approach where a combination of kinematic constraints and computed heuristics are used to estimate occluded keypoints to produce a partial pose estimation of a user's hand which is then used with our gestures recognition system to control a display. The results of our user study demonstrate that the proposed algorithm significantly improves the gesture recognition rate of the setup.

We then explore gesture interface designs for situations where the user may (or may not) have a large portion of their hand occluded by a hand-held tool while gesturing. We address this challenge by developing a novel interface that uses a single set of gestures designed to be equally effective for fingers and hand-held tools without the need for any markers. The effectiveness of our approach is validated through a user study on a group of people given the task of adjusting parameters on a medical image display. Finally, we examine improving the efficiency of training for our interfaces by automatically assessing key user performance metrics (such as dexterity and confidence), and adapting the interface accordingly to reduce user frustration. We achieve this through a framework that uses Bayesian networks to estimate values for abstract hidden variables in our user model, based on analysis of data recorded from the user during operation of our system.

Preface

The majority of the the content for chapters 3 and 4 has been submitted for publication as a journal article in the IEEE Transactions on Human Machine Systems. The content for chapter 5 is predominately based on an accepted peer-reviewed publication presented at the 36th International Conference of the IEEE Engineering in Medicine and Biology Society. The content for chapter 6 is predominately based on an accepted peer-reviewed publication presented at the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry. In each case, I was the primary individual involved in creating the software code, collecting the data, and writing up the results for publication. I hereby use the first-person plural throughout this thesis in acknowledgement of the input, feedback, and other contributions of my advisors and collaborators on my various works.

Publications during PhD study

- N. Rossol, I. Cheng, and A. Basu. A multi-sensor technique for hand pose estimation through novel skeletal pose analysis. *IEEE Transactions on Human Machine Systems*, 2015. (Submitted and under revision).

- N. Rossol, I. Cheng, I. Jamal, R. Shen, and A. Basu. Touchfree medical interfaces, in *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, Aug 2014, pp. 6597–6600.

- N. Rossol, I. Cheng, W. F. Bischof, and A. Basu. A framework for adaptive training and games in virtual reality rehabilitation environments. In *Proceedings of the 10th International Conference on Virtual Reality*

Continuum and Its Applications in Industry, VRCAI'11, pages 343-346, New York, NY, USA, 2011. ACM.

- N. Rossol, I. Cheng, J. Berezowski, and I. Jamal. An extensible interactive 3d visualization framework for N-dimensional datasets used in heterogeneous software display environments. In *Proceedings of the 7th International Conference on Advances in Visual Computing - Volume Part I*, ISVC'11, pages 508-517, Berlin, Heidelberg, 2011. Springer-Verlag.

- N. Rossol, I. Cheng, I. Jamal, J. Berezowski, and A. Basu. A real-time 3d visualization framework for multimedia data management, simulation, and prediction: Case study in geospatial-temporal biomedical disease surveillance networks. *Int. J. Multimed. Data Eng. Manag.*, 2(2):1-18, April 2011.

- N. Rossol, I. Cheng, and M. Mandal. A workflow based process visual analyzer (proviszer) for teaching and learning. In *Advances in Vi- sual Computing*, volume 6454 of *Lecture Notes in Computer Science,* pages 406-415. Springer Berlin Heidelberg, 2010.

Non-first Author publications during PhD study

- I. Cheng, R. Shen, R. Moreau, V. Brizzi, N. Rossol, and A. Basu. An augmented reality framework for optimization of computer assisted navigation in endovascular surgery. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE,* pages 5647-5650, Aug 2014.

- C. Leng, W. Xu, M. Li, N. Rossol, L. He, and D. Liu. Image registration based on the projection theorem of energy conservation in graphs. *In Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on,* volume 3, pages 1976-1980, July 2011.

Patents during PhD Study

- H. Wang, and N. Rossol. Balancing Accuracy and Power Consumption for Wearable Devices. (US: 20141235)

If you can't explain it simply, you don't understand it well enough.

– Albert Einstein.

Acknowledgements

First and foremost, I would like to acknowledge the work of my supervisors and thesis committee for making my thesis possible. I would also like to thank the jury members of both my candidacy and final thesis exams for taking the time and interest to learn about my work and offer valuable feedback. Furthermore, the completion of my degree was helped made possible by the support of NSERC and MITACS as funding agencies involved at various points in my degree program.

Besides my official academic advisors and mentors, I would like the acknowledge the help I received from fellow lab-mates and graduate students throughout my degree. In particular, I would like to thank Rui for his advice and in-depth knowledge on the academic writing/publishing process, Navaneeth and Nasim for their engaging conversations over coffee, Matt for giving me an outside perspective on everything, and Sheehan for being someone I could discuss my early ideas with, as well as just generally being a very good friend over the years. Thank you all.

Finally, I cannot imagine how different the past 3.5 years of my life would have been without the love and support of Miranda throughout it all. Your unfading encouragement and belief in me helped keep me going even over the long distances, late evenings, and especially during the times when the final goal seemed even farther away than ever. In a life full of uncertainties, my love for you is the the one thing that I can be absolutely sure of. You make my life complete!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this work, we study techniques for improving the efficiency and robustness of interaction with hand gesture interfaces. For the purposes of this work, hand gestures refer to hand and finger movements (such as pinching, waving, swiping, etc.) explicitly performed in mid-air without physically touching anything for the purpose of being interpreted as input into some type of computer interface. There are many devices that can be used to track human hand movements over time, including data-gloves, Infrared (IR) markers, or marked tools such as IR-pens. This work is instead primarily focused on markerless Computer Vision (CV) based techniques. CV-based techniques use no devices or markers attached to the human hand and instead track motions through either image or depth cameras alone.

In our work, as summarized in Figure 1.1, we investigate both the recognition and interface aspects of real-time high-dimensional data visualizations. On the recognition side, we examine techniques to make more accurate pose estimations and better recognition rates for gestures. By pose estimation we refer to being able to accurately predict the all the finger positions and joint angles in a human hand (i.e. full pose estimation) or a subset of these (i.e. such as only palm and/or fingertip positions in the case of partial pose estimation). Gesture recognition rate is related to the number of intentional hand

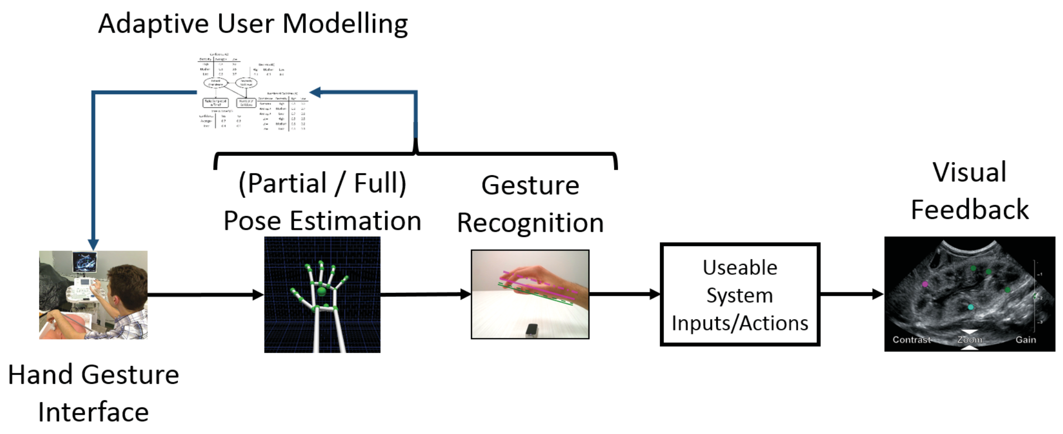


Figure 1.1: An overview of the different processes involved in our proposed approach to gesture interfaces.

gestures a user performs that we are successfully able to recognize as having occurred.

On the interface side, we examine how we can design the gesture interface and interactions to improve efficiency when users have their hands partially occluded by unmarked hand-held tools. As part of this, we look at how we can improve visualization interfaces further by creating a model of the user's performance and adapting the interface accordingly.

1.1 Motivation

Hand gestures have been investigated for many years as a more natural and intuitive form of computer input for many applications. Although much of the early work in hand gesture recognition used a variety of input devices such as data gloves or tracking markers, in recent years there has been more focus on techniques that use no markers or devices on the hand, and track the hand movements through Computer Vision (CV) techniques alone. These interfaces are the most practical for users because they can be used immediately upon arrival at the interface and require no additional equipment other than users' own hands.

CV-based hand gesture interfaces are ideal for many applications where one wishes to interact with a display without touching a physical interface. Consider the example of controlling the visual parameters of a medical display during a live procedure such as an ultrasound examination, or a dental operation. During such a procedure, practitioners can adjust displays without any concern about spreading biological contamination that may otherwise have resulted from using a touch interface or physical device. Touchless CV-based gesture interfaces such as these hold the potential impact of improving the safety and reducing costs (associated with cleaning and sterilization) in medical domains such as these. Non-contact interfaces such as these are also useful for cleaner public computer interfaces such as interactive building directories.

A key challenge to making such interfaces robust, efficient, and practical to use, is to have them track human hand motions with a high degree of accuracy, both spatially and temporally. Highly accurate pose estimations can help create more robust gesture recognition by eliminating noisy finger or hand movements that might otherwise be falsely labeled as gestures. They also allow for more subtle movements to be permitted as gestures if there is more confidence that such subtle movements are not due to tracking errors.

The efficiency of gesture based interfaces can also be improved through design of the interface itself. Making the interface robust to the presence of hand-held tools allows for more streamlined workflows in situations such as medical examinations. Similarly, each user has a unique set of skills relevant to controlling an interface (e.g. dexterity, confidence, precision, etc.). By estimating key characteristics of a user's skill level in several dimensions, the system can adapt the user experience to better match their abilities. In the example of a gesture-based interface, for example, if an adaptive user model determines that a user does not have a high degree of dexterity, the interface can automatically adjust threshold values to be more tolerant of noisy movements at the expense of requiring larger motions. The end result is an individualized user interface experience that is better matched to the user's specific skills and abilities.

1.2 Problem Definition, Focus, and Scope

In order to improve the robustness of real-time CV-based hand-gesture interfaces, there are a wide range of key challenges to overcome. In this work, we narrow our scope to focus on two key challenge areas: 1) improving the accuracy of pose estimation and gesture recognition, and 2) designing interfaces to be robust and adaptive on a per-user basis.

In regards to the first challenge, robustly tracking the full pose of a human hand without markers or any kind of visual aids has been a difficult problem to achieve in real-time. Inter- and intra-hand occlusions, rapid motions, shadow/illumination changes, uncontrolled backgrounds, and the large number of degrees of freedom of the human hand all make reliable pose tracking quite difficult through marker-less CV techniques alone. However, recent advances in depth sensor technologies have helped alleviate many of the issues faced, especially hand/finger segmentation and issues with illumination or noisy backgrounds. In this work, we focus on real-time techniques to overcome challenges with occlusion in order to improve the reliability of gesture recognition.

In regards to the second challenge, there are many considerations that must go into designing the gesture interface itself. Naturally, the gestures must be designed such that they can be robustly recognized by the system, and be as unambiguous from each other as possible. However, in order to ensure as much flexibility as possible, we propose an interface design that freely allows users to interact in either bi-manual or uni-manual modes with or without the presence of unmarked handheld tools. We also explore the potential to improve interfaces for users through customized intelligent user models. By learning a model of each user over time, a more efficient, individualized experience can be achieved. Although we explain how our model can be applied to hand gesture recognition, we also demonstrate how it is more broadly applicable to other areas such as adaptive training.

The scope of our work is focused on gesture interfaces that control a display of some kind, such as a medical display, or some type of public display. However, while such systems were used as a test-bed to evaluate the effectiveness of our approaches, they are not limited to them. Similarly, while our work is generally not limited to any specific CV-based sensor technology (such as IR-patterned light, Time of flight, stereo-scopic reconstruction, etc.) our main focus is on newer state-of-the-art sensor systems that are capable of real-time sampling rates of over 100 pose estimations per second which thus have very tight computational time constraints to maintain real-time tracking.

1.3 Contributions

1) The first significant contribution of this work is a technique for improving the pose estimation accuracy of state-of-the art hand-tracking sensor systems when used for gesturing in front of a display. In this approach, we develop a novel technique that uses pose estimations from multiple sensors without the need to transmit video or depth images. Our experimental results demonstrate that our proposed technique significantly improves the pose estimation accuracy while still maintaining the ability to capture individual hand poses at over 120 frames per second.

2) Another contribution of this work is a technique for greatly improving the gesture recognition rate of real-time hand gestures used for controlling a display from a single-sensor. Specifically, we proposed, implemented, and evaluated a system which uses a novel customized Kinematic Constraint Model

and dynamic relabeling to vastly the reduce the parameter space of occluded keypoints on a tracked human hand, while still preserving a pose recognition rate of over 120 samples per second. Besides improving gesture recognition reliability for interactive visualizations, our technique can also be applied as a pre-processing step to other hand tracking systems with similar inputs and architectures.

3) In our investigations of gesture-controlled medical displays we have developed a gesture interface that is effective for real-time control while seamlessly allowing users to transition between bi-manual or uni-manual modes, and between using empty hands or handheld instruments. The results of our experimental studies in this area demonstrate that users are just as effective when gesturing using small handheld instruments as with empty hands in our gesture interface contrary to what is expected in light of their strong preference for interacting using only their empty hands.

4) In order to create interfaces that can adapt to their users' specific skill levels in a variety of dimensions, we create a user modelling technique based on Bayesian networks. We apply our proposed system to the domain of 3D virtual rehabilitation medicine, and show the effectiveness of our system through a user study.

1.4 Organization of the Thesis

The rest of the thesis document is organized as follows: In Chapter 2, we cover the background research and related works in the areas of human computer interaction, hand pose estimation, and gesture recognition. Chapter 3 discusses our approach for improving upon state-of-the-art hand pose estimation sensor systems through an analysis of multiple sensors at high real-time sampling rates without the costly bandwidth needed to transmit image data. Chapter 4 covers the design and implementation of our technique for improving the gesture recognition rate of gestures used for controlling displays while maintaining high real-time frame-rates. In Chapter 5 we present the results of our investigation into robust medical gesture interfaces that are able to adapt to the presence of hand-held tools during gesturing. In Chapter 6 we investigate interfaces that automatically adapt to users in an individualized manner. Finally, in Chapter 7 we state our conclusions and indicate directions for future work.

Chapter 2

Background and Related Work

In this section, we begin with a brief overview regarding how hand gestures relate to the topic of natural Human Computer Interaction (HCI). Next, the motivation for selecting a computer vision based approach is explained. Finally, key challenges regarding Computer-Vision based approaches are outlined as well as recent advances in techniques used for pose estimation and gesture recognition.

2.1 Gesture-based Interfaces

As computers continue to grow in computational power and graphical display capabilities, the limited means by which users interact with their systems threatens to prevent these advances from being used to their full potential [70]. Depending on the application domain, user efficiency and effectiveness can potentially be increased by novel, more intuitive interaction mechanisms. Specifically, allowing a user to interact naturally with a computer through the use of gestures (hand gestures in particular) has been recognized as a much more intuitive interaction mode for many applications [43].

Typically, most user interaction modalities introduce a tradeoff between expressiveness and ease of use; i.e. whereas a keyboard may be more expressive than a mouse, it is often easier to use a mouse for many data manipulation tasks [70]. Natural hand gestures are seen as an attractive alternative mode of intuitive human computer interaction because their dexterity and general-purpose usage offer a high degree of expressiveness [28].

Gesture-based user interfaces can offer a number of advantages over existing traditional interfaces [94] [92]. For example, systems using computer-vision (CV) gesture-based interfaces can have a reduced need for buttons or other physical interface components, thus reducing the size of the device. CV gesture-based interfaces can also be operated at a further distance from the actual computer hardware which can be quite beneficial in many situations. For example, interfaces used in store-fronts or public spaces can be protected from environmental damage or wear-and-tear sources (such as weather or vandalism). Finally, CV gesture based interfaces also avail themselves nicely to interaction with multiple users given that, theoretically, new users can enter and leave the interaction space freely as they would not require any dedicated interaction device. The lack of a physical interface to touch is also useful in situations where one wishes to avoid spreading biological contagions [1].

2.2 Implicit vs. Explicit Interaction

Human interactions with gesture interfaces can roughly be divided into two main categories: implicit and explicit.

Implicit interaction refers to human actions that are not intentionally conveyed by the user for input into a system, but which are recognized and used as input nonetheless [80].The motivation behind interpreting implicit interactions is based on the observation that a significant amount of communication between humans is exchanged implicitly. An example of implicit HCI is in the field of affective computing where systems attempt to determine the emotional states of users so as to respond more appropriately when user frustration is detected [22].

Explicit HCI, on the other hand, refers to user actions that are intentionally conveyed as input into the system, whether it is through a keyboard, mouse, speech recognition, or a gesture-based interface, etc. The focus of this work is on explicit gesture interactions.

2.3 Gesture Taxonomies

Hand gestures for explicit human computer interaction are commonly classified into three categories depending upon their usage [70]: deictic, manipulative, and communicative.

Deictic gestures refer to gestures designed to indicate an object or location, often to be used as the focus of context for subsequent action. One common example of deictic gestures would be selecting on-screen virtual objects by pointing directly at them with one's index finger.

Manipulative gestures refer to gestures where one or more parameters (such as location, orientation or scale) defining a virtual object are directly modified. Moving a virtual cursor with hand movements through a gesture-based interface environment is an example of a manipulative gesture. As the 3D position of the user's hand changes over time, the 2D or 3D position of the virtual cursor changes accordingly.

Finally, communicative hand gestures are those where the pose or movement of a user's hand is meant to communicate some information; for example, holding up 3 fingers to communicate the number "three" or sticking up a thumb to indicate approval. Several researchers have cautioned against heavy use of communicative gestures for natural HCI [96] [28]. This is because simple communicative tasks can typically be achieved much easier through a keyboard as opposed to forcing the user to memorize and perform a large number of hand poses or movements. Therefore, it is usually ideal to keep the number of communicative gestures small and as natural or intuitive as possible. Fortunately, research has also shown that for quite a number of 3D gesture-based application domains, a combination of only a few gestures is usually sufficient for completing many non-trivial tasks [40] [49] [48]. Nonetheless, balancing the tradeoff between how easy a gesture is to perform in comparison to how easy it is to recognize can be quite challenging.

2.4 Gesture Interface Challenges

Even if human hand poses could be tracked with perfect accuracy, there are still challenges in the interpretation of human motions that remain. Serveral well-known prominent challenges include:

Gesture spotting Distinguishing intentional, explicit gestures designated as input from implicit gestures that are not intended as input (such as a user lowering his or her hands because they are tired) is a classic problem in HCI systems [92]. Also known as the “Midas Touch” problem [9], the key challenge is to determine when to start and stop tracking human actions as input into the system.

User Fatigue Depending on the design of the gesture system, physical fatigue can quickly result from moving or holding hands and arms in awkward positions for extended periods of time [51].

Cognitive Load Similarly, requiring users to recall how to perform certain actions from a large set of complex gestures can result in a large amount of cognitive load which will slow down the efficiency of the system. A common approach taken to reduce this problem is to design hand gesture libraries that mirror (as closely as possible) the way that gestures are used in the real world for common interactions [66][78][82].

2.5 Multi Modal User Interfaces

When combined with other input modalities, such as speech, the expressive power of hand gestures increases further. Such interfaces that make use of two or more synchronized input modalities are referred to as Multi-Modal User Interfaces (MMUIs) [43]. MMUIs are important to consider because several studies have revealed that for quite a number of tasks, users tend to consistently choose to interact using multiple input modes into a user interface if the option is available [69] [40].

Performing gestures while speaking is one of the most commonly examined forms of multi-modal interaction due to its inherent naturalness and common use in human communication. Gestures that are performed while speaking (known as "gesticulations") can be useful for MMUIs in a number of ways [28]. Gesticulations typically come in two main forms: those that are complementary to the spoken information, and those that are redundant [80]. An example of a case where a gesticulation carries redundant information is as simple as a person saying the word "yes" while nodding. Both the gesture and the spoken word convey the same information, but if, for example, the spoken word was not recognized correctly, the gesture could be used to help determine the user's meaning and improve recognition accuracy [80].

An example of where a gesticulation carries complementary information to spoken words would be a user pointing at a location and giving the instruction "move right there". Here, the speech and gesture contain unique, complementary information; the speech indicates the action to be performed and the gesture indicates the location. The seminal work on using this form of interaction was achieved by Richard A. Bolt in 1980 [13] with his multimodal interface that allowed users to create, move and otherwise manipulate objects projected onto a large screen simply by pointing at them with their hands and issuing a voice command regarding the action to perform. Since then, many MMUI designs have incorporated this interaction technique with modified interaction designs and improved recognition accuracy [11].

2.6 Motivations for CV-Based Tracking

Although gesture recognition can be implemented through the use of peripheral devices such as IR pointers or other hand-held devices, the focus of this thesis is on empty-hand gestures that are tracked through computer vision techniques. Peripherals such as IR pens, motion sensitive devices, such as the Wii remote [78], or other handheld devices may be much easier to track but can restrict ad hoc user sign in and reduce the expressiveness of empty-hand gestures which are much more dexterous, natural to use, and have more degrees of freedom [28].

A commonly used, accurate, and reliable method of tracking hand poses and gestures in real-time for general purpose applications is through the use of data sensor gloves which have been successfully applied to a wide range of gesture-based applications [28]. Due to the high accuracy of data sensor gloves, they are well suited to tasks such as natural interaction with computer-aided 3D design (CAD) which is a highly spatial task involving interaction in 3-dimensions [3]. More recent implementations use data gloves for tasks such as immersive simulation and visualization [7]. However, data sensor gloves have a number of drawbacks that render them unsuitable for a large number of application domains. In situations where the desire is for users to freely enter and leave the interaction space, data gloves are not ideal. Furthermore, for the best results, they often need to be re-calibrated on a time-consuming, per-user basis [28][66].

Cost can also become an issue, particularly in the case of systems designed for multiple users where the number of data gloves needed must be at least equivalent to the number of simultaneous users the system supports. More generally, the most accurate data sensor gloves often present the problem of being restricted with regards to user motion, making even basic gestures more difficult to perform [82]. Pinch gloves, which measure when finger tips are touching each other, tend to be much less restrictive and are useful in certain data manipulation tasks [18], but convey less information, and are often less practical than simple handheld button interfaces [55].

2.6.1 Applications of CV-Based tracking

Several application areas have been identified where CV-based gesture interaction can be applied to improve the interface design. For example, in the field of medicine, gesture based interfaces can be used to control medical equipment, including surgical robotics in order to allow for more complex procedures or even to perform procedures remotely if the option to do it locally is not available [94]. Another example would be interfaces for use in surgical rooms where a surgeon can interact with a visualization system to view patients medical scans without the need to physically touch a computer. This avoids the need to constantly re-sterilize hands, as shown in Figure 2.1.

In the field of rehabilitation medicine, gesture tracking can be used to measure a patients recovery over time, and allow patients to practice rehabilitation exercises without the presence of a clinician, or allow clinicians to work with patients remotely who may then be treated at home [94]. Interactive games and training have also been proven to be effective in rehabilitation medicine in terms of increasing patient engagement and promoting improved rehabilitation results. Immersive 3D virtual environments are an attractive option for education and training as they enable educational or training tasks that are not possible or as effective in 2D environments [24]. Gesture based interfaces have been identified as potentially making interaction in such virtual 3D

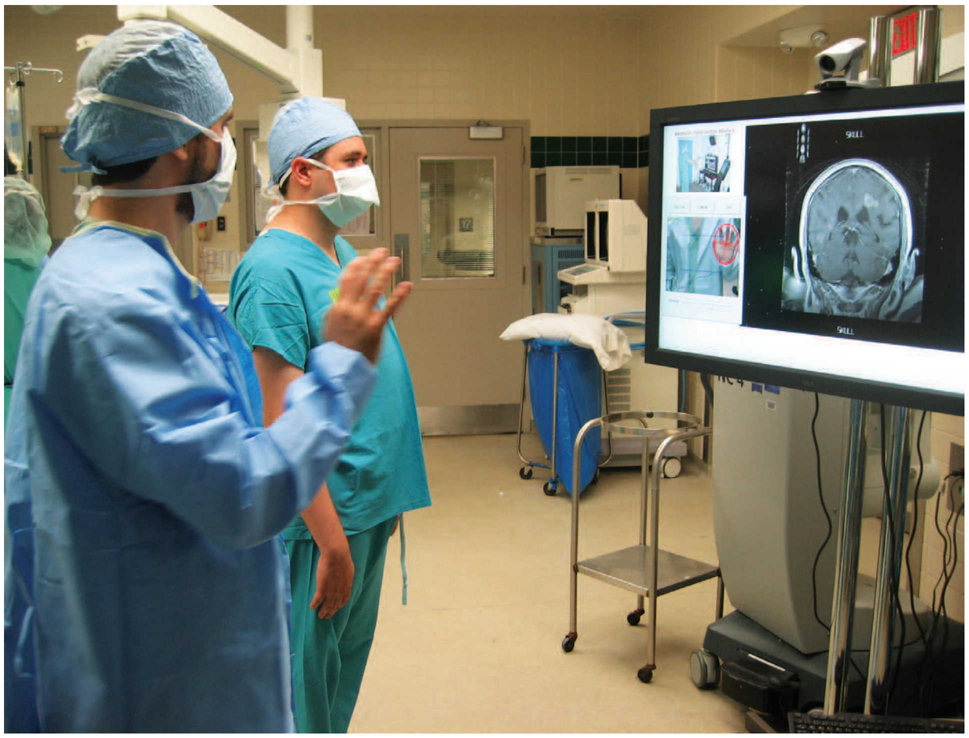


Figure 2.1: A touch-free gesture interface for interacting with Medical images (figure is from [94]).

environments much more natural and intuitive [9]. Gesture-based interfaces have also been considered as useful for more immersive and effective interaction with data visualization of large, complex, multi-dimensional scientific datasets [82] [59] [9].

Gesture-based interfaces have also been examined in the field of robotics. Human gestures are potentially seen as a much more natural mode of interaction with complex robots because they allow users to carry out robotic tasks in a more effective and straight-forward manner [78].

Hand gestures are viewed as particularly useful for interaction in 3D virtual environments given that to place an object in 3D typically requires about 6 DOFs (i.e. x, y, z position, and yaw, pitch, and roll for orientation) whereas a standard mouse interface offers only 2 by default [82]. Hand gestures offer a more immersive and natural way to manipulate 3D objects in virtual

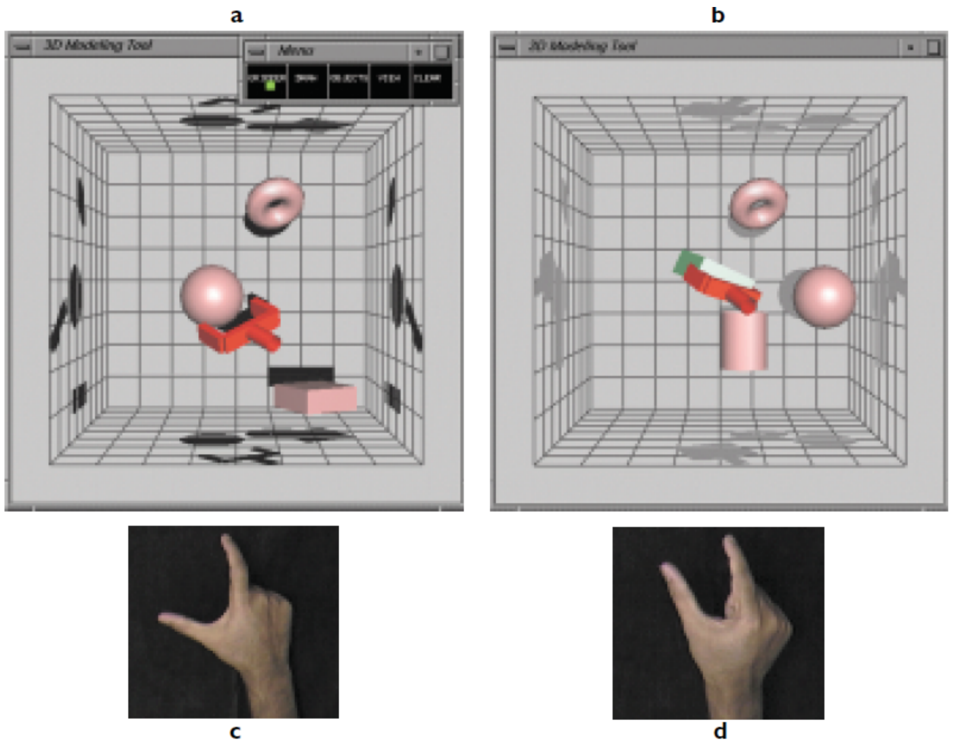


Figure 2.2: Gesture based manipulation of 3D objects (figure is from [82]).

environments as shown in Figure 2.2.

2.6.2 Challenges with Computer Vision Based Techniques

Intuitively, given that human beings can easily recognize hand gestures using only their eyes, it should theoretically be possible for a computer to accomplish the same through visual sensors alone. However, many computer vision-based tracking challenges exist which complicate this goal. The more prevalent and recurring issues are as outlined below:

Motion Blur Due to the fast movements human hands are capable of, tracking of fingers or hand poses during rapid motion can be quite difficult due to motion blur [92] as shown in Figure 2.3. Fortunately, research has shown that people tend not to change hand poses whilst the hand is in motion, which potentially simplifies the recognition tasks somewhat by making assumptions that finger poses will remain unchanged during quick movements [34].

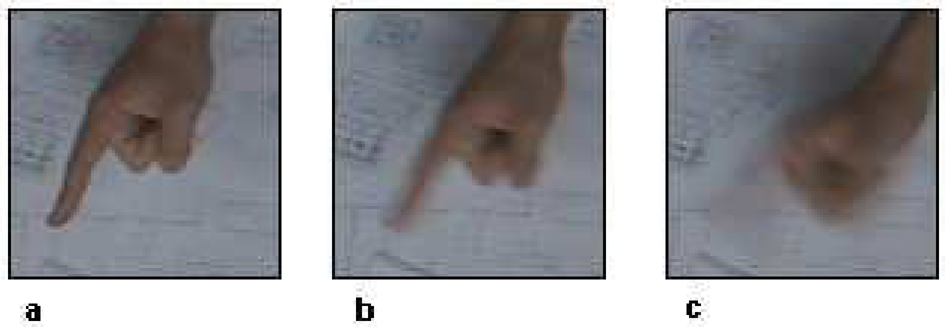


Figure 2.3: Motion blur induced by rapid motion (figure is from [92]).

Processing Speed Prior investigations have shown that in order for users to comfortably feel that a system is tracking their movements in real-time and free of lag, the delay between the input and response time should be less than approximately 45ms [94]. This can be challenging due to the high computation time involved in complex image processing tasks. Often, reduced or simplified recognition goals must be set to keep a system responsive in real-time [28].

Uncontrolled Background and Illumination Many gesture-based applications involve image backgrounds that are cluttered, dynamic, and otherwise uncontrolled making hand segmentation quite difficult [92]. While variations in skin color amongst different individuals is an issue for identifying hands, the issue can be complicated further by changes in illumination in the environment. The problem is particularly prevalent in projection-based systems where the ambient room brightness is often quite low in order to see the display properly, and can be made even more complicated by light from the projection display shining onto the user's hands as they are being tracked [49].

Finally, shadows cast by others or by the users themselves onto their own hands can cause illumination changes. Often, an approach taken to deal with the issue of illumination changes is to convert the video images from the standard Red Green Blue (RGB) colour space into the Hue, Saturation, and Intensity/Value (HSI or HSV) colour space [104][82][61] or the YCbCr colour space [61][53]. With this approach, the illumination intensity can be partially isolated to make the system somewhat more tolerant to changes in brightness (as shown in Figure 2.4).

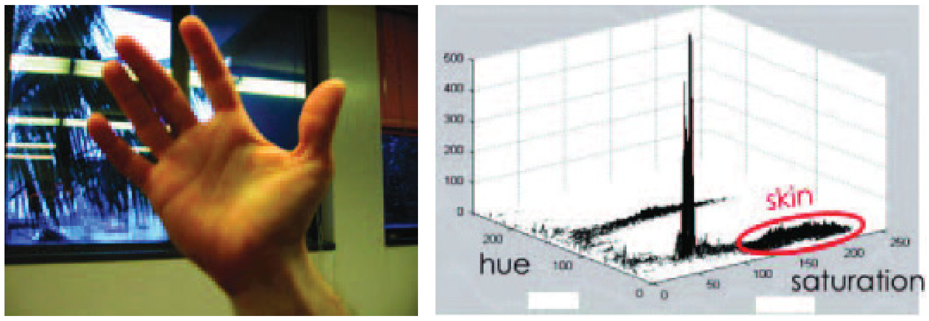


Figure 2.4: The Colour histogram of the image on the left is shown on the right (figure is from [94]).

High Dimensionality The tracking of hands is made difficult by their highly dimensional features including up to 24 or 27 degrees of freedom (DOF) as shown in Figure 2.5 depending upon the model used [59]. In addition, there can be a great deal of variation in users regarding the relative sizes and positions of joints in their hand, although research has shown that there is much less variation in joint angles between individuals [34] [68].

Self Occlusions When posing gestures, finger can frequently occlude one another, or the hand can occlude entire regions of itself [34] [59] [92] [104] [94]. In the case of multiple hands, it is quite easy for one hand to occlude the other and thus disrupt the tracking. To address this issue, many systems place restrictions on the gestures or postures allowed for

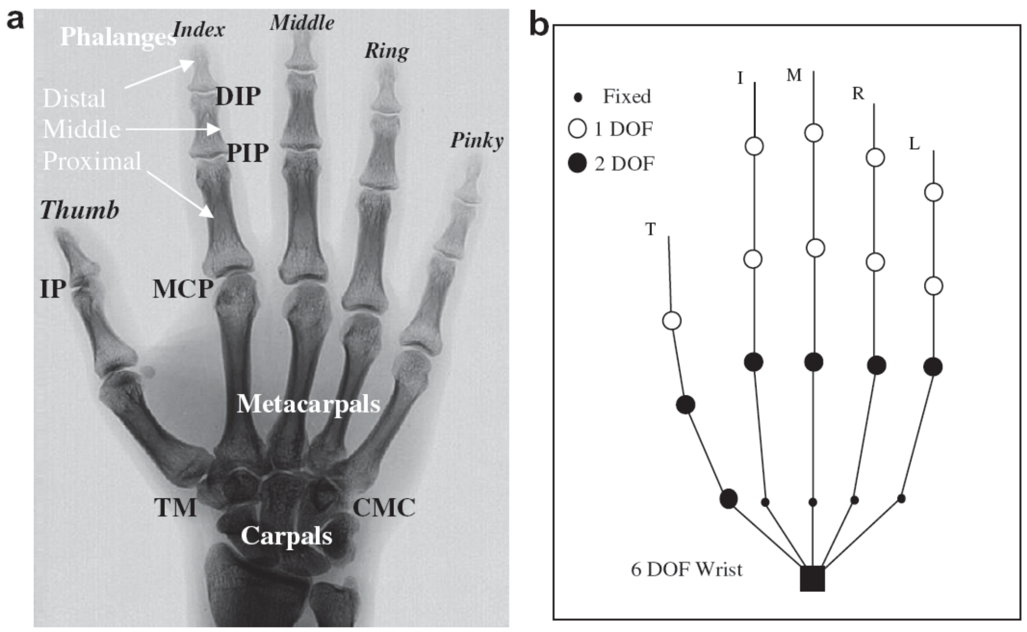


Figure 2.5: Examination of the kinematic structure of the human hand (figure is from [28]).

interaction, such as requiring that a user's palm always face the camera, or allowing only one hand to be used for interaction at a time [28].

2.7 Pose Estimation and Gesture Recognition

Computer vision approaches for hand tracking and gesture recognition generally fall into two major categories: Model Based approaches and Appearance/Image based approaches [70] as shown in Figure 2.6.

Model-based approaches attempt to determine the complete 3D position, orientation, and ariculation of a human hand. This typically includes the position and joint deflection angles of every joint in the user's hand as well. One of the primary advantages of real-time model-based approaches is that they are usually aimed at giving complete and continuous pose estimations regarding the entire kinematic structure of the hand [70]. This allows the system to be more general-purpose and useful for a much wider range of application areas. Unfortunately, a drawback of model-based approaches is that

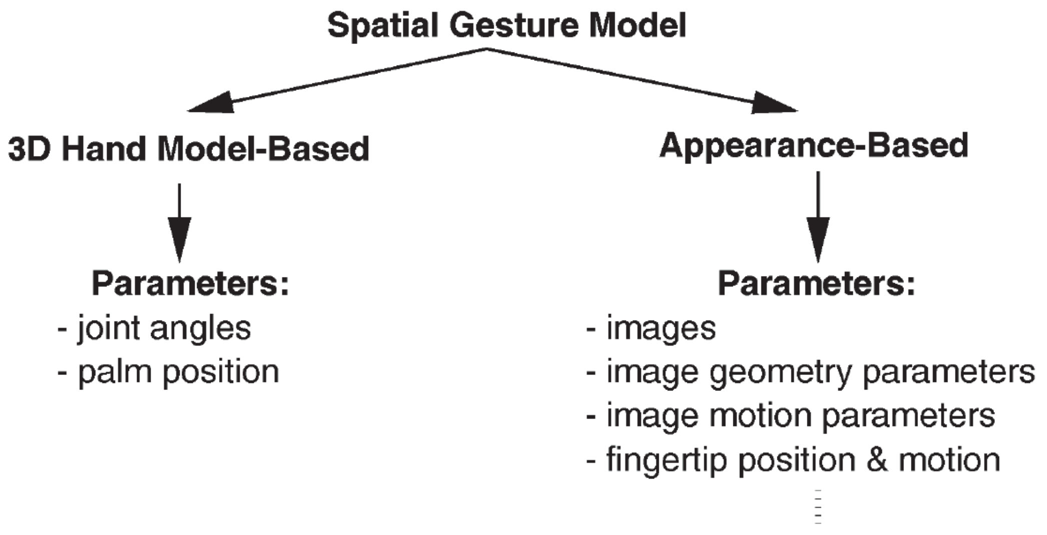


Figure 2.6: Breakdown of model vs. Image/Appearance based approaches (figure is from [70]).

in order to achieve precision, they are often computationally expensive, and do not operate in real-time. In recent years, however, by using new low-cost commercially available depth sensors [95] [83] and GPU programming techniques [68], real-time model based tracking for a wide range of arbitrary hand poses is becoming possible.

For most earlier real-time applications, however, model-based approaches do not offer a strong balance of computational complexity vs. accuracy. Instead, alternative appearance based approaches are used. Appearance-based approaches differ from model-based approaches in that they are not focused on full hand-pose estimation, but rather are only aimed at identifying a reduced set of hand parameters [70]. Appearance-based approaches tend to be much lower in computational complexity, and can readily achieve high accuracy real-time results at the cost of having a smaller, limited gesture vocabulary [104]. However, for many application areas (such as virtual object manipulation in 3D environments) the complete kinematic hand pose is generally unnecessary, and a smaller more modest gesture vocabulary is preferred in order to reduce the cognitive load on the user. A small set of gestures is usually sufficient for many tasks [40] [49] [48]. These factors have made the appearance-based strategy useful for some real-time gesture based systems, but the most appropriate approach is usually dependent upon the application domain in which it is used [94].

2.7.1 Appearance Based Methods

The first challenge in most appearance based methods is to localize and usually segment the user's hand(s) from the background. The most popular means to achieve this is to segment out pixel regions with colour histograms that are similar to human skin colour [45]. Once skin-coloured pixel regions have been determined, it is necessary to isolate the pixels that belong to the hand from pixels that belong to other parts of the human body (such as the arm or face) and noisy background regions that are only false-positives. One common approach used is to use the skin-coloured pixel region with the largest number of connected pixels and remove all others [39] [17]. One can also carefully control the image background (e.g. using a solid colour) to help improve hand segmentation, although this greatly reduces the practicality of the approach [31] [2] [86]. Alternatively, many past approaches restrict use-cases to a static, non-moving background in which case the foreground pixels (including the user's hand) can be segmented by simply subtracting away the known background image [17] [41]. In order to deal with dynamic backgrounds, previous researchers have used more advanced methods such as elastic graph matching [90], tracking multiple hypotheses [5], or more recently, depth data [75]. Although appearance-based approaches typically only locate the position of the hand in the 2 dimensional space of the image, some techniques make use of multiple cameras to find the 3D position [2] [102].

With the 3D position of a user's hand known, many different image features are typically used to try to understand the gesture being performed. Optical flow correlation has been used effectively in many past appearance based approaches for tracking dynamic hand gestures [17] [27] [103] [23] [26]. This feature is a way to model motion in the image as pixel regions flow in a correlated manner over a series of video images. Other methods include making use of edges and contours [17], symmetrically correlated movements [100], and orientation histograms [31].

2.7.2 Gesture Recognition

Hidden Markov Models (HMMs) have been the most widely used models for effectively detecting dynamic hand gestures over the past several decades [86] [54] [97] [64] [74]. In HMMs, one can model a system as being in one of a number of unknown (i.e. hidden) states and then make use of known measured values to try to predict which hidden state the system is most likely in. In the case of hand gesture recognition from appearance-based methods, the true pose of the hand is generally unknown (i.e. it can be in a number of hidden states) but we can attempt to predict the hidden state of the hand based on measureable appearance-based features. For example, to model a swipe gesture, the hidden states might include the hand being raised, moving from left to right, stopping, and then being lowered again. Based on measured features, such as optical flow [27], the HMM can be make updated prediction on what state the system is currently in. If the system predicts that the hand has traversed through all hidden states, one after the other, a gesture can be reported as having been detected. It is common to create a separate HMM network for each possible gesture the system can detect, and to update them in parallel [54].

While HMMs have proven themselves to be highly effect in gesture recognition systems, they are naturally limited by the quality of the features used as input. For this reason, many approaches pre-filter their data features before inputting them into the HMM (e.g. through the use of a Kalman filter [74]). Furthermore, the models must be trained offline for each gesture. This means the quality of the training set becomes another important factor in the overall performance of the system.

Other important techniques previously used for dynamic gesture recognition include Dynamic Time Warping (DTW) [21] [5]. Originally used in speech recognition [73], this technique allows for direct comparisons of feature trajectories despite different amounts of time being taken to complete each part. It is well suited for hand gesture recognition because there can be a great deal of variation in the speed at which users perform different parts of a gesture. Older techniques have also made use of Conditional Density Propagation (Condensation) to track gesture trajectories through noisy data [41] [12].

Another common approach is to build a rule-based recognition system. That is, for a given set of input features, there are manually defined if-else rules that determine when to report that a gesture has been detected [39] [23] [88]. While this technique avoids the need for any model training, manually encoding appropriate rules is difficult unless the gestures are simple or well-described by robust, accurate input data. The noisy and indirect nature of features from appearance-based methods usually makes the direct encoding of gesture recognition rules difficult except for simple movements such as large motions related to tracking the position of the centroid of a hand. If one had an accurate, continuous measure of the exact pose of a user's hand at all times, however, rule-based approaches could encode much more detailed gestures.

Static Pose Recognition

Besides dynamic movements, it is also meaningful to attempt to determine if the hand is in a certain static pose such as a pinch or some other communicative gesture. In this case, we wish to determine whether the hand is in a certain pre-defined pose based on a single static image. Previous work has made connections between this task, and the task of general object recognition (i.e. shape matching), where the silhouette of the segmented hand is matched against a reference template [8] [57] [58]. Other approaches involve decomposing the segmented hand image into a skeletal graph and using these skeletal graphs for comparisons of similarity [6] [75].

2.7.3 Model Based Approaches

Unlike appearance-based methods, model-based approaches to hand gesture tracking have generally been offline methods due to computational complexity. One of the eariler works in model-based tracking of an articulated hand was completed by Stenger et al. [87] who tracked the pose of a simpler 7 DOF hand model. His approach made use of an unscented Kalman filter to minimize the error between the edges of the rendered virtual 3D hand model, and the edges detected from the captured images. Approaches by Lin et al. [99] [56] used a Monte Carlo search method to try find optimal pose estimations in a highly-dimensional search space. Their method succeeded only when they made use of additional constraints on joint deflection angles to significantly reduce the search space.

Later work by Dewaele et al. [26] used a set of stereoscopic cameras to compute 3D point correspondences between the two images. A 3D representation of the hand is then modelled as a series of rendered spheroidal shapes. Gorce et al. [25] make use of hand texture, illumination, and shading as ways to help detect occlusion, and produce improved pose estimations. Finally, Bray et al. [14] made use of a particle filter approach [52] to estimate hand poses of a full 27 DOF hand model. Their technique can handle multiple hypotheses as it tracks hands over time, but, like previous techniques, can still result in hand poses being caught in a local minimum.

Real time Model Based Approaches

In 2011 Oikonomidis et. al. [68] claimed to be the first to develop a model-based approach for accurately tracking a full pose estimation of a human hand (27 DOF) at near real-time speeds of 15Hz (Figure 2.7). They achieved their approach through the use of a modified Particle Swarm Optimization (PSO) [72] [44]. Their PSO method works by starting off with a number of hand hypotheses with random velocities near an initial calibration starting pose. As the user moves their hand away from the initialization pose, the acceleration of the particles is adjusted to guide them towards a single particle (i.e. pose hypothesis) which is measured to have the best match with the sensor data. In order to enable good tracking results in real-time, the approach made use of a highly-parallelized GPU implementation to evaluate the pose hypotheses' depth maps against real-time depth data from a Kinect sensor. As with particle filter approaches, the technique is still vulnerable to being caught in a local minimum, particularly when there is a great deal of occlusion in the hand pose. However, the most significant limitation, due to the use of a PSO, is that users are restricted to only slow hand movements, because faster movements will cause the particle swarm to become desynchronized with the true hand pose, and will thus provided lower quality estimates.

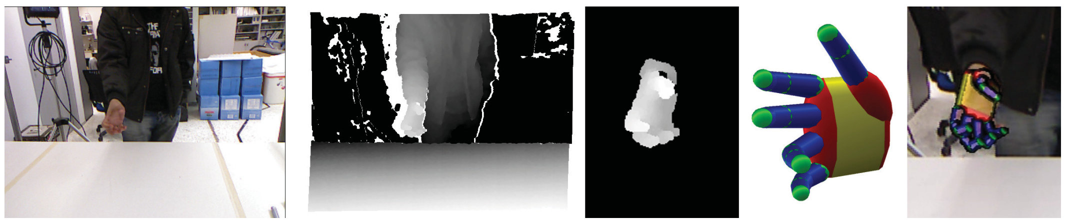


Figure 2.7: Outline of a real-time approach for model-based hand pose recognition (figure is from [68]).

In 2013, Keskin, et. al. [47] proposed a novel approach for real-time hand pose recognition using Random Decision Forests on a synthetically generated dataset of hand poses. Their approach is similar to the approach used by Microsoft Research to create the real-time skeletal pose recognition system of the Kinect [83]. Full pose estimation is achieved by using their classifier in order to label each depth pixel, and then mean-shift to identify the position of each part of the hand. The classifier is able to run at a rate of up to 30Hz where the bottleneck is due to the sampling rate of the Kinect depth sensor. However, their technique was only able to achieve the classification of a few discrete poses (namely, hand poses related to American Sign Language) although the authors claim that the technique should theoretically be able to classify hand poses in any arbitrary number of configurations given enough synthetically generated data to train their classifier. Their approach is not able to give values for hand poses in the continuous space of all possible hand configurations. This discrete pose limitation is also present in the real-time discrete hand pose system proposed by Romero et. al. in 2009 [76].

Chapter 3

A Multi-Sensor Technique for Hand Pose Estimation through Novel Skeletal Pose Analysis

3.1 Introduction

Interacting with computer interfaces through mid-air hand gestures is emerging as an intuitive and effective alternative to traditional touch-screen interfaces. For example, interacting with medical displays via touch screens or a traditional mouse and keyboard can present a major problem for medical professionals wishing to keep equipment sterile. Even in small clinical settings, examinations may often use fluids such as ultrasound conductive gel which users do not want to spread onto a physical interface by touching it [77] [35] [37] [93].

However, even as new hardware advances have been made in high-accuracy CV-based hand-tracking systems (such as the Leap Motion sensor system [95]) these state-of-the-art systems still have low pose estimation accuracy when the hand is in a pose that causes a high degree of occlusion between fingers. These situations frequently occur when the palm is not directly facing the camera, or when performing certain gestures such as pinches. This presents a problem for mid-air hand gesture recognition because intermittent disruptions in pose estimation accuracy can interrupt recognition of a dynamic gesture that was in-progress (such as a pinch-drag, for example). Likewise, false pose estimations can lead to the detection of gestures that were not actually performed, possibly leading to unintended interactions.

3.1.1 Our Approach

In this work, we present a novel technique that improves upon the hand pose estimation accuracy when using current state-of-the-art sensors for tracking hand poses. Our technique addresses the issue of occlusion by using pose estimations from multiple sensors placed at different viewing angles. One of the primary advantages of our approach is that we avoid the need to fuse sensor data at the 3D depth-map level which is not available for certain modern sensors such as the Leap Motion. We instead gain wider flexibility by intelligently analyzing each sensor system's independently derived skeletal pose estimations. A key challenge that makes skeletal pose fusion especially difficult is the tendency of the underlying pose estimation algorithms to settle into stable local minima that can nonetheless have a large amount of disagreement between each other. This renders most classical sensor fusion techniques, such as the Kalman filter [89], largely ineffective in this problem domain because it there will be no useful difference in noise profiles with which to make a meaningful weighting of incoming sensor data.

We instead present an alternative strategy by re-framing the challenge as a type of classification problem. Specifically, we demonstrate that by using a carefully-designed subset of the skeletal pose estimation parameters, we can build an offline model which can then be used in real-time to intelligently select pose estimations while still running at over 120 frames per second. In an experimental analysis involving two sensors, as shown in Figure 3.1, we analyse the pose estimation accuracy for a number of hand gestures typically used to control displays (i.e. pinch, tap, open hand) taken from a variety of different angles. In this context we demonstrate that we are able to achieve a 31.5% reduction in pose estimation error as compared to using only a single sensor and that, crucially, we are able to disproportionally eliminate the false poses that tend to interfere with gesture recognition.

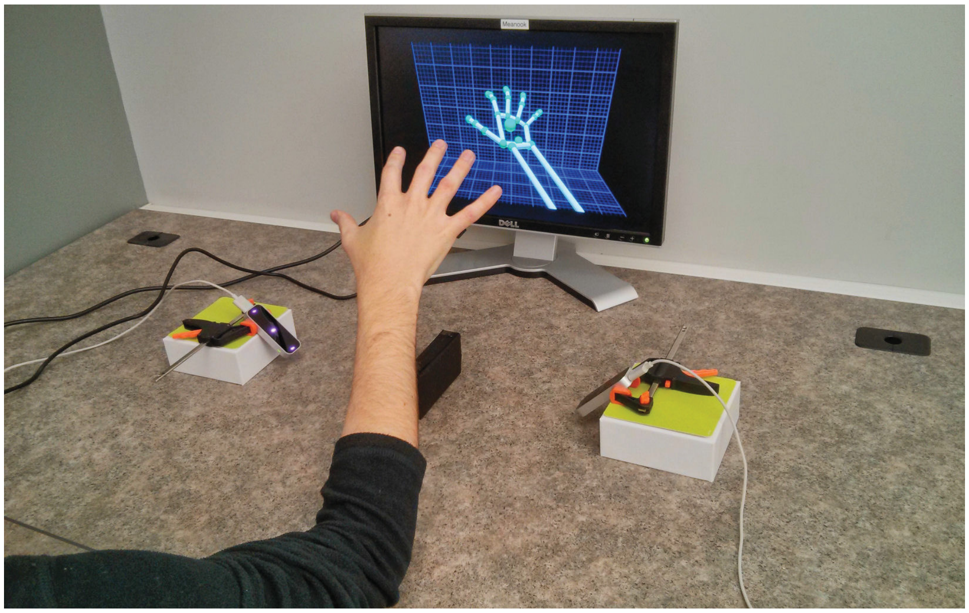


Figure 3.1: An example setup used in our lab to capture a single hand pose from two different viewing angles.

3.2 Comparisons with Related Work

Previous work in markerless CV hand-tracking has made use of colour and/or depth cameras (such as with the Microsoft Kinect [68]) in order to analyze either static or dynamic hand gestures in real-time. Using raw depth/colour data, features such as hand and finger positions/orientations can be extracted, and from this, an estimation of the hand's pose can be determined. The main drawbacks of these past approaches included a large degree of noise in the computed 3D positions of fingertips, and that they are not able to robustly capture small precise gestures (like quick subtle finger-taps) due to their low sampling rate and motion blur [92]. The reason for estimating human hand poses at high sampling rates is largely a result of the rapid motions that the human hand can regularly achieve. Previous work has reported that human hands can regularly reach movement speeds of up to 5 m/s and the wrist can reach rotational speeds of up to 300 degrees per second while gesturing normally [28]. The combination of high movement speeds, and low sampling rates often creates inaccuracies in gesture recognition because hand poses between successive frames become increasingly uncorrelated to each other as the hand moves faster [28]. It is for this reason that we tested our proposed method using the Leap Motion Sensor technology [95].

The Leap Motion Sensor is an example of a new generation of Computer Vision based sensors that provides state-of-the-art real-time hand tracking and pose estimation. Unlike past Infrared Red (IR) patterned light depth sensors (such as the first generation Microsoft Kinect), or Time of Flight sensors such as the Soft Kinectic DepthSense Camera, the Leap Motion sensor is able to provide much higher 3D positional accuracy for hands and fingertips (better than 1 millimeter of precision if near the sensor and un-occluded) and sampling rates in excess of 120 frames per second [95]. As a drawback of the Leap Motion's high resolution tracking and sampling rate, the sensor is not able to provide a full high-resolution depth maps at over 120Hz due to USB bandwidth considerations. The maximum effective tracking distance is approximately 50cm from the device.

3.2.1 Sensor Fusion

Regardless of the underlying hand pose recognition technique, or sensor used, occlusion inevitably becomes a problem in many situations if only a single CV-based sensor from a single point of view is used. For this reason, we propose analyzing results of multiple sensors placed at different viewing angles. Specifically, we propose directly analysing the skeletal pose estimations rather than the depth maps which may not always be available (as is the case with the Leap Motion Sensor which does not generate any depth maps or 3D point clouds at all).

A key challenge for analyzing these alternative hand poses is related to how they behave when in a high-occlusion situation. As previously mentioned, many skeletal hand pose estimation techniques follow a type of particle-filter approach. This leads to tendency for the pose estimation to enter a local minimum (i.e. a false pose that mostly matches the limited visible data), when there is a high degree of ambiguity in the hand pose. Here, the pose estimation model can remain in this stable but inaccurate state with little noise for a considerable amount of time, or even indefinitely if the user's hand is staying in a static pose. Essentially, this means that if a user performs a static hand pose, two sensor systems from two different viewing angles can result in substantially different pose estimations that are nonetheless both highly stable over time. An example, recorded from our sensor setup, is shown in Figure 3.2.

This characteristic of hand pose estimation techniques renders typical real-time sensor fusion techniques, such as the Kalman filter [89] unsuitable. This is because the similar noise profile of correct and incorrect hand pose estimations will result in them being approximately equally weighted [62]. It is also not feasible to use aspects of finger motion such as the angular velocities of finger joints to predict future positions given how rapid finger movements can be.

It should be noted that the current implementation of our proposed approach is similar to sensor fusion in that we take data from multiple sensors and produce a single output that attempts to be as accurate as possible. A difference in our approach compared to classical sensor fusion approaches, however, is that our classification step (which uses an Support Vector Machine [71] in our test implementation) results in only one sensor's data being used at any

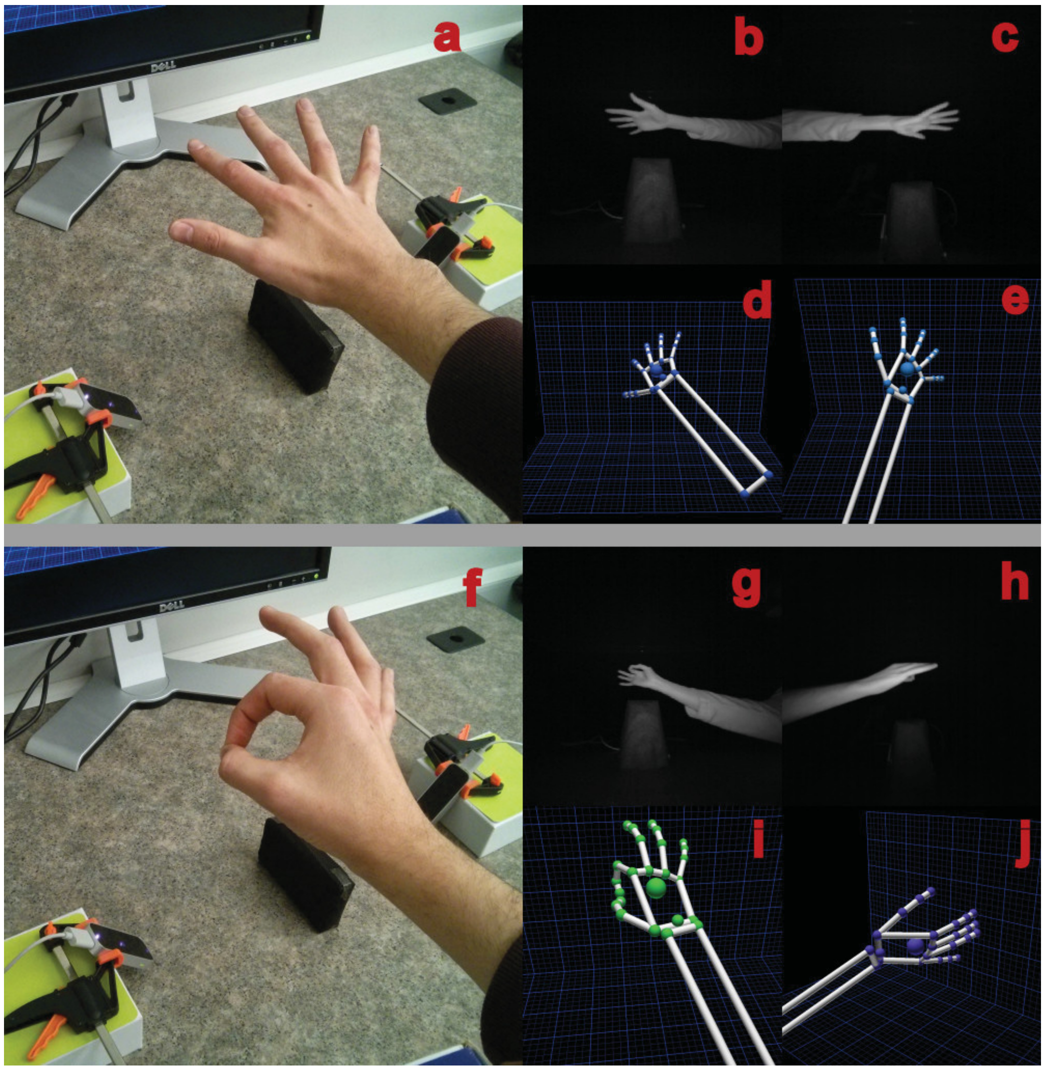


Figure 3.2: In the image on the top, the open hand pose (a) is readily visible by both sensors (b)(c) and thus each gives a similar pose estimation (d)(e) in their local sensor coordinate spaces. In the case on the bottom, however, the sensor on the left (g) has a fairly good view of the pinch pose (f) but this pose is mostly occluded from the right side sensor (h), which leads to large disagreements in the pose estimation (i)(j).

single time-step. The rationale for this is again related to the tendency of certain underlying sensor pose estimations to become trapped in a very poor local pose estimation. These low-accuracy estimations are removed completely by our classification step in order to prevent them from negatively impacting our final estimation.

3.3 Implementation

Our multi-sensor skeletal pose estimation approach is composed of several steps:

1. First, we use a trained Support Vector Machine [71] (SVM) model to intelligently determine the optimal pose estimation from an array of sensors. We build this model offline (only once) where the training set uses a feature vector composed of a subset of each sensor's output.

2. Next, we convert the computed single pose into a global coordinate system, so that the pose of all fingers is known in a single unified space. Similarly, we also compute the position of all fingers in a local hand coordinate system which provides key information for the pose estimation model we use.

3. Finally, the local hand pose information, and global hand pose information are fed into our gesture recognition model so that dynamic and static hand gestures can be tracked.

An overview of these steps is shown in Figure 3.3.

3.3.1 Sensor Array Setup

The goal in our implementation was to demonstrate the viability of our approach. We tested this by using a basic sensor setup involving two sensors. As shown in Figures 3.1 and 3.4, our sensor setup involves two Leap Motion sensors placed at a 45 degree angle to the table surface they rest upon. These angles were chosen so as to make the sensor view angles orthogonal in an attempt to optimize the amount of unique information available to each sensor during situations of high occlusion. The point at which the center of the field of view of both sensors intersects was set at 20cm for our experiment because

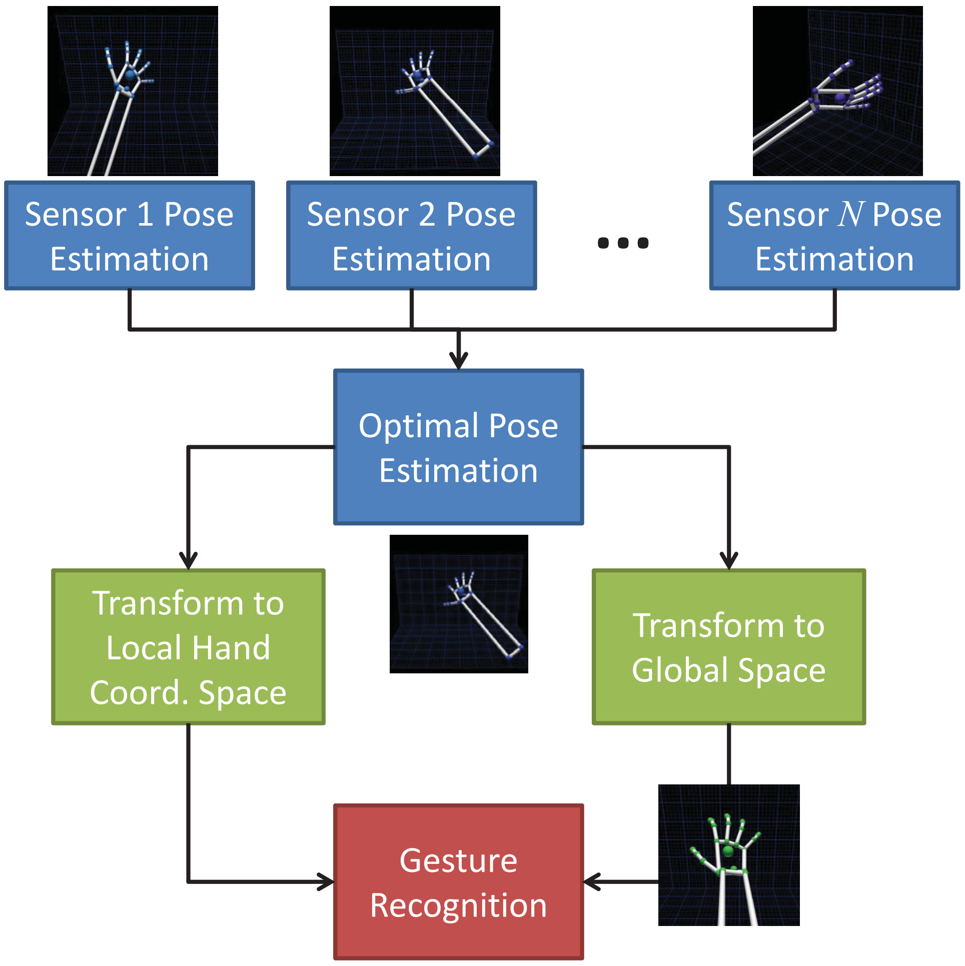


Figure 3.3: Overview of our approach during real-time execution.

this is the default pre-calibrated interaction height for the Leap Motion Sensor. It should be noted that our proposed approach is not necessarily tied to this particular sensor configuration. Depending on the specific user interface application, the sensors can be placed farther back from each other or at different viewing angles if necessary. Similarly, our approach is not limited to any specific number of sensors.

One consideration when using multiple Leap Motion Sensors for our implemented configuration is that the IR (infrared) projectors of one sensor can shine directly into the IR cameras of the other which can sometimes cause interference and unstable hand tracking. This can be solved by placing the sensors such that they lie outside the 150 degree field of view of each other or by simply placing a small non-reflective object somewhere in-between the sensors to prevent a direct line-of-sight. The latter is the approach used in our setup. As an alternative, if our approach is used with time-of-flight sensors, direct interference of this nature would not be an issue.

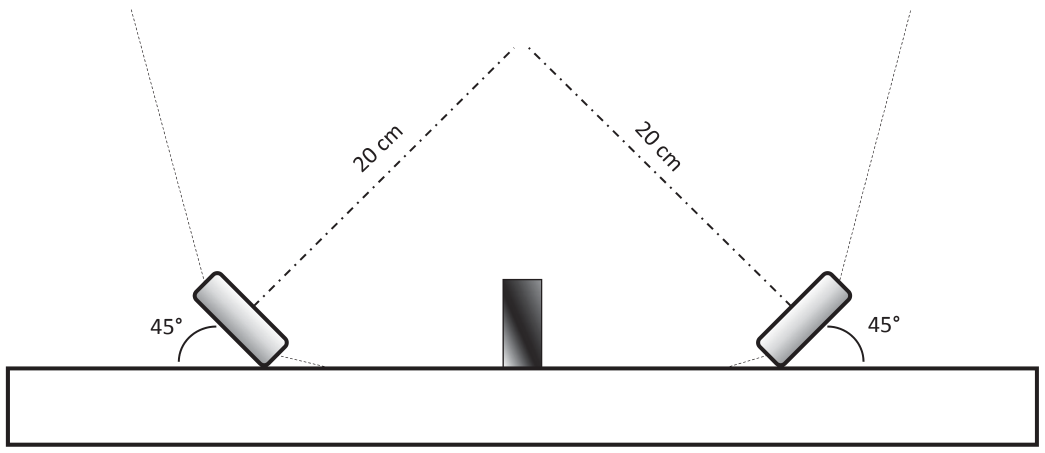


Figure 3.4: A Diagram of our implemented sensor setup. A small non-reflecting object is placed between the sensors to prevent interference.

3.3.2 Selecting Optimal Pose Estimations

In our approach, we define the amount of pose estimation error as the average of the euclidean distances (in millimeters) that each fingertip is from its ground truth position. The positions are expressed locally relative to the reported palm position and normal of the hand (provided directly by the sensor data in our setup case). That is, where $\mathbf{f_i}$ is a 3D vector representing the position of the fingertip of the ith finger on a hand, and $\mathbf{g_i}$ is the actual ground truth position, we then define the total error of a hand pose estimation as:

$$E = \frac{\sum_{i=1}^{5} ||\mathbf{f_i} - \mathbf{g_i}||}{5} \tag{3.1}$$

Given a set of pose estimations from a sensor array (2 sensors with 2 indpendent estimations in our setup), the goal at each time-step is to select the single pose estimation that has minimum total error. The issue is that there is usually no useful difference in tracking noise between accurate and highly inaccurate pose estimations which makes it difficult to decide which sensor's readings are more trustworthy. As explained previously, the stable inaccurate pose estimations are a common artefact of the underlying particle filter algorithms typically used to generate them.

The key novelty in our approach lies in that we intelligently determine which sensor is likely reporting the most accurate pose estimation. Specifically, we observed that even in situations with high occlusion and pose estimation error, the tracking of the hand palm position and orientation often remains accurate. Our system exploits this property in order to build a SVM model that learns which hand positions and orientations are likely to associate with higher pose estimation error. The SVM model uses a polynomial kernel function with a complexity constant of 1.0. Results were evaluated using a 10-fold cross-validation.

At run-time, the model composes a normalized feature vector based on each sensor's reported hand position/orientation in the local sensor space. This feature vector is then run through the pre-built classifier in order to predict which sensor is likely providing the best pose estimation. The selected pose is then passed along for subsequent processing and gesture recognition.

For our 2-sensor setup using the Leap Motion, the feature vector per sensor is composed of the following 12 floating point values per pose estimation:

1. X,Y,and Z values of the palm position relative to the sensor.

2. X,Y,and Z values of the palm plane normal relative to the sensor.

3. X,Y,and Z values of the "forward" direction of the hand (i.e. the direction the fingers point when extended).

4. The local ”Roll” rotation of the hand.

5. The dot product between the palm normal and the direction the sensor is facing

6. Sensor Confidence Estimation. This value is a floating point between 0 and 1 that is generated by an internal proprietary part of the Leap Motion API. Its limitations when used on its own are shown in the Evaluation section.

SVM was chosen as the classifier for our approach because almost all of the features used have a geometric meaning. We expect, therefore, that in the higher dimensional space of the palm position/orientation, it should be possible to find a relatively good hyperplane that separates the spatial regions in which each sensor would perform the best.

In order to build the training data for our model, we require a set of feature vectors that are already pre-labeled with which sensor performed best. Several strategies are possible for generating the training data for the model. If the setup did not use IR sensors (such as the Leap Motion) it may be possible to use an IR-based motion capture rig such as OptiTrack[1] with markers on the fingertips during the training process. Alternatively, data-gloves with only minimal error levels can possibly be used if they are not bulky enough to skew the results. In our Evaluation section, we demonstrate our approach through yet a third option whereby we use an articulated artificial hand model with a known pose to represent a ground truth for the training and evaluation data.

3.3.3 Re-Projecting into Global and Local Spaces

After a hand pose is selected from the sensor array, it must be converted into a unified global space in order to be interpreted for meaningful gestures or

[1]http://www.optitrack.com

interaction. The matrix used to determine the hand pose in global space can be defined manually if the sensor placement is known precisely enough such as in our proposed 2-sensor setup. More generally, for an array of sensors, the user can use several static finger positions in view of each sensor to build a set of 3D point clouds, and then apply an ICP (Iterative Closest Point) algorithm [10] to compute the matrix required to project the points from one sensor's coordinate system into the other. In parallel with the step above, we also compute a projection into a local hand coordinate space for purposes of aiding gesture recognition. The precise details of projecting the hand into the local coordinate space for gesture recognition is explained in section 4.3.3.

3.4 Evaluation

In our experiment, ground truth values for hand poses were determined by using an artificial hand model that was manually placed into fixed, known poses. Because the number of hand poses that could be created are virtually limitless, we focused on tracking three static hand poses that were of interest to us because of their use in controlling displays. Specifically, the three hand poses we tracked were:

1. An open hand.
2. A pinch gesture.
3. A tap gesture.

These gestures are shown in Figure 3.5.

The articulated artificial hand model used has dimensions similar to that of a male right hand. Fingers are approximately between 7.0cm to 10.5cm in length, and the entire length of the hand from the bottom of the palm to the tip of the middle finger is approximately 20.5cm when fingers are fully extended.

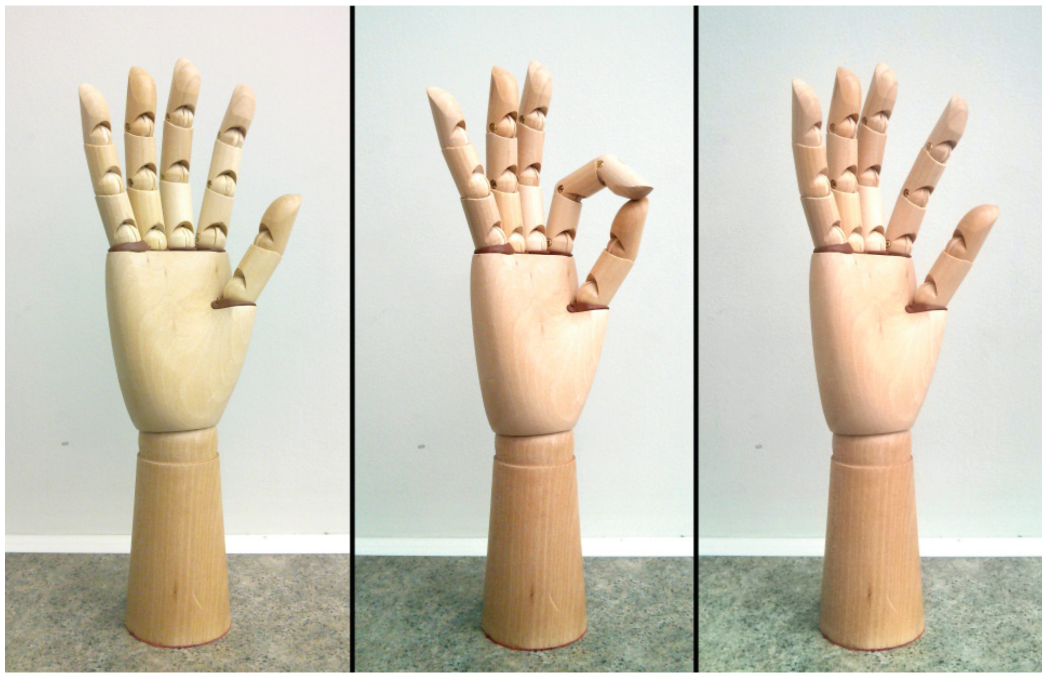


Figure 3.5: The three ground-truth hand poses tracked by our system.

Empirically, it was observed that the tracking accuracy of the artificial hand was roughly equivalent to that of a normal human hand.

In order to investigate the ability of the Leap Motion Sensors to track these hand poses as the hand moves, we fixed the hand position to be in the middle of the field of view of both sensors (20cm away) and slowly rotated the artificial hand counter-clockwise along the axis of the wrist as shown in Figure 3.6. The rotation was paused every 10 degrees in order to record a measurement from each sensor. This resulted in 36 pairs of pose estimations per gesture and 108 pairs of pose estimations in total.

We then computed the pose estimation error each sensor produces for all 108 cases. To train our model, we constructed all 108 feature vectors and labelled every case with the sensor that produced the lower pose estimation error. Note that the average pose estimation performance of both sensors are made identical because both sensors will eventually see all the exact same hand poses from the same distance and rotation angle (but not at the same time).

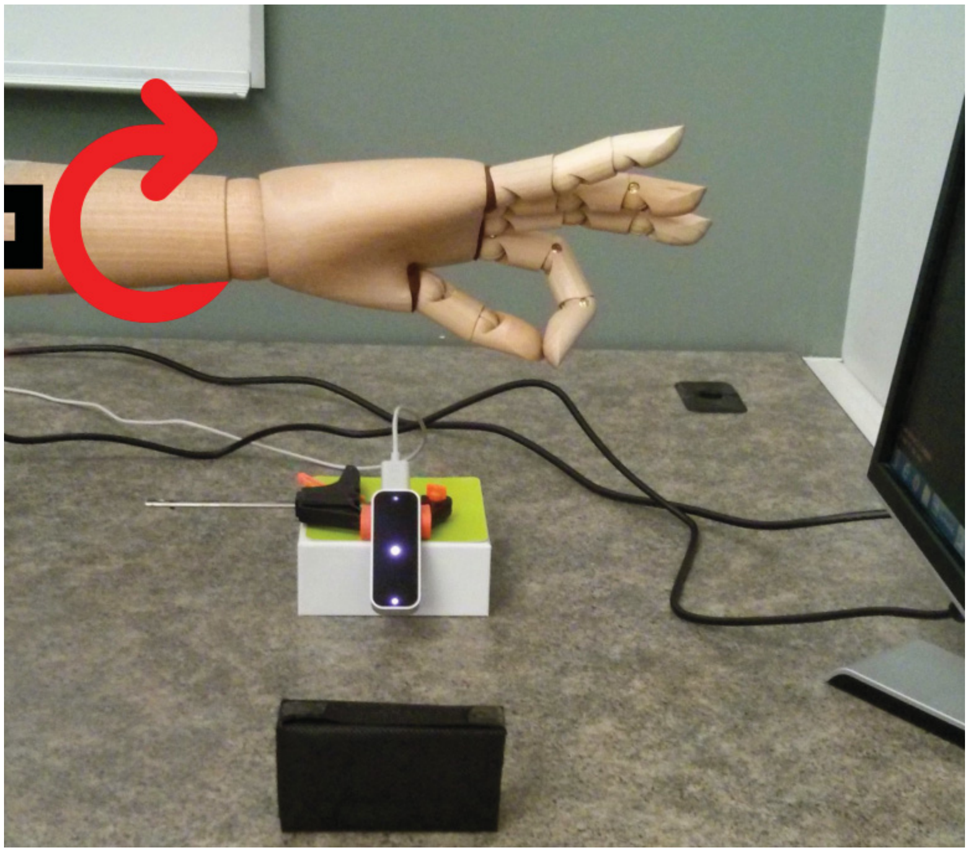


Figure 3.6: The artificial hand model was mounted such that it was fixed in space relative to the sensors and rotated along the axis of the wrist.

We evaluated the performance of our model using a 10-fold cross-validation across all 108 data points, which is a standard approach commonly used by other classification techniques.

3.4.1 Performance

Overall, the experimental results indicated that our approach reduced the total pose estimation error by 31.5% (in comparison to using only a single Leap Motion sensor). We summarize the performance comparisons in Figure 3.7 and 3.8. Results are reported in Table 3.1 and Table 3.2.

In our comparison, we computed values for the worst case and best case performance as 21.45 mm and 9.5 mm respectively. The worst case performance is a representation of the theoretical worst average pose estimation error that

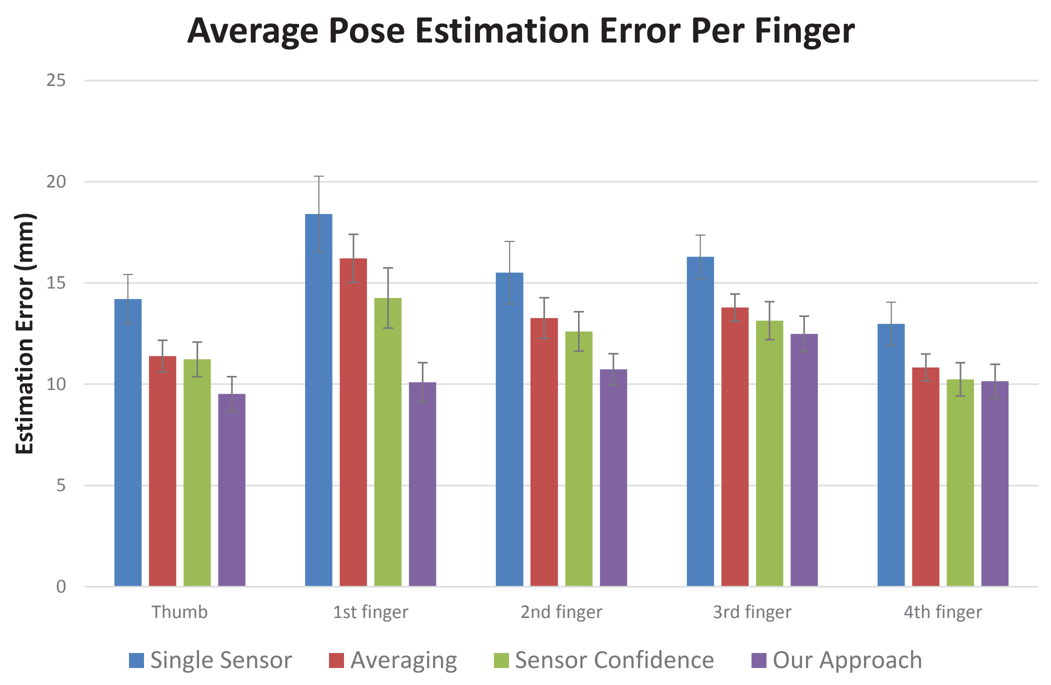


Figure 3.7: Comparison of average pose estimation error per fingertip between our approach and several others. Error bars indicate the standard error of the mean.

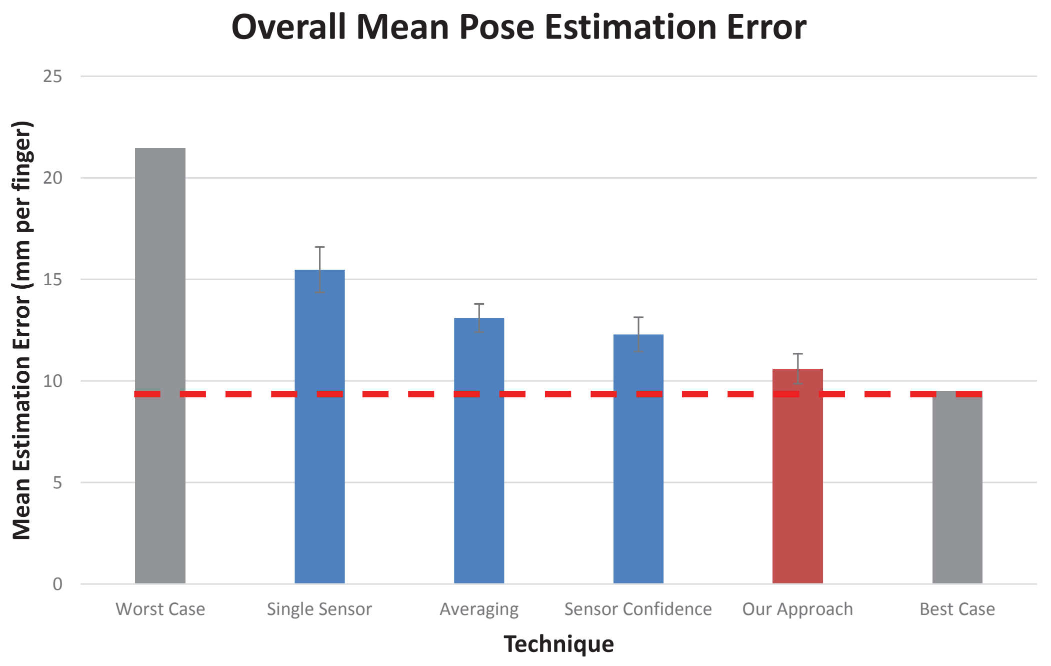


Figure 3.8: Overall performance comparison of our approach with several alternatives. Error bars indicate the standard error of the mean.

Table 3.1: Experimental Results

Technique	Finger	Estimation Err. (mm)	Std. Err
Single Sensor	Thumb	14.20	1.22
	1st Finger	18.40	1.87
	2nd Finger	15.51	1.55
	3rd Finger	16.29	1.08
	4th Finger	12.98	1.06
	Mean	**15.48**	1.12
Averaging	Thumb	11.39	0.78
	1st Finger	16.22	1.19
	2nd Finger	13.27	1.00
	3rd Finger	13.79	0.67
	4th Finger	10.83	0.67
	Mean	**13.10**	0.69
Sensor Confidence	Thumb	11.23	0.86
	1st Finger	14.26	1.49
	2nd Finger	12.61	0.97
	3rd Finger	13.14	0.94
	4th Finger	10.24	0.83
	Mean	**12.29**	0.85
Our Approach	Thumb	9.53	0.85
	1st Finger	10.09	0.97
	2nd Finger	10.74	0.77
	3rd Finger	12.49	0.88
	4th Finger	10.14	0.84
	Mean	**10.60**	0.74

could result from our approach. For it to occur, our classifier would need to incorrectly choose the sensor with the worse pose estimation every time (i.e. essentially have 0% accuracy). Similarly, the best case represents the result we would expect if our approach consistently managed to always choose the sensor with the smaller of the two pose estimation errors. In this context, our

Table 3.2: Overall Performance Comparison

Technique	Mean Pose Estimation Error (mm)	% Optimal
Worst Case	21.45	0%
Best Case	9.50	100%
Single Sensor	15.48	50%
Averaging	13.10	69.9%
Sensor Confidence	12.29	76.6%
Our Approach	10.60	90.8%

approach can be thought of as approximately 90.8% optimal. That is, our final result is about 90.8% of the distance from the worst case performance towards the best case.

3.4.2 Comparison to Alternative Techniques

The single sensor measurement shown in Figure 3.8 and Table 3.1 represents the average amount of pose estimation error that results from using only a single Leap Motion Sensor. Given the completely symmetrical nature of our setup, it is also the amount of pose estimation error that one would expect if choosing randomly between the two sensors. Although the mean sensor pose estimation error was approximately 15.48 mm, there was quite a large variation in estimation error across the dataset with a maximum recorded error of 114.24 mm, and the minimum recorded error of less than 0.07 mm. Generally, when the pose estimation error for a finger exceeded about 30 mm, the sensor was producing a pose estimation that was substantially different from the ground truth. There were about 22 such cases in our dataset of 108.

We also analyzed the performance of using unweighted averaging to fuse pose estimations. That is, for each finger in the local hand coordinate space, the final pose is the midpoint between the two sensor estimations. This technique similar to the results that one might expect a Kalman filter or similar technique to converge to after the differing pose estimations have settled. The results indicate that, to a certain extent, it can help in situations where it reduces the error from infrequent, poor pose estimations. A drawback to consider, however, is that if the system enters a state where the user's hand pose is constantly at a poor viewing angle for one sensor, the averaging will continually pull the final pose estimation away from the mostly accurate readings of the other sensor. In these asymmetric situations, one can expect that averaging in this manner could in fact reduce the overall accuracy in comparison to using only one sensor.

Finally, we also compare our approach to the strategy of using the Leap Motion sensor's self-reported confidence score. In this technique, the choice of which sensor reading to select is based on which one is reporting a higher confidence using a scale of 0 to 1. This approach did show some merit suggesting that there is some reasonable motivation to include it in the training model. By itself, it is shown to be not as effective as our approach which also takes the palm position and orientation into account. Also, not all sensor systems produce self-reported confidence scores.

Ultimately, a one-tailed paired T-test confirmed (with $\alpha = 0.05$) that our approach significantly outperforms all the aforementioned alternative techniques.

3.4.3 Analysis of Classification Performance

A unique aspect of our system's performance is that while our approach was about 90.8% optimal in terms of reducing the mean pose estimation error, our classifier accuracy was only 76.9% (i.e. 83 correctly classified instances out of the total 108). Normally, one would expect that this would result in the final performance being similar to this value. In order to investigate this, we compared the difference in pose estimation error between the correctly and

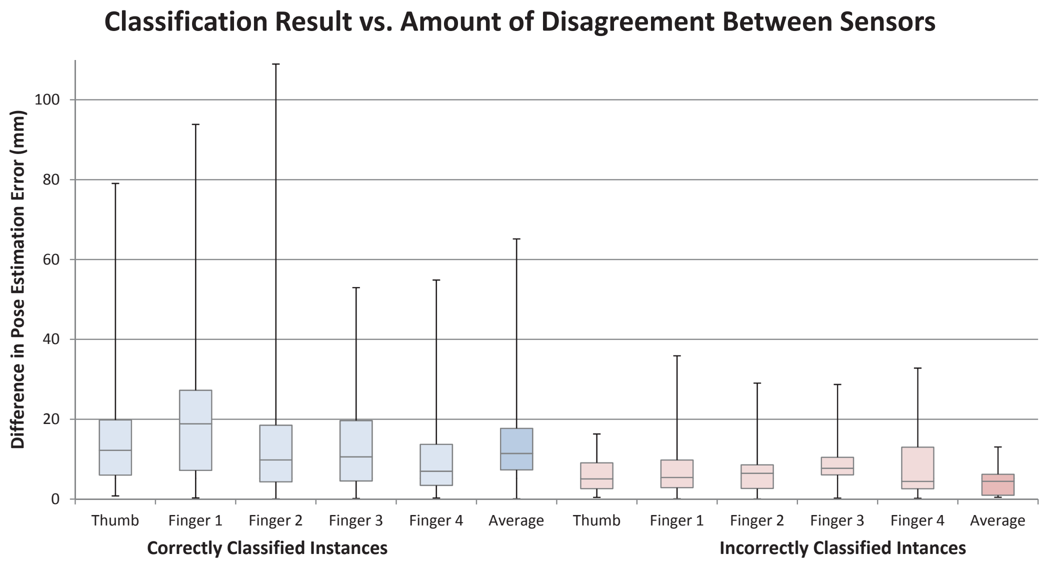


Figure 3.9: A box and whisker plot showing the distribution of the level of disagreement between the sensors for correctly and incorrectly classified instances. The analysis shows that cases where the instance was incorrectly classified tended to have proportionally smaller differences between the sensor measurements. The tops and bottoms of the boxes indicate the 1st and 3rd quartiles of the distributions respectively, with the line through the middle of the boxes indicating the median value. The top and bottom whiskers indicate the max and min values of the distributions.

incorrectly classified instances.

A large difference in the pose estimation error between the two sensors for a datapoint typically indicates that the sensors had a large disagreement between each other, and that likely one of them (but perhaps both) was producing a very low quality pose estimation. In cases where the difference between the two sensors was low, it indicated that both sensors were likely producing very similar pose estimations, and thus the choice of which sensor to use was less important. In our analysis, we computed the difference in sensor pose estimation errors for all our datapoints and show the corresponding distributions in Figure 3.9.

The analysis showed that for the correctly classified instances, the median difference between pose estimations was 11.44 mm. For the incorrectly classified instances, the median difference was only 4.47 mm. This indicates that in cases where there was a large disagreement between the two sensors, our approach tended to make the correct decision. The instances our approach missed tended to be cases where the sensors only had a small level of disagreement. This supports the practicality of our approach because it shows that it disproportionally eliminates the worst pose estimations, which we would expect would have the largest interruption to gesture recognition.

3.5 Summary

In this work, we presented a novel technique for improving full hand pose recognition accuracy from multiple sensors at framerates in excess of 120 pose estimations per second. Our technique achieves this through a model that is built to intelligently select the more accurate pose estimations based on a subset of the underlying pose estimation data of each individual sensor. One of the main applications for our technique is to improve the quality of tracking accuracy for gesture-controlled display interfaces. The results of our experiment show that we are able to reduce the overall pose estimation error for such gesture recognition by over 31% in a 2-sensor setup as opposed to the single sensor approach.

Our approach is independent of any particular sensor or underlying pose estimation algorithm, does not require the array of sensors to all be the same type (as long as their pose estimations are in a common format), and more importantly, it preserves the real-time performance of the system. Techniques which attempt to evaluate high-resolution images at high sampling rates from a large number of sensors quickly run into hardware limitations related to computational complexity and data bandwidth. In comparison, our approach is able to scale better with a large number of sensors because the data processed (i.e. the pose estimation models) are much smaller. One of the primary impacts of our work is that it demonstrates the viability of combining pose estimations from multiple sensor systems even without the presence of the underlying image data.

Chapter 4

High Precision Markerless Hand Gesture Tracking based on Kinematic Constraints and Re-Labelling

4.1 Introduction

Recent advances in integrating computer vision techniques with smart sensors, such as the Leap Motion Sensor, have made the tracking of human hand and finger movements more accurate at high sampling rates. However, these latest sensors present novel challenges for real-time gesture recognition due to processing time constraints and limited data output (i.e. only a small number of lossy keypoints). In this paper, we present a novel algorithm that improves the robustness of real-time computer vision gesture recognition from a single depth sensor. We achieve our results through a combination of kinematic constraints and computed heuristics to estimate the occluded keypoints and create a partial pose estimation of a user's hand. Finally, we apply our algorithm to the task of interacting with a display system through touchless hand gesture interaction alone. The results of our user study demonstrate that the proposed algorithm significantly improves the gesture recognition rate, while maintaining the ability to capture individual hand poses at over 120 frames per second. The direct impact of our work is on improving the practicality of touchless display systems. Applications include health services and education.

4.1.1 Problem and Motivation

Hardware innovations in new high-sampling rate CV-based hand-tracking sensor systems (e.g., Leap Motion) allow for millimeter-level precision for tracking fingertips, and sampling rates of over 150 frames per second [95]. However, the challenge of using newer sensors like Leap Motion is that certain versions (such as the commercially available v1 API) only provide the 3D position/direction of a user's hand and the position and direction of between zero to five fingertip-like features that are connected to a hand-like object. There is no depth map or image data availabe to use classical pose recognition strategies, and the extremely high sampling rates would make it difficult to process such images in real-time regardless. Furthermore, the actual pose of the hand is unknown because no labelling of the tracked fingertips is computed in these versions (i.e., thumb, index, middle, etc.) and no data is available regarding the position of obscured fingertips. This poses a problem for mid-air hand gesture recognition because the loss of tracking on a fingertip can interrupt recognition of a dynamic gesture that is in progress. What is ideal for robust gesture recognition is a 3D virtual model of the user's hands and fingers that persists even when the tracking of keypoints on the hand are occasionally lost. Motivated by these weaknesses, we propose a partial pose estimation system to recognize hand gestures real-time making use of kinematic constraints and motion characteristics.

4.1.2 Contributions

Resulting from the flexibility and low processing overhead of our partial pose estimation system, our algorithm can also be effectively applied as a pre-processing step to various touchless interaction interfaces which use a Leap Motion or similar sensor system to improve the robustness of gesture recognition.

Finally, we designed a novel gesture vocabulary to allow intuitive control of displays. Our interaction vocabulary is specifically designed around gestures that can be robustly identified and yet are easy for users to adopt because they use similar concepts as touch screen interfaces commonly deployed on handheld devices and smart-phones. The touchless control of medical displays is one of the orignal motivations for this work, but the techniques presented are generally applicable to other application areas such as education.

4.2 Comparisons with Related Work

In 2011 Oikonomidis et. al. [68] successfully developed possibly the first real-time, accurate, model-based approach for tracking the 3D position, orientation, and full articulation of a hand at over 15 Hz on high-end hardware. Their approach used a combination of video and depth images from a Microsoft Kinect sensor as input into a modified particle swarm optimization (PSO) algorithm. In contrast to our work, a drawback of their proposed approach is that it is not able to robustly capture small precise gestures (like quick subtle finger-taps) due to its low sampling rate, and the limitations of the sensor technology used. The aforementioned PSO technique is not a good match for our particular application not only because of computational costs, but also because there is no colour or depth map from the newer sensor systems we are focusing on. This results in no information on partially occluded fingers to guide the optimization towards the correct pose.

In 2013, Keskin, et. al. [47] proposed a novel approach for real-time hand pose recognition using Random Decision Forests on a synthetically generated dataset of hand poses. However, their technique was only able to achieve the classification of a few discrete poses, whereas in our application we are interested in the continuous space of all possible hand poses in order to maximize flexibility and future extendibility of our system.

4.2.1 Partial Pose Estimation

It is important at this point to make a distinction between full hand pose estimation, and partial pose estimation. The aforementioned works by Oikonomidis, Keskin, and others are full pose estimation solutions because they provide values for a 27 degrees of freedom hand model (i.e. the position and orientation of the wrist, and the joint deflection angles for every joint in every finger). On the other hand, our proposed hand pose estimation model is a partial pose estimation system. Partial pose estimation systems are concerned with tracking a reduced number of degrees of freedom of the hand [28]. For example, a gesture recognition system may only need to track the hand position, and the direction the index finger is pointing, or simply the hand position/orientation, and label each finger as either extended or retracted [4] [79] [67]. In contrast, our model is much closer to approximating a full hand pose estimation because we provide continuous values for the position and orientation of the palm, as well as the position and pointing direction of all 5 fingertips. What is notably absent is the joint deflection angles for the knuckles. However, using the position and orientation of the palm, and the 3D position and pointing direction of each fingertip, these values can be reasonably approximated to extend our model into a full pose estimation. For the needs of our application domain the joint deflection angles of the knuckles were not important for gesture recognition, and thus were not included in the model at this point.

4.2.2 Gesture Controlled Medical Displays

There has been a great deal of interest in the literature regarding the touchless (or non-contact) control of medical displays over the past few decades. The primary motivations include the reduced likelihood of spreading biological contamination if a medical interface is never physically touched, and the time/cost savings that result from reduced sterilization needs [37].

One of the earlier works in CV-based gesture control of medical displays was completed by Grange et. al. in 2004 [35] with their development of a system that controlled computer mouse movements in an operating room display via hand gestures. Their approach used stereo colour cameras and a combination of static background subtraction and pixel colour thresholding to locate and track the user's hand positions in 3D space. As the system did not track fingers, virtual mouse clicks were performed by either pushing the hand forward 20 cm, or holding it absolutely still for several seconds. Similarly, the Gestix system developed in 2008 [93] uses a nearly identical approach but does not control a virtual cursor and instead uses communicative gestures (such as hand swipes or circular motions) to perform various tasks.

As is typical of appearance-based approaches that use colour information to segment the user's hands, both of these approaches leave themselves vulnerable to segmentation errors due to changes in illumination, shadows, or dynamic backgrounds.

In 2011 Gallo et al. [33] proposed a Kinect-based interface for visualizing medical images in a sterile surgery room requiring the user to be standing, and using both hands for interaction. The system tracked the 3D position of both hands at 30 Hz and recognized the hand as either being in the open or closed state. As with the previous works, the inability to track the finger positions means that the gesture language requires large hand movements for recognition purposes, which can cause the user physical fatigue over time [82]. Another limitation of the system is that it requires both hands to be free for operation, which may not always be an option depending upon the situation.

4.3 Our Approach

In this work, we use lossy hand and fingertip keypoint data from a Leap Motion Sensor, which has the required precision, for persistent partial hand pose estimation in real-time at over 120 Hz. This allows us to have continuous fingertip positions and orientations in 3D space at all times.

The proposed algorithm uses sensor data to construct a local coordinate system for each hand tracked. When tracking on a finger is lost (frequently by occlusion or segmentation failure from contact with another finger) an estimate of the location of the lost finger is computed based on kinematic constraints and movements of the remaining tracked fingers as well as the hand itself. As lost fingers are rediscovered, the algorithm determines the identity of these fingers and labels them appropriately (e.g., thumb, index finger, middle finger, etc.). This labelling is computed based on positions in the local hand coordinate system, several finger features that are learned over time, and the known or estimated positions of the other fingers. As fingers move around or are lost and rediscovered over time, the algorithm can recognize labelling errors, and dynamically re-label fingers as more data becomes available.

4.3.1 Algorithms and Implementation

Our implementation is based on data output from the Leap Motion v1 API and sensor, which is an example of a new generation of CV-based hand tracking sensors. Unlike past Infrared Red (IR) patterned light depth sensors (such as the first generation Microsoft Kinect), or Time of Flight sensors such as the Soft Kinectic DepthSense Camera, the Leap Motion sensor is able to provide much higher 3D positional accuracy for hands and fingertips (better than 1 millimetre of precision if the finger is unoccluded and sufficiently close to the sensor) and sampling rates in excess of 120 frames per second. As a compromise for the Leap Motion's high resolution tracking and sampling rate, the sensor is not able to provide a full high-resolution depth maps at over 120Hz due to USB bandwidth constraints. Also, the maximum effective tracking distance is approximately 50 cm from the device.

The output of the Leap sensor v1 API includes the following data:
Hands:

- Position of palm in 3D space (in millimetres)
- Estimate of palm normal direction (direction the palm is facing)

Fingers:

- Position of fingertips in 3D space (in millimetres)
- Estimate of direction the finger is pointing
- Estimate of finger length and width
- The ID of the hand the finger belongs to (if the hand is visible)

What is notably missing is a classification regarding whether a hand is a left hand or right hand, and whether a finger is an index finger, middle finger, thumb, etc. Also, the aforementioned output is only available for hands and fingers that the sensor has an unobstructed view of. Fingertip positions are not reported by the sensor API if the view of the finger is obstructed by another finger, or often if two or more fingers come in contact with each other. This means that at any given time, the number of fingertips that a sensor is tracking and reporting values for may be much less than the expected total of 5. There is no data regarding the position or orientation of the untracked fingers, and no way to view partially obstructed fingers due to a lack of a colour or depth map.

The lossy tracking of fingertips poses an issue for identifying hand gestures because the tracking of a fingertip may be lost part-way through a gesture being performed, which interrupts and prevents its recognition. Even if tracking on a fingertip is just lost temporarily but then quickly restored, there is no way to be certain that it is the same finger as before. This is especially true in situations where the majority of the fingers on a hand are currently not being tracked.

What we require for robust hand gesture recognition is a real-time partial pose estimation of the user's hand state that can provide continuous values for hand and fingertip positions at all times. We achieve this through combining past tracking data and kinematic constraints with heuristics that can be computed in real-time. Finally, we outline our gesture vocabulary for the task of controlling displays with hand gestures. An overview of our approach is shown in Figure 4.1.

4.3.2 Estimating Untracked Finger Positions

In our depth sensor setup, tracking of fingers is generally lost in 3 situations:

1. Complete or partial occlusion by another finger or part of the palm: For example, the simple act of turning the hand sideways usually causes a great deal of occlusion from other fingers.

2. Coming into contact with another finger. For example, when performing the pinch gesture, the sensor may have a completely unoccluded view of both the thumb and index finger, but tracking of both fingers is usually lost due to an inability to identify the resulting shape as two distinct fingers.

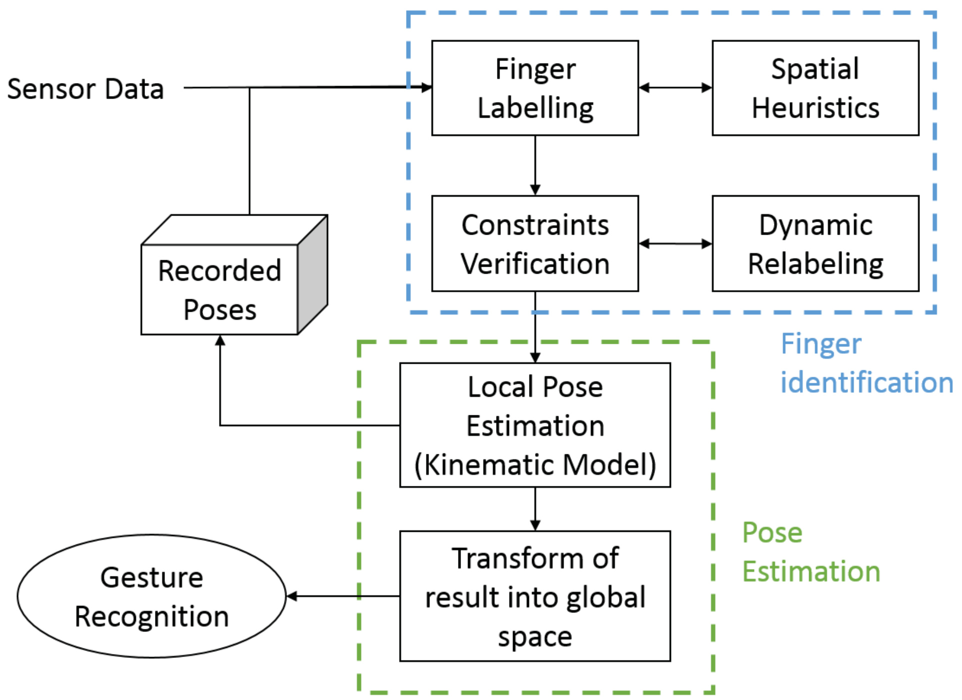


Figure 4.1: Overview of our algorithm.

3. Curling-up. If a finger is curled-up past a certain extent (such as the case where one is making a fist) it once again cannot be segmented and recognized as a finger.

Several real-world examples of cases where finger tracking is lost are shown in Figures 4.2 and 4.3.

4.3.3 Kinematic Model for Pose Estimation

In order to help estimate the pose of untracked fingers, we use the following key properties and human hand motion characteristics:

- Fingers generally tend to not move (relative to their hand) if the parent hand is moving above a certain velocity. Prior research has empirically proven that, when humans are performing fast hand movements while gesturing, fingers tend to remain stationary relative to the hand [34]. Following this we can assume that fingers tend to stay close to their last

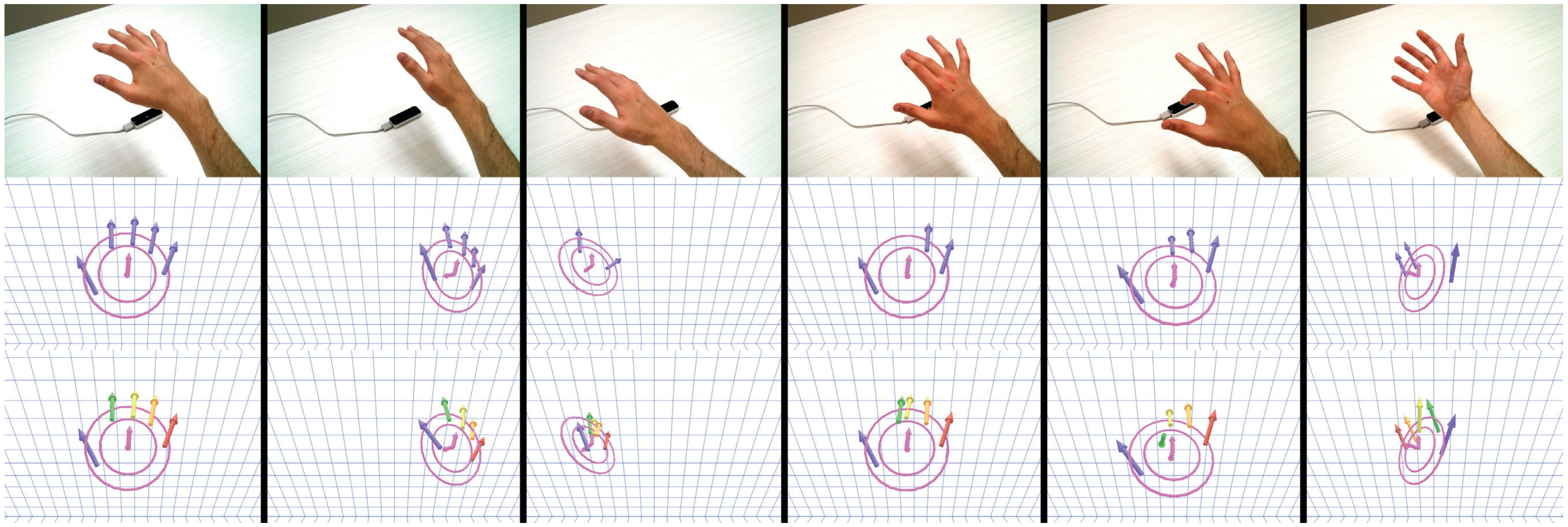


Figure 4.2: Top row: the hand pose made relative to the Leap Motion Controller on a normal tabletop. Middle row: a visualization of the raw data output from the Leap Motion API. Bottom Row: results of our proposed approach (note: the colour values in our approach indicate the identity of the finger). The figures show, from left to right: (a) tracking output when all fingers are clearly visible to the sensor, (b) good visibility at the start of a sweeping gesture, (c) loss of finger tracking for several fingers despite maintaining the exact same pose and only moving the hand (finger tracking is lost because the small finger is obscuring the view of the fingers next to it from the sensor). However, our approach is able to estimate the position of these missing fingers, (d) loss of tracking in the raw sensor output due to two fingers coming too close to each other, (e) temporary loss of tracking due to fingers pointing directly at the sensor (this tracking failure presents a major issue for tracking tapping gestures), (f) the hand at an angle and in the inverted position.

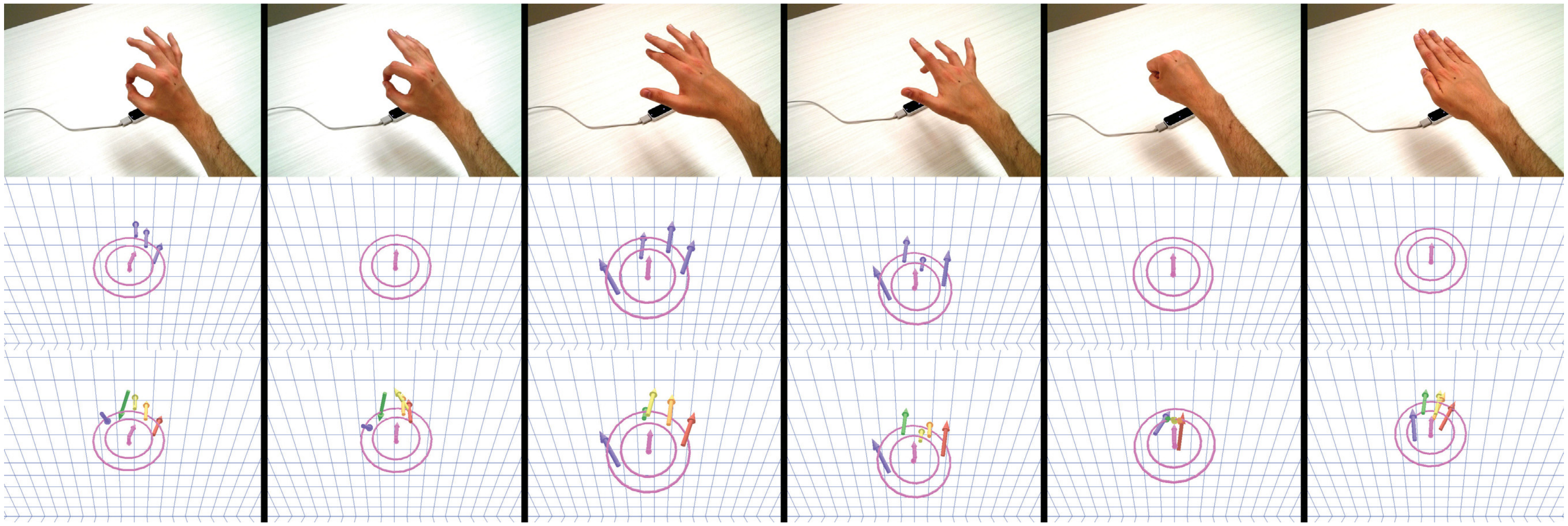


Figure 4.3: Top row: the hand pose made relative to the Leap Motion Controller on a normal tabletop. Middle row: a visualization of the raw data output from the Leap Motion API. Bottom Row: results of our proposed approach (note: the colour values in our approach indicate the identity of the finger). The figures show, from left to right: (a) the Leap Motion API looses tracking on both the index finger and thumb during a pinch gesture, although our persistent model is able to preserve their positions, (b) in this pinch gesture, all the remaining fingers in the gesture are touching which typically leads to a complete loss of tracking on all fingers, (c) tracking on the middle finger is temporarily lost due to occlusion. Without our technique, it is difficult to determine if the finger in the center is the index or middle finger using the sensor data alone, (d) similarly, the middle finger is occluding the ring finger in this pose, often this can result in the middle finger itself being untracked too, (e) and (f) a closed fist and halt gesture are normally indistinguishable from each other or from gestures like (b) without our approach.

known position relative to the hand if the hand is moving. In contrast, if a hand is mostly stationary, we can expect that the loss of tracking is likely due to the motions of the fingers instead of the motions of the hand.

- Fingers are very limited in their side-to-side motion. Crossing one's fingers is generally not a situation that occurs if one is gesturing naturally (though it may occur if one is performing an intentional communicative gesture). Much of the previous work in hand pose estimation models assume little to no side-to-side motion of the fingers [62]. This allows us to make several assumptions about the labelling of unknown fingers based on their positions relative to each other.

Our first step in determining the position of untracked fingers involves defining a local hand coordinate system in order to convert the fingertip positions from world space into a local hand coordinate space. If we consider a hand with a normalized palm direction vector $\hat{\mathbf{h}}_N$ (i.e. normal to the plane of the palm and in the direction the palm is facing), and a normalized orthogonal forward vector $\hat{\mathbf{h}}_F$ we can compute an additional basis vector $\hat{\mathbf{h}}_C$ for the local hand coordinate system as the cross product of these two:

$$\hat{\mathbf{h}}_C = \hat{\mathbf{h}}_N \times \hat{\mathbf{h}}_F \tag{4.1}$$

The main challenge in this prior step involves determining $\hat{\mathbf{h}}_F$, the "forward" direction of a hand. Multiple definitions are possible, but given the sensor data available, we propose to define the forward direction of the hand based on the mean direction of all tracked fingers. This means that for n tracked fingers with normalized direction vectors $\hat{\mathbf{d}}_0, \hat{\mathbf{d}}_1, ..., \hat{\mathbf{d}}_n$ we compute the unnormalized mean forward direction $\mathbf{d}_M$ as shown in equation 4.2.

$$\mathbf{d}_M = \frac{\sum_{i=1}^{n} \hat{\mathbf{d}}_i}{n} \tag{4.2}$$

We then project this computed vector onto the palm normal plane (provided directly from the sensor data), and normalize the result as shown in equation 4.3.

$$\hat{\mathbf{h}}_F = \frac{\mathbf{d}_M - (\hat{\mathbf{h}}_N \cdot \mathbf{d}_M)\hat{\mathbf{h}}_N}{||\mathbf{d}_M - (\hat{\mathbf{h}}_N \cdot \mathbf{d}_M)\hat{\mathbf{h}}_N||} \tag{4.3}$$

With the 3 basis vectors of the local hand coordinate system computed, and the known position of the hand palm (T), we construct a matrix M that transforms fingertip positions and direction vectors from the global coordinate system into our defined local hand coordinate system.

$$M = \begin{bmatrix} \hat{\mathbf{h_{Cx}}} & -\hat{\mathbf{h_{Nx}}} & -\hat{\mathbf{h_{Fx}}} & T_x \\ \hat{\mathbf{h_{Cy}}} & -\hat{\mathbf{h_{Ny}}} & -\hat{\mathbf{h_{Fy}}} & T_y \\ \hat{\mathbf{h_{Cz}}} & -\hat{\mathbf{h_{Nz}}} & -\hat{\mathbf{h_{Fz}}} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \tag{4.4}$$

Note that the directions of the hand normal and forward vectors are inverted when used as basis vectors. This is because the coordinate system of the global sensor space defines the y axis as pointing upwards, and the z vector as pointing towards the user whereas the palm normal is defined as facing downwards, and the fingers (used for the forward direction) normally point away from the user.

Note that the transformation matrix is a 4×4 matrix because we are using a typical homogeneous coordinate system. This means that 3D points are augmented with a 1 at the end before multiplication, and direction vectors are augmented with a 0.

Using this setup, when the tracking of a finger is lost while the hand is moving, we compute its position in our local hand coordinate space, and assume that it will remain at its original position in this space. The estimation of its position in the global sensor coordinate space is continually updated using the previously defined transformation matrix, and its stored last known position in local space.

4.3.4 Classifying Fingers

Although the previous step provides position and orientation values for fingers that become untracked by sensors, a problem arises when dealing with multiple untracked fingers. Consider the situation where only the thumb and index finger of the right hand are being tracked: if we regain tracking on a third finger that is somewhere to the right of the index finger, we cannot be certain whether this finger with newly-restored tracking is the index, ring, or small finger. Furthermore, if a newly tracked finger on the same hand is discovered to the left of the thumb, it implies that the current labelling is incorrect, and fingers must be relabelled on the fly. This latter case is particularly common when a new hand is entering the field of view of the sensor for the first time.

In order to classify newly rediscovered fingers on a hand, we implement a two-stage approach that first uses kinematic constraints that dynamically re-classifies existing fingers if necessary, and a second stage that uses computationally fast heuristics.

In our first step, from the list of all possible candidate classifications for a finger, we filter out classifications that break the kinematic assumptions of our hand. For example, if we have a situation where the middle finger and thumb are fully tracked on the right hand, and the newly rediscovered finger has a position in our local coordinate space that is to the right of the middle finger, then we can eliminate the index finger as a possible classification (naturally, the thumb and middle finger are also eliminated as possible classifications because they are already being tracked).

If this kinematic constraint filtering stage eliminates all possible candidate classifications except for one, then we can immediately assign the sole remaining candidate to the newly rediscovered finger. However, because classification errors can occur, we can frequently find ourselves in the situation where no candidates remain that do not violate our kinematic assumptions. Consider, for example, the situation where we are tracking three fingers believed to be the thumb, index finger, and ring finger on the right hand. If we discover a new finger, but its local position is to the right of the thumb and the left of the index finger, then we have most likely made one or more classification errors at some previous timestep, and the labels on our currently tracked fingers must be updated.

4.3.5 Dynamic Relabeling

The relabeling algorithm works by finding the closest labelling scheme that preserves all of our kinematic constraints and assumptions while altering the minimal number of existing finger labels (i.e., we assume that the most likely correct labelling scheme is the one that is closest to our current prediction). The formal breakdown of this algorithm is as follows:

Assume all currently tracked fingers $F = [F_1, F_2, ...]$ are assigned an integer index value label between 0 and 4 (i.e., $label(F_n) = 0, 1, 2, 3, 4$) where $0 =$ thumb, $1 =$ index finger, $2 =$ middle finger, etc. Assume U_0 is the newly discovered, unlabelled finger that we just started tracking, and must assign a label to, and then insert into the sorted array F.

DYNAMIC RELABELING ALGORITHM:

1. Project the 3D position of U_0 and all fingers in F into our local hand coordinate system.

2. Find the finger F_L in F where F_L has the minimum Euclidean distance from U_0, and is also on the left side of U_0 (if such a finger exists). Similarly, find F_R for the right side.

3. Calculate N_L as the number of finger index values away the nearest unused index label is on the left (if any). Similarly, calculate N_R for the

right side. For example, if only two fingers are tracked, such as the thumb and index finger, and the newly tracked finger U_0 is discovered to be physically in-between these two fingers then, we would have $label(F_L) = 0$, $label(F_R) = 1$, $N_L = MAX\_INTEGER$, and $N_R = 1$. Technically, the value of N_L is undefined because there are no unused labels to the left of the thumb, so we set it to a sufficiently large value to indicate this. In all of these steps, the left and right directions are swapped for the left hand versus the right hand.

4. If $N_L < N_R$, then we set $label(U_0) = label(F_L)$ and repeatedly re-assign $label(F_n) = label(F_n) - 1$ for a total of N_L times starting from F_L and using decreasing values of n. Otherwise, If $N_L \geq N_R$, then we set $label(U_0) = label(F_R)$ and repeatedly re-assign $label(F_n) = label(F_n) + 1$ a total of N_R times starting from F_R and using increasing values of n.

The reason to favour pushing finger labels to the right instead of the left (on the right hand), in the case of a tie where $N_L = N_R$ is that empirically it was discovered that labelling errors were more common on the right side of the right hand where fingers tend to be closer together, thus resulting in less confidence in the accuracy of these labels. This left-right direction bias is swapped for the left hand versus the right hand.

4.3.6 Heuristics

In case the kinematic constraint filtering step results in multiple ways to label a newly tracked, unlabelled finger, we apply a set of computationally fast heuristics in order to predict the identity of the finger. This situation is most common in cases where tracking on all or nearly all fingers has been lost due to a recent gesture such as a closed fist, or rotating the hand such that the sensor lies on the plane of the palm.

Amongst the possible candidates that the new unlabelled finger may represent, the chosen candidate is based on a combination of the Euclidean distance between the unlabeled finger and a candidate finger's last known position (in the local hand coordinate space), and differences in the estimated width and length of the fingers. The formula for determining which candidate finger identity to assign is given by Equation 4.5.

$$h(C) = \begin{array}{c} W_L(C_L - U_L)^2 + W_W(C_W - U_W)^2 \\ +W_D\sqrt{(C_x - U_x)^2 + (C_y - U_y)^2 + (C_z - U_z)^2} \end{array} \tag{4.5}$$

Where U_L and U_w are the estimated length and width of the unlabelled finger (output by the sensor), and U_x, U_y, U_z represent the 3D position of the unlabelled finger in 3D space. Similarly, C_L and C_W are the estimated length and width of the candidate finger C, and C_x, C_y, C_z represent the last known 3D position of candidate finger C in the local hand coordinate space before it became untracked. To reduce noise, the values for C_L and C_W are computed as a running average based on their values reported in the last 30 frames of sensor data (the running average is halted when tracking is lost). W_L, W_W, and W_D are weighting factors (values were set empirically based on experimentation) applied to the differences in finger lengths, widths, and distances respectively.

Amongst all the possible candidate fingers, the candidate chosen is the one that returns the minimum value for $h(C)$.

4.3.7 Gesture Detection

Our gesture interface is designed to allow users to intuitively control displays using hand gestures without actually touching the screen. All of the interactions in our interface are based on a combination of two basic components: air taps and dragging (or swiping). Finger airtaps are analogous to clicking mouse buttons or tapping a touch screen with your finger, but in mid-air. In our interface, this is used to select items. The air-tap was chosen because it can be recognized robustly via our system setup, and a great deal of previous work has demonstrated that most users find it exceptionally intuitive to learn and remember [91][30]. Dragging (or swiping) refers to moving the hand while at least one of the fingers is in the tap-down state (i.e., essentially performing the first half of an air tap, but halting before raising the finger). This is analogous to dragging file icons on a computer interface while having a mouse button down, or dragging objects across a touch screen display with the tip of one's finger, or swiping between pages displayed on a tablet computer.

In past works, Hidden Markov Models (HMMs) have proven to be effective at gesture recognition from appearance-based systems [98]. In our case, however, the actual finger positions in 3D space have values continuously reported. Thus, we instead use the current state of the fingers directly in a Finite State Machine (FSM) model removing the need for indirectly modelling hidden states. Likewise, unlike previous works, we found it was not necessary to filter the finger positions with a technique such as a Kalman filter [62] because the inherent sensor accuracy for tracked fingertips. The result is a computationally inexpensive direct analysis of the finger states using a FSM model for gesture recognition. Figure 4.4 shows a graph summarizing the state transitions for our prototype interface.

Tap Recognition

Using the local coordinate system defined earlier, it is tempting to recognize if an air tap is being performed by simply measuring if a finger has a local y value that is sufficiently below a threshold value. Experimentation shows, however, that the primary challenge in identifying air-taps lies in eliminating false positives that occur due to the tendency of all of a user's fingers having increasingly negative local y values as they naturally tend to drift towards a

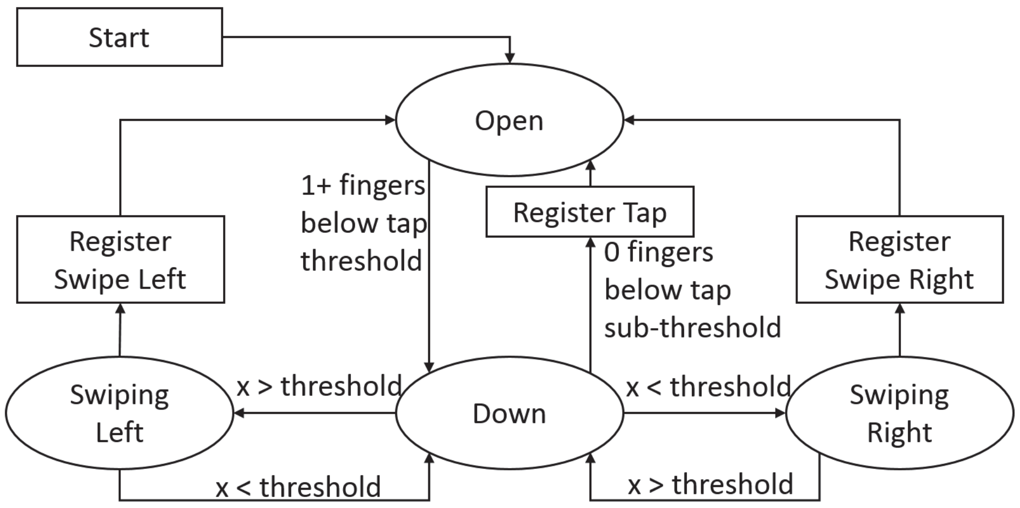


Figure 4.4: Overview of the state transitions for the proposed gesture interface.

more relaxed state over time as shown in Figure 4.5.

The threshold y value of the downward distance a finger must move in the local hand coordinate system in order to be classified as an air tap is therefore adaptive. It is based on Equation (4.6):

$$T_f = D_T(1 + W_T m_T) \tag{4.6}$$

Where T_f is the threshold y value (in millimetres) that the finger f must pass below in order for the finger to be recognized in the downward part of an air-tap, D_T is the default threshold value for an air tap assuming that all the other fingers are in the plane of the palm, and W_T is a constant weighting factor. m_T is the mean y position of all other fingers besides the finger f in our partial pose estimation model.

To prevent noisy toggling between the tapped and untapped states when a finger hovers near the threshold value, once a finger has been recognized as being in the tapped state, having passed below T_f, we require that it return through a different upper threshold value U_f nearer to the palm plane in order to be recognized as being in the untapped state. Where:

$$U_f = cT_f, \quad 0 < c < 1 \tag{4.7}$$

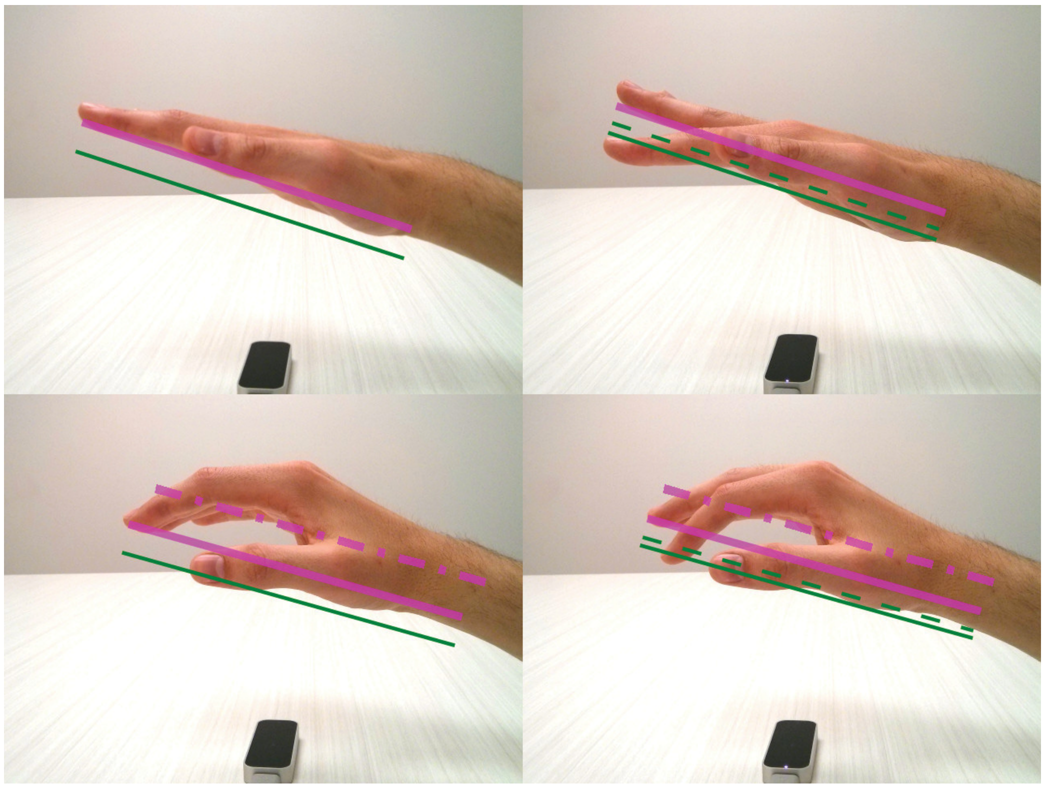


Figure 4.5: An overview of the tap recognition system, highlighting the need for dynamic thresholds. The dashed magenta lines indicate the palm plane as reported by the sensor, whereas the solid magenta line indicates the side view of the offset plane that is parrallel to the palm plane, but offset according to the average displacement of the fingertips relative to the palm plane. The solid green line indicates the threshold that must be exceeded to enter that tap Down state, whereas the dashed green line indicates the threshold that the finger must be above to return to the Open state. The top left image shows the ideal case where the user's fingertips are mostly lying the in the palm plane, and the tap is easy to recognize (top right). The bottom left shows a more realistic case where the user has taken a more relaxed hand posture over time and the tap gesture is more subtle (bottom right).

Figure 4.5 demonstrates the tap recognition in action corresponding to different situations.

Drag or Swipe Recognition

Drag gestures are recognized as hand movements that occur whilst one or more fingers are in the down position of an air tap. The primary novelty of our dragging gesture is that unlike a touch screen, where one drags virtual items according to the position of their fingertip, in our application items are tapped by the user's finger, but dragged according to the hand position. This is due to the lack of haptic feedback in mid-air gestures which makes it difficult to distinguish the movements of communicative tap gestures from being confused with the movements used to manipulate the position of a virtual object. Our approach of using the finger movements to indicate taps, while the hand position is used to manipulate the position of 3D virtual objects, is based on the similar experience one has with using mouse buttons for selections, and hand palm movements (i.e., mouse movements) to move objects on a computer screen.

4.4 User Study Setup

To evaluate the effectiveness of our approach in improving gesture recognition reliability, a user study was conducted in which users were instructed to complete 50 randomly assigned gestures as fast as possible using our interface. The three types of gestures users could be randomly asked to complete were: (i) swipe left, (ii) swipe right, and (iii) tap. Figure 4.6 shows how the experimental interface was set up.

Each test subject completed the experiment a total of six times where each separate trial alternated between gestures being recognized by our approach (B) and a baseline approach (A) of using the Leap Motion v1 API directly. To control the impact of repeated practice over time, half of all test subjects were randomly selected to complete the repeated experiments in the order A B, A B, A B, while the rest completed the experiment in the order B A, B A, B A. Each pair of experimental runs was grouped into a round, making three rounds in total.

During the experiment, users were given 8 seconds to complete a gesture during which time they could make as many attempts as they liked. Upon

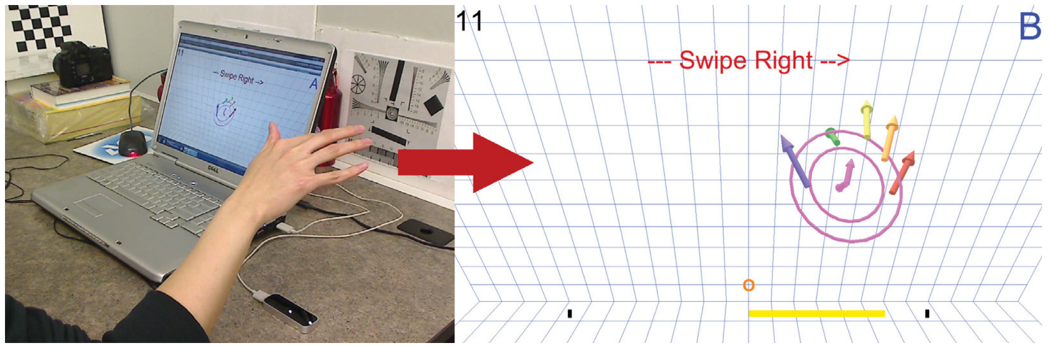


Figure 4.6: The experimental set up. On the experimental interface (right image) the values indicate: (top left) number of gestures completed thus far, (top center) current gesture to perform (top right) which recognition technique is currently being used (users not told what the letters mean), (bottom center) the orange circle appears when the hand is considered to be in the tap-down state, and the yellow progress bar appears during swipes to indicate to users how far to go to complete them (as indicated by the black goal markers).

successfully completing a gesture, the system flashes the word GOOD for 0.2 seconds and then immediately displays the next randomly chosen gesture to complete. To prevent user frustration in the event the system fails to recognize hand movements as the appropriate gesture, after 8 seconds the system flashes the phrase MOVING ON and continues on to the next gesture regardless. Therefore, the theoretical maximum time a user could take to complete the experiment was 410 seconds if none of their gestures were ever recognized. At the end of the experiment, the total time taken to complete all 50 gestures was reported, as well as the number of gestures (if any) that were not completed within the 8 second time window. The experiment was tested and designed such that the entire trial could be completed in approximately 60 seconds (with some practice) if there are no recognition errors. The experiment was completed by 25 participants (11 female, 14 male) between the ages of 22 and 41. One participant chose not to complete the third round of the experiment.

Given that the state-of-the-art pose estimation systems mentioned in the related works section [68] [47] are not applicable for this new class of sensors and the high sampling rate gesturing environment we are studying (i.e. the gestures we focus on are not readily recognizable at lower spatial resolutions or sampling rates), we therefore measure the improvement in gesture recognition rate of our approach in comparison to the baseline method of using the Leap Motion v1 API directly.

4.5 Results and Performance Analysis

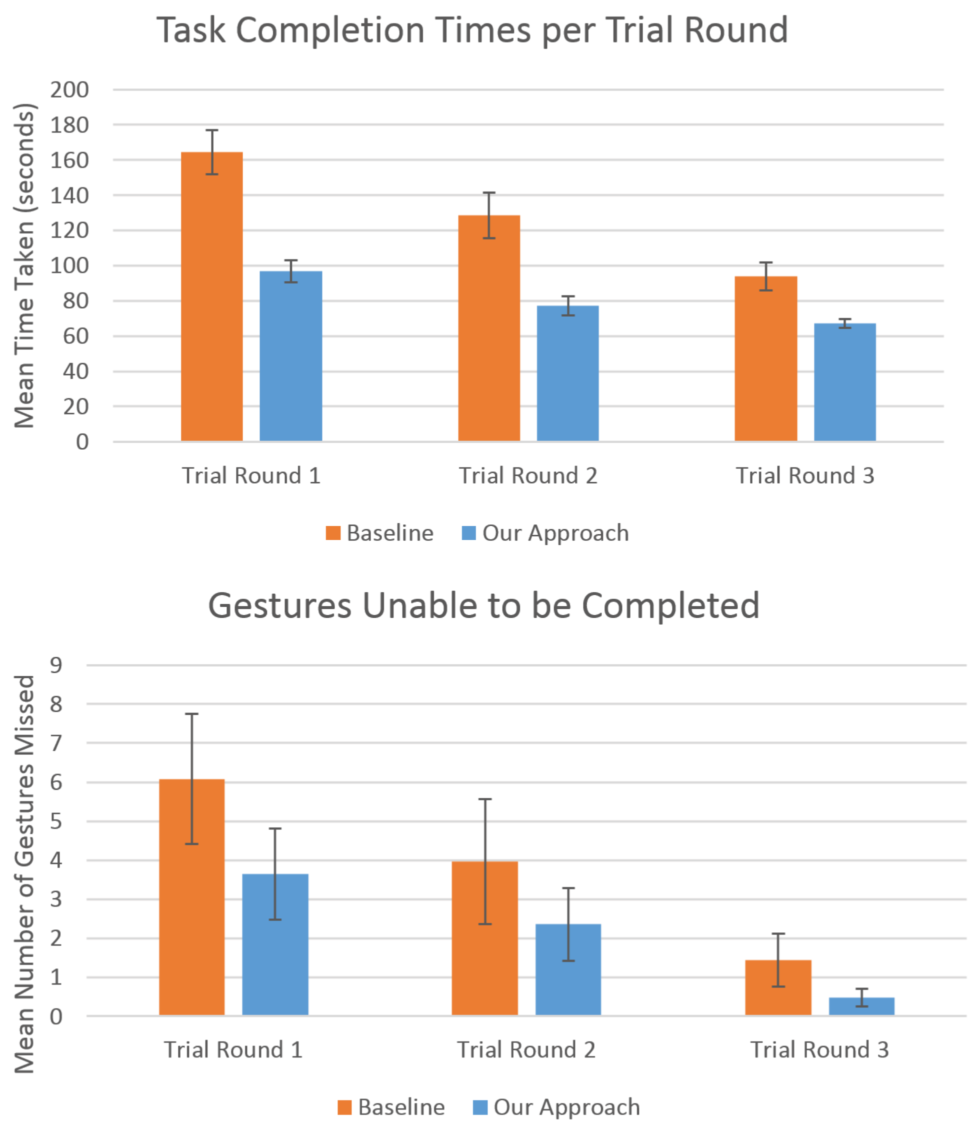


Figure 4.7: Results of user study.

The results for all six trials in the experiment are summarized in Figure 4.7.

A paired one-way mixed-design ANOVA revealed that for all three rounds, our approach significantly out-performed the baseline and reduced the total time taken to complete the 50 gestures ($\alpha = 0.05$, with P-values of 0.0001, 0.0001, and 0.0058 respectively). We also see that there is a significant amount of improvement in completion time with repeated uses of the system as users become more comfortable with the interface. The results are highly encouraging because they show a clear improvement in the gesture recognition rate when our method is applied, allowing users to complete tasks and interact through the interface more efficiently. In the study feedback, several users also reported that during Trial B (our approach) the interface generally felt more "reliable."

There are two main reasons why we chose to measure the total completion time for all 50 gestures instead of the proportion of gestures correctly identified compared to those that were missed. Firstly, attempting to define what constitutes a missed gesture is not always clear due to users commonly performing ambiguous movements that could subjectively be interpreted by a human as a swipe attempt, or simply re-positioning their hand to prepare for another gesture. Likewise, spontaneous flicks or twitches are difficult to discern as possibly being an intentional tap gesture. Secondly, measuring the total completion time of the task is a more objective assessment of what we are actually interested in, which is the overall communicative efficiency of using our interface in comparison to the baseline. We therefore define the gesture recognition rate in this paper as:

$$\text{Gesture Recognition Rate} = \frac{\text{Gestures Recognized}}{\text{Time}}$$

The average number of times the system was not able to recognize the appropriate gesture in the 8 second time window is also summarized in Figure 4.7.

4.5.1 Observations and Discussion

We observed that the mean number of completely missed gestures in each round was consistently higher for the baseline approach. Over time, it seemed users began to slowly learn what movements tended to cause interruptions in gesture recognition, and thus performed motions to avoid them. This came at the cost of slowing users down in baseline trials (A). Whereas in trials with our approach (B), users appeared to generally be slow and cautious at first, but then seemed to gradually learn that they could go faster and be less careful about their hand movements, and still have their gestures successfully recognized.

It was also generally noted that most of the failed gesture attempts with the baseline method were occurring during airtaps and swipes to the right. During airtaps, it was observed that at some point during the tap, the tapping finger would frequently point directly at the sensor as it passed over it, which caused it to become temporarily unrecognized as a finger, and interrupted the gesture. However, in our approach the persistent model successfully interpolated the finger position in these situations, and the gesture could be recognized properly. The issue of the tapping finger directly pointing at the sensor also occurred frequently during swipes as the finger passed over the sensor.

During swipes to the right, it was observed that users' fingers would frequently become crowded together towards the end of the swipe, causing a great deal of loss in tracking and frequently interrupting the gesture recognition. Once again, our persistent pose model was usually able to maintain the hand pose state and correctly recognize the gesture. This issue presented itself less often with swipes to the left, because the finger used to perform the swipe usually does not have fingers directly behind it from the point of view of the sensor.

4.6 Summary

In this work, we presented a novel partial pose estimation system that uses data input from a Leap Motion sensor v1 API to identify hand poses, and is capable of running at over 120 frames per second on a single computer. Our technique uses a customized kinematic constraint model and computationally efficient heuristics to achieve the partial pose estimation at high speeds. The results of our user study evaluation showed a significant improvement in gesture recognition rate, when our model was used in comparison to the baseline Leap Motion v1 API. While our markerless hand gesture detection approach can be deployed in general to execute interface commands, e.g., slider control and drag-and-drop, to make human-computer interaction more engaging, an important application is the control of sterile touchless medical displays via hand gestures.

The impact of our work is that our approach can be applied as a pre-processing step to almost any gesture recognition interface which uses the Leap Motion or similar sensor for gesture tracking because our model adds estimations for the positions of untracked fingers without incurring a large computational cost. As depth sensor technology evolves, occlusion continues to pose challenges for CV-based sensors. Our partial pose estimation based on kinematic constraints and human hand motion characteristics is a robust approach for dealing with this missing data in a manner that facilitates efficient gesture recognition.

Chapter 5

Touchfree Medical Displays

5.1 Introduction

Markerless, mid-air hand gestures tracked by computer vision techniques have become increasingly popular for human-computer interaction. Prior work has shown that their non-contact nature makes them especially well-suited for controlling visual displays at a distance [63]. They are also useful for controlling medical visualization interfaces where the lack of physical touch improves safety by avoiding possible contamination [1].

Some of the challenges when users are performing tasks that require them to hold a tool in their hand(s) such as a pen, computer stylus, laser pointer, or a medical instrument, are that it can interfere with finger tracking or make it awkward for the user to perform gestures in a way that can be recognized. For example, a clinician might be holding an ultrasound probe in one hand, and a biopsy needle in the other during a procedure, but would still like to control their real-time medical display. As another example, an artist might be using a computer drawing tablet with a stylus but would still prefer to make display interactions without switching to their non-dominant hand. Being forced to constantly put a tool down causes an undesirable interruption in workflow.

To address the challenges above, we propose a novel interface design for controlling displays that enables mid-air gestures from hands and also hand-held tools. In our approach, hand-held tools can automatically become gesture devices that the user may use to help control the display as shown in Figure 5.1. Previously, our proposed interface design would not be practical with low resolution commercial depth sensors, such as the Kinect [83], but is achievable in this work using the Leap Motion sensor's [95] higher resolution and sampling rate.

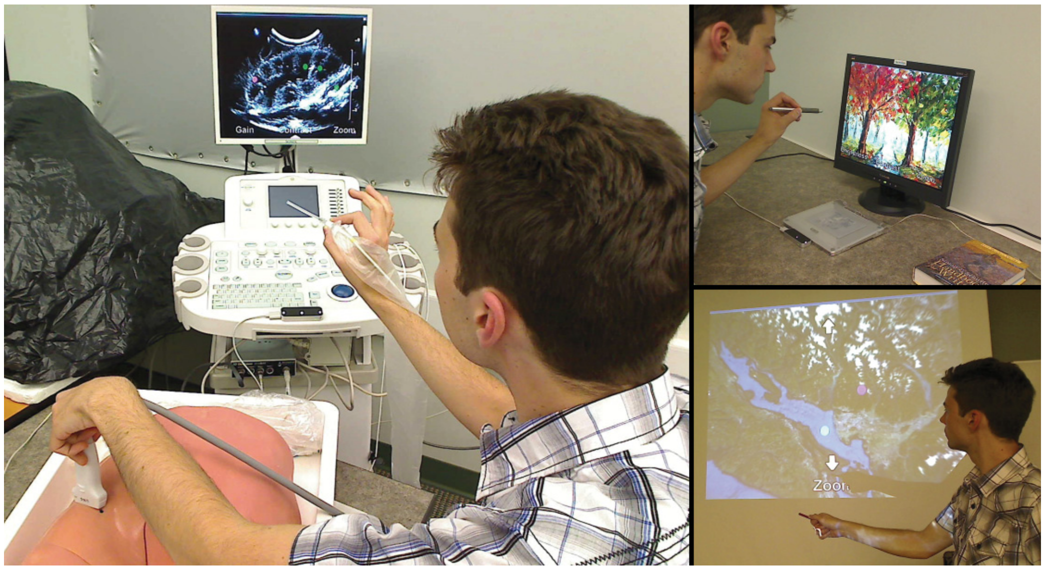


Figure 5.1: Several applications of our approach. Left: The user controls an ultrasound display without touching it, using a needle to gesture. Top Right: A computer stylus is lifted off its drawing table and used to adjust the display without needing to switch modes. Bottom Right: During a presentation, an ordinary laser pointer is used to interact with the display.

To summarize, the contributions of our work are:

1. An interface design that uses gestures specifically designed to be equally efficient with bare hands or hand-held tools.

2. Unique filtering rules introduced to help prevent unintentional gestures caused by hand movements while holding tools.

3. Developing an application for touch-free control of an interface coupled with an Ultrasound display.

4. The ability to operate in both bimanual, and unimanual modes, without the need for context switching, or the use of markers on hands or tools.

Finally, to evaluate the effectiveness of our approach, a prototype medical display interface was implemented that uses our interaction framework. A 3-stage user study conducted on our implementation revealed that users were capable of using either their hands or pointed tools to perform the gestures with no major loss in task performance times being observed. Applications of our work include touch-free interfaces for ultrasound devices, where a technician does not need to worry about having one hand free of gel for interacting with the device or a keyboard, or surgical environments where a dedicated operator (in addition to surgeons) is not needed for visualizing surgery related images.

5.2 Comparisons to Related Work

In 2003, the VisionWand [16] was proposed as a simple, low-cost tool for interacting with displays. The wand itself was simply a colored plastic rod that contained no electronics and was tracked in 3D by a pair of calibrated color cameras as the user performed 3D mid-air gestures. In 2007, Guo et al. [38] proposed a wand with similar features but with more degrees of freedom. However, in both of these approaches, only pre-calibrated marked tools were considered for interaction, and the tool gestures were considered in a completely separate context from hand gestures.

In 2005, Vogel et al. [91] investigated unimanual hand gestures for highly precise manipulations on a large high-resolution display. Their work identified that small quick finger gestures, such as tapping in mid-air (called an "air tap"), were highly intuitive for users and could be used for efficient and precise context switching. Similar findings were also confirmed by Fikkert el al. in 2010 [30] in their interactive map display. These small, precise finger gestures provide a good solution to the common issue of determining when to begin and end tracking of a user's hand for manipulation tasks, which is known as "gesture spotting" [94]. However, Vogel's system requires a costly, pre-calibrated motion capture rig and needs infrared reflectors placed on users' hands in order to achieve its precise tracking. Therefore, the system is designed to track only one hand at a time and does not consider the presence of unmarked hand-held tools. Techniques using data gloves or other sensors placed on the hand are similarly limited [32].

To overcome the limitations of markers, several recent approaches have made use of low-cost commercial depth sensors, such as the Microsoft Kinect, to interact with displays via hand gestures. Gallo et al. [33] proposed a Kinect-based interface in 2011 for visualizing medical images in a sterile surgery room requiring the user to be standing, and using both hands for interaction. Song et al. [84] proposed a novel 2-handed 3D gesture interface in 2012 for making precise manipulations using a handle-bar metaphor, and similarly, Schwaller et al. [81] used a 2-handed approach for panning and zooming displays. These techniques achieve their goals by using different, asymmetrical static hand poses on each hand for gesture spotting and context switching. This overcomes the inability of modern commercial depth sensors to precisely track dynamic finger gestures in 3D, because of noise and low resolution depth images. This approach also takes advantage of the fact that 2-handed gestures are often preferred by users for interacting with displays, and can allow for tasks to be completed quicker, as was demonstrated by Nancel et al. [63]. Users may not always have both hands free for interaction depending on the application domain, but it is ideal for the option to be available if possible. The weakness of this class of methods is the low accuracy which makes dynamic finger tracking impossible.

In contrast to the above approaches, our method: (i) is completely markerless while accommodating the use of tools, (ii) can track dynamic fingers and tools, thereby gestures like tapping, by exploiting the higher sensor precision and novel filtering rules.

5.3 Proposed Approach and Implementation

To test the efficacy of our design, we chose to apply it to the challenge of controlling an ultrasound machine's display parameters during procedures. A design was made in collaboration with our industrial partner who manufactured and provided the test machine. The parameters chosen were the ones identified as most frequently adjusted during procedures: Gain, Zoom, and Contrast. The gesture interpretation system was implemented on a dedicated low cost commodity laptop for mobility purposes (Intel x86 dual core processor at 2.2 GHz, 2 GB RAM), while display parameter adjustments were computed asynchronously and then sent to the display of an Sonix RP ultrasound machine. The sensor used was a low cost commercial Leap Motion controller which uses small IR (Infrared Red) LEDs and a pair of IR cameras to track unmarked objects with millimeter level precision at over 120 frames per second in an interaction volume of about 1 cubic meter. The Leap Motion v1 commercial API was used to track the the positions of finger tips, hands, and tool tips.

The API restricts the tools that can be tracked to have dimensions similar to that of a pen with a thickness between approximately 30mm and 3mm, which still covers a large range of hand-held tools (pens, markers, styluses, needles, probes, laser pointers, etc.) without any pre-calibration required. For larger tools, our interface allows users to still hold it in their palm and use free fingers to gesture, as the hand tracking is usually robust enough to remain stable in this state. Visual feedback lets users know what is being tracked as shown in Figure 5.2.

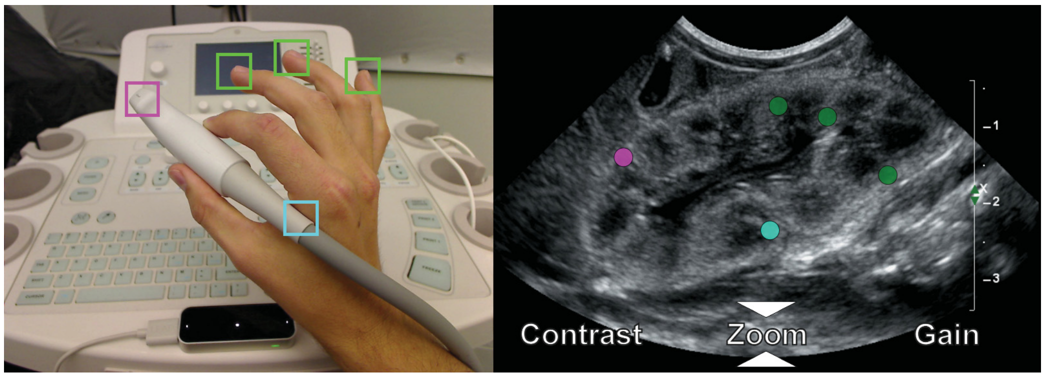


Figure 5.2: We use green indicators to identify tracked fingertips. Blue is used to indicate the palm position and magenta for tool tips, when recognized. Note that the coloured boxes on the left image correspond to the coloured circles on the right image, which is what is actually displayed to the user.

5.3.1 Gesture Interaction Design

To minimize cognitive load, our system uses a vocabulary of 3 communicative gestures for finger tips or tool tips, and 1 manipulative gesture for hands. A state transition diagram outlining the operation of the system is shown in Figure 5.3.

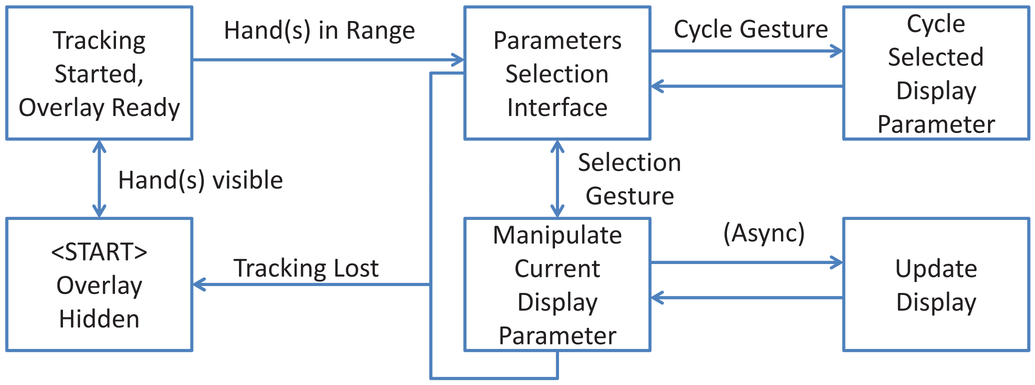

Tracking Started, Overlay Ready
Hand(s) in Range
Parameters Selection Interface
Cycle Gesture
Cycle Selected Display Parameter
Hand(s) visible
Selection Gesture
<START> Overlay Hidden
Tracking Lost
Manipulate Current Display Parameter
(Async)
Update Display

Figure 5.3: State transition diagram. Note that the Cycle Selected, and Update Display states loop back automatically, whereas the Selection Gesture is needed to toggle between the Parameters Selection and Manipulation states.

In its initial state, the interface overlay is hidden, and only the display is shown. As users bring their hands closer to the interaction volume, the system begins to track their hands, fingers and tools, and a partial overlay is shown as well as tracking indicators (represented as small colored circles) to provide visual feedback. When at least one hand is fully in the interaction volume, the parameter selection interface is shown and the currently highlighted display parameter is indicated. Drawing a small clockwise or counter-clockwise circle in the air (with a radius as small as 10 mm in size) will either cycle the currently highlighted parameter forward or backward through the list. To actually manipulate the currently highlighted parameter, the user performs a mid-air tap as if they were clicking a mouse. In parameter manipulation mode, the system uses the Y position (height) of the hand closest to the monitor to raise or lower the selected display value, which is updated on the display asynchronously in a separate thread to preserve system responsiveness. A tap performed by any finger tip or tool tip exits the adjustment mode and returns the user to the parameter selection state. Figure 5.4 shows gesturing possibilities in several hand configurations.

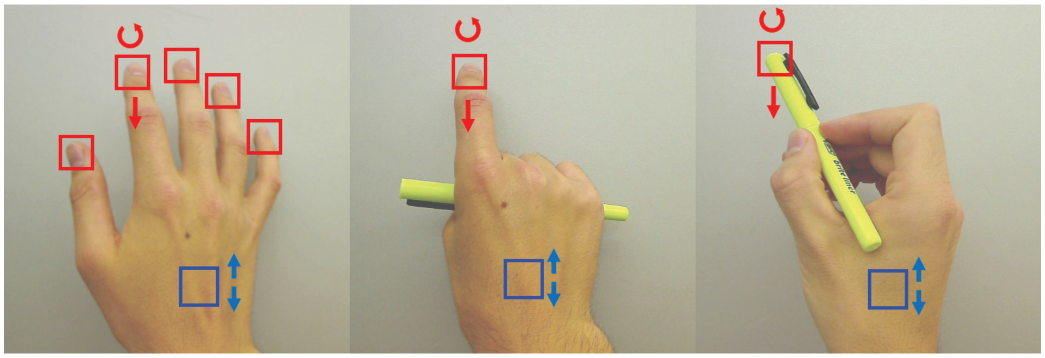


Figure 5.4: Left: Any finger may be used to perform mid-air tap gestures or circle gestures (shown in red). Middle: Only the index finger movements will be tracked for circle and tap gestures, the tool is hidden. Right: The tip of the felt pen is tracked for gestures while the remaining fingers are not, being closed. In all cases, the hand palm position (blue) is tracked independently, and may be moved for manipulative gestures.

Because the system does not restrict the tooltip or fingertip gestures to the hand being used for parameter manipulation (i.e., the hand closest to the display) highly precise adjustments can be made by using tapping gestures on one hand (with a tool or finger) while using the closer hand for parameter adjustment. For one-handed gesturing, the largely independent tracking of the palm and fingertips or tooltip tracking still allows for highly precise manipulations.

5.3.2 Rule Based Filtering of Gestures

A drawback of our approach described thus far is that unintentional fingertip or tooltip movements on one or both hands may be tracked and interpreted as input. Therefore, a set of custom rule-based filtering techniques were implemented to reduce unintentional gestures. For circle gestures, as soon as a successful circle is recognized (within radius requirements of about 10mm - 80mm and time < 1.0s), the time window of the gesture is established. Any other gestures with a time windows that overlaps a successful gesture are suppressed. This avoids the common issue of users accidentally moving multiple fingers in a similar manner while performing gestures. Quick gestures with a small time window, such as tapping, require a secondary time window of fixed size to be imposed to artificially suppress multiple fingers or tools unintentionally being involved in a tap.

A unique drawback of the chosen Leap sensor is that it occasionally causes partially occluded fingertips or tool-tips to have highly inaccurate pose estimations when partially occluded. These erroneously tracked 3D positions can sometimes be interpreted as a gesture, and thus must be recognized and suppressed.

5.4 Evaluation

To evaluate our interface, a user study was conducted in which users were required to perform a list of 6 arbitrarily chosen parameter adjustments (i.e. Gain, Zoom, and Contrast) on our ultrasound machine's display interface using only mid-air hand gestures. Participants were first instructed on how the interface worked, and were made to practice 10 successfully tracked circle gestures and 10 tapping gestures with their fingers before beginning. The participants repeated the exact same task list using their dominant hand in 3 different configurations:

Open Hand The hand is empty and completely open in a relaxed state. All fingers are tracked in this configuration and users are allowed to use any finger to perform the gestures.

Pointing Finger The empty hand is closed except for the index finger which is pointing forwards. This pose mimics holding a tool in the hand but not gesturing with it.

Tool in Hand For the experiment, a pencil was chosen as the tool to gesture with rather than a piece of medical equipment. Users were instructed to hold it as naturally as possible while leaving at least 30mm protruding at the tip.

The list of arbitrary parameter adjustment tasks was designed to take about 1 minute to complete. Upon completing the task in all 3 modes (Trial 1), the users were required to repeat the entire round of tasks two more times (Trials 2 and 3). The experiment was conducted with 12 participants (4 female, 8 male) between the ages of 20 and 55. After completing the experiment, users completed a 5-point Likert scale questionnaire.

5.4.1 Results and Discussion

The task completion times for all 3 trials were recorded and are shown in Figure 5.5. The results of the Questionnaire are summarized in Figure 5.6. The error bars in both graphs indicate the standard error. A two-way repeated

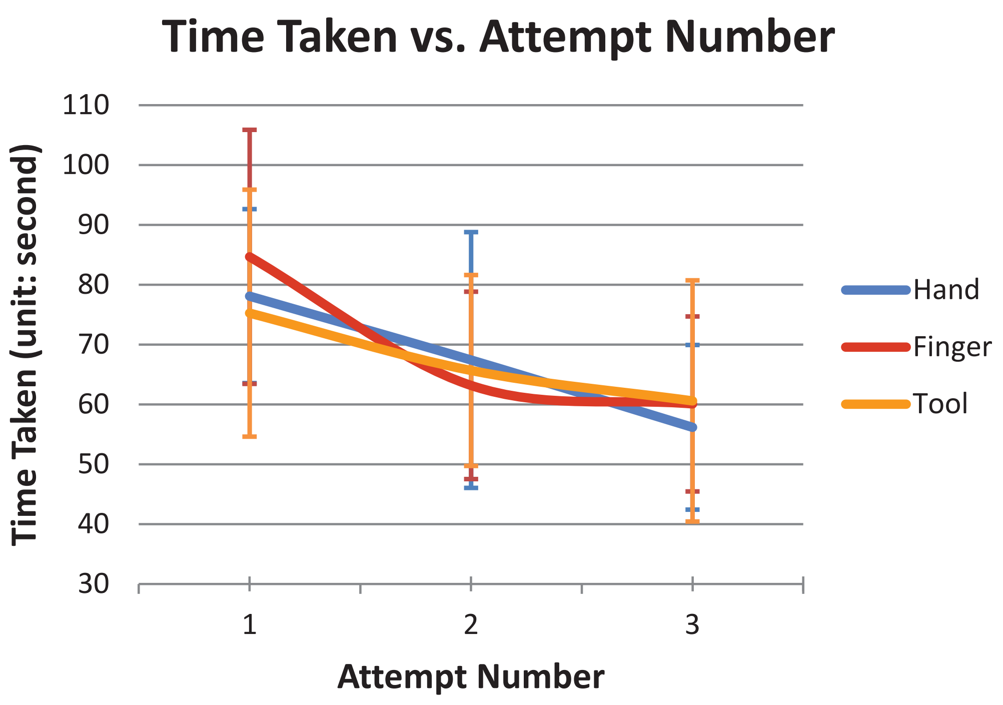


Figure 5.5: Comparison of Time taken vs. Attempt number.

measures ANOVA test examining the effect of trial round on completion time rejects the null hypothesis at the 5% significant level (p-value: $0.00004 < 0.05$) which indicates that the user completion time varies from one trial round to another. A two-way repeated ANOVA examining the effect of hand configuration on completion time does not reject the null hypothesis that the three interaction modes do not significantly differ from each other in this experiment with respect to user completion time at the 5% significance level (p-value: $0.85158 > 0.05$).

These results demonstrate that users can significantly improve their performance with practice over a small number repeated uses. The results also seem to suggest that there is no major difference in observed user performance with respect to the 3 different hand configuration modes. This is encouraging because it suggests that there is no exceptional loss in performance when gesturing with a tool instead of putting it down. However, due to the high variance measured, more user studies need to be conducted in the future to test for more subtle differences.

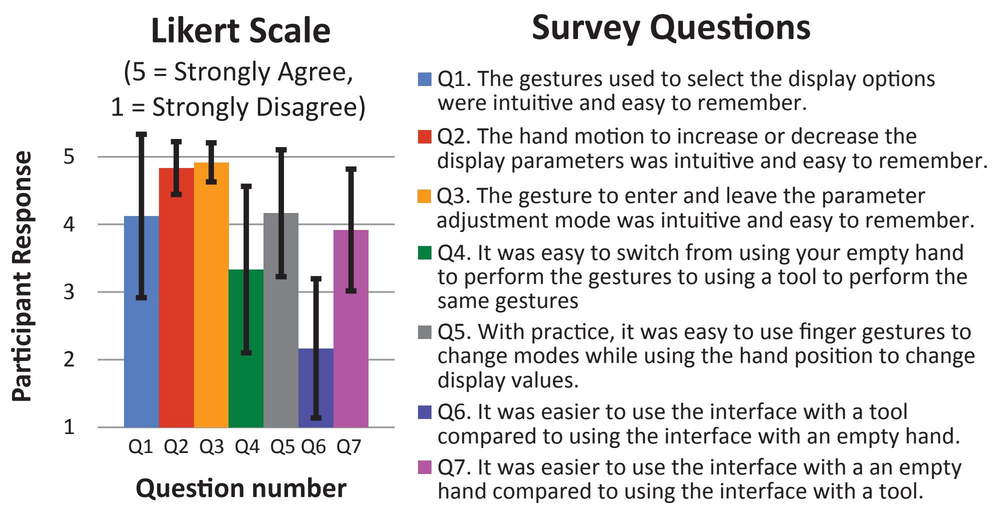


Figure 5.6: Results of Questionnaire.

The questionnaire revealed that most users generally found the gesture interface design to be intuitive and easy to remember, although they expressed a preference for interacting with an empty hand rather than a tool, though their performance between the two were similar. However, this peculiar result is consistent with results from [63] who found a similar pattern, and suggests that user opinion may change over time as they become more familiar with using tools for gesturing.

5.5 Summary

In this work we presented a novel, markerless gesture interface using a leap motion sensor and robust filtering rules. Our approach seamlessly integrates gesturing with hands and unmarked tools in a single interface for precise control of display parameters via mid-air gestures. Application areas include controlling medical displays, or more streamlined interaction when using devices like computer styluses and tablets. The results of our user study suggested that users can quickly make a significant improvement on their performance with practice, and no exceptional loss in performance between different hand configurations was observed.

Chapter 6

Framework for Adaptive Training in Complex Interfaces

6.1 Introduction

In this work we explore a technique to adaptively train users on the use of a challenging interface. As users begin to use a difficult interface (such a gesture-controlled display interface or a medical device such as a joystick-controlled electric wheelchair) their skill and proficiency can be expected to change over time. A key challenge during the training steps is ensuring that the difficultly of the training tasks is close to the user's proficiency level. If the task is too easy, then users are learning at a sub-optimal rate. If the task is too difficult, users may become frustrated and quit. To address this, we propose a novel, flexible, adaptive framework based on Bayesian networks to allow a system to probabilistically measure a user's performance level in several skill areas at once and intelligently adapt the interface training tasks appropriately.

In order to demonstrate our model in a useful real-world environment, we applied it to the task of helping rehabilitation patients learn to use an electric power wheelchair, where the device is controlled through careful hand movements on a sensitive joystick. To enhance the immersion of our approach, the hand movements control an engaging virtual reality power wheelchair simulation. The effectiveness of our approach was supported by the results of a real-world user evaluation and feedback from our rehabilitation specialist collaborators.

Motivation for Virtual Rehabilitation Interfaces

Typically, patients being trained on the use of the electric power wheelchairs must have their wheelchairs and training regimes highly customized to their specific needs and capabilities. This presents a problem because the number of rehabilitation specialists and power wheelchairs available on hand at any time is limited. As a result, patients are quite limited in terms of the amount of time they may receive training from a specialist.

However, virtual reality training offers potential solutions to many challenges associated with rehabilitation medicine and power wheelchair training. For example, previous research has shown that skills acquired in virtual reality wheelchair training can transfer effectively to real-world wheelchair use [29] [20] [60] and virtual reality has also been shown to improve training results through the use of virtual reality rehabilitation games [46] [50] [101] [15] that engage and motivate patients to complete longer, more effective training sessions [42].

Given these benefits, a virtual reality power wheelchair training system was designed and implemented with collaboration and feedback from a group of rehabilitation specialists and clinicians with expertise in a wide range of rehabilitation medicine including power wheelchair training. Our virtual reality training system is unique in that it was designed to be low-overhead, portable, and focused on indoor training environments that are interactive in a reasonably realistic way. The system offers standard training modes and interactive virtual reality games for increased patient engagement. A particularly unique feature in comparison to similar approaches is that the system provides a framework to allow clinicians to design, build, and customize interactive indoor rehabilitation training environments that can dynamically respond to user actions in a meaningful way.

6.2 Related Work

Several approaches to virtual wheelchair training have been made in recent years using a variety of proposed techniques. At one end of the scale, Niniss et al. [65] proposed an immersive virtual wheelchair training system whereby an actual power wheelchair rests on a raised hydraulic platform that can tilt in response to the user's movements in the virtual environment which is displayed on a 110 degree panoramic screen. Despite its highly immersive features, one of the goals of virtual reality rehabilitation is to increase the amount of training time per patient in a realistic rehabilitation setting where resources are not unlimited. A system such as the one implemented by Niniss et al. [65] is not ideal given that its apparatus cost is higher than that of an actual power wheelchair and availability will still be limited to one patient at a time. Also, customizing such a system on a per-patient basis still remains time consuming.

At the other end of the scale, there are systems such as the one developed by Spaeth et al. [85] which simply uses an ordinary projector and a power wheelchair fitted with a special head-brace to track the head position. The patient controls virtual racing cars which are shown on the projector as a 2D top-down visualization of a race track. Although systems such as these satisfy the goal of creating a low-cost automated supplement to real-world power wheelchair training, as the authors point out, the simplistic non-immersive game-like interaction is more focused on measurement of patient responses over time, and the skills learned are not readily applicable to use of powered wheelchairs in the real world [85].

Commercial desktop training simulators such as Wheelsim[1] offer reason-

[1]http://www.ottobock.com/

able immersion and are low-cost, but typically lack the possibility for specialists to adjust environments to patients' needs. They also do not provide realistic interactions between objects in the simulation (for example, in Wheelsim, collisions simply result the virtual wheelchair stopping instantaneously). This is an issue given that previous research has shown that realistic physical interactions between objects in virtual reality rehabilitation simulations are important for ensuring that skills learned are more effectively transferred to the real world [60].

Adaptive Training Interfaces

By automating the task of constantly adjusting patient exercies to provide the optimal challenge level, specialists can have additional time to work with more patients. Several previously implemented virtual reality rehabilitation systems use a single metric, for example, how long it took to complete a task or what the range of motion was, and then slightly increment the difficulty level accordingly [42]. Other systems make use of a game-based levelling system in their virtual rehabilitation training. As patients successfully complete levels, they automatically move on to more challenging levels [50]. While such systems are simple to implement and successfully provide some degree of automatic adaption, they are not able to measure nor address the enormous range of disabilities and residual capabilities that patients may have, and they do not make full use of the background knowledge and skills of the rehabilitation specialists. While previous work by Ma et. al. [60] has proposed adaptive virtual reality games for rehabilitation that attempt to measure a much wider range of patient skills and adapt according to each patient's unique skill set, the skill estimation and adaption rules are hard-coded into the games, thereby not enabling rehabilitation specialists to apply their expertise and specialized domain knowledge to modify them as needed.

To address this challenge, we propose a novel, flexible framework based on Bayesian networks. Although previous research has proposed Bayesian networks as a way to measure student performance in online web-based multimedia educational games [19], to the best of our knowledge, this is the first attempt to adapt this concept to the continuous input of hand movements to train users to control an interface.

6.3 System Design

6.3.1 Customizable Virtual Reality Environment

To handle different computer hardware configurations, the proposed virtual reality system was designed as a stand-alone open-source 3D application capable of running on any modern desktop or laptop using any standard operating system (Mac, Linux, or Windows). The system requires no specialized input hardware, though the main mode of input is an analog joystick. The Open-Source Graphics Rendering Engine (OGRE[2]) was selected as the 3D graphics engine due to its substantial feature set, open-source nature, and the successful use in previous rehabilitation systems [60].

To encourage the transfer of skills learned in the virtual reality environment to the real world, interactions between the virtual wheelchair and other objects in the simulation were realistically modelled using the Newton Game Dynamics[3] physics engine. Newton was chosen due to being open source and its integration with OGRE through the OGRENewt[4] API. Furthermore, previous research has concluded that game engines could play an important role in developing realistic, effective rehabilitation systems [36].

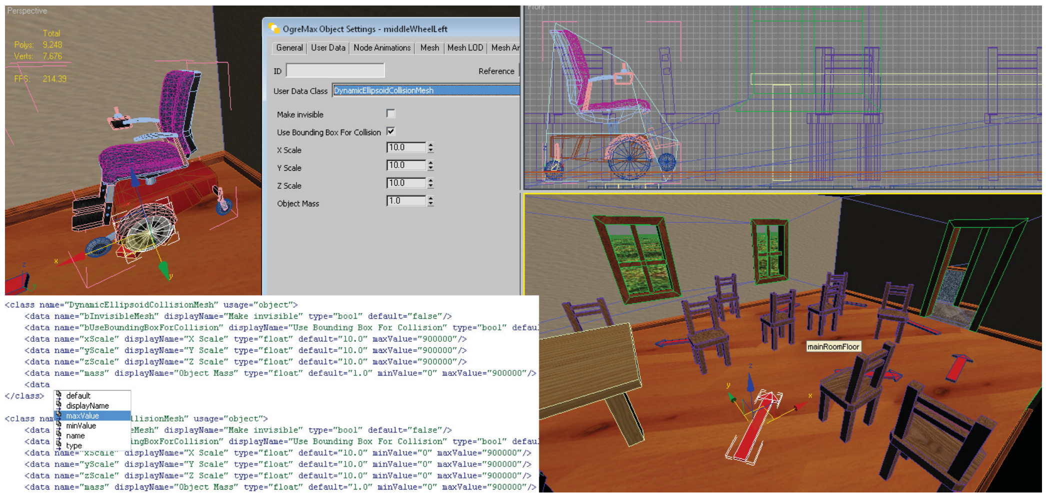
A key challenge posed by the rehabilitation collaborators involved in the

[2]http://www.ogre3d.org/

[3]http://newtondynamics.com/

[4]http://www.ogre3d.org/tikiwiki/OgreNewt

project was the desire to allow clinicians to easily design and modify virtual environments for their patients so that they could practice manoeuvres such as navigating around the patient's house. In order to achieve this, an XML scene node based framework was designed and implemented as a 3D-modelling plug-in tool (see Figure 6.1). Using this tool, clinicians can change the layout of training rooms, adjust the position, scale, and rotation of objects, and even assign physical properties such as object mass. The resulting customized environments are stored in XML format which can also be modified directly as shown in Figure 6.1. To the best of our knowledge, this ability to provide clinicians with a flexible, extensible means by which to customize their virtual reality training environments is a feature not provided by any other virtual reality power wheelchair training environment.



OgreMax Object Settings - middleWheelLeft

General | User Data | Node Animations | Mesh | Mesh LOD | Mesh Ar

ID:
Reference

User Data Class: DynamicEllipsoidCollisionMesh

Make invisible: ☐
Use Bounding Box For Collision: ☑
X Scale: 10.0
Y Scale: 10.0
Z Scale: 10.0
Object Mass: 1.0

```
<class name="DynamicEllipsoidCollisionMesh" usage="object">
    <data name="bInvisibleMesh" displayName="Make invisible" type="bool" default="false"/>
    <data name="bUseBoundingBoxForCollision" displayName="Use Bounding Box For Collision" type="bool" defau
    <data name="xScale" displayName="X Scale" type="float" default="10.0" maxValue="900000"/>
    <data name="yScale" displayName="Y Scale" type="float" default="10.0" maxValue="900000"/>
    <data name="zScale" displayName="Z Scale" type="float" default="10.0" maxValue="900000"/>
    <data name="mass" displayName="Object Mass" type="float" default="1.0" minValue="0" maxValue="900000"/>
    <data
</class>

<class nam                llisionMesh" usage="object">
    <data                 eMesh" displayName="Make invisible" type="bool" default="false"/>
    <data                 ingBoxForCollision" displayName="Use Bounding Box For Collision" type="bool" defau
    <data name="xScale" displayName="X Scale" type="float" default="10.0" minValue="0" maxValue="900000"/>
    <data name="yScale" displayName="Y Scale" type="float" default="10.0" minValue="0" maxValue="900000"/>
    <data name="zScale" displayName="Z Scale" type="float" default="10.0" minValue="0" maxValue="900000"/>
    <data name="mass" displayName="Object Mass" type="float" default="1.0" minValue="0" maxValue="900000"/>
```

Figure 6.1: Interface for customization of the virtual environment.

The primary advantages of the design chosen are that the issues of cost and power wheelchair availability are both addressed. The computer hardware and software needed to run the system are low-cost and readily available, and can even be used for other purposes when not used for virtual reality training. In light of research demonstrating the ability of interactive games to encourage patient engagement and increase the length and effectiveness of rehabilitation sessions [46] [50] [101] [15], the system is also capable of training through interactive games which are implemented in the same way as standard training environments, but with the addition of special goals. An example of one such simple rehabilitation game called Virtual Reality Wheelchair Soccer was implemented (see Figure 6.2) where the goal is to carefully navigate the wheelchair so as to push the ball past two goal posts in a certain amount of time.

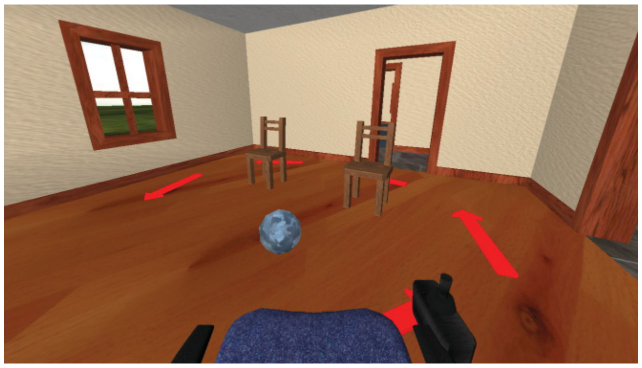
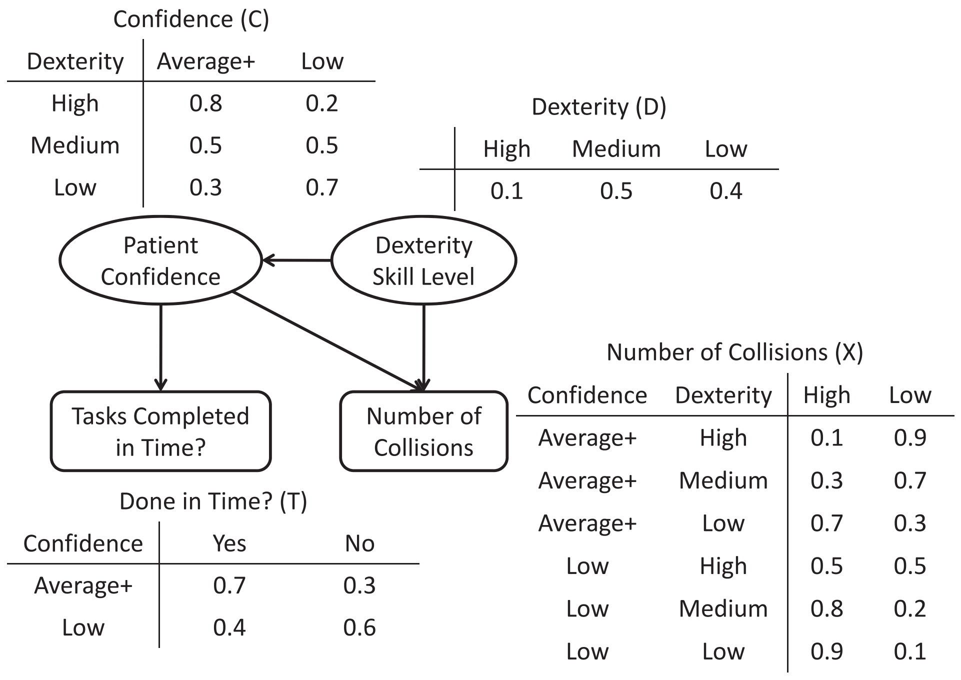


Figure 6.2: A simple game of virtual reality soccer where the objective is to carefully maneuver the virtual wheelchair and use it to push the ball past the two goalposts.

6.3.2 Interface Adaption

The second challenge of the implemented framework was to have the system automatically assess a complex range of patient skills, many of which are not independent of each other, and then have the virtual rehabilitation system automatically and intelligently adapt the training simulation accordingly. The designed approach uses Bayesian networks to attempt to determine patient skill levels in a variety of skill areas based on measurable results collected by completing rehabilitation tasks.

Consider the simple Bayesian network shown in Figure 6.3. In this example, the rehabilitation specialist is interested in determining the patient's dexterity level based on how long a patient takes to drive through a virtual reality obstacle course, and the number of collisions that occur along the way.



Confidence (C)

Dexterity	Average+	Low
High	0.8	0.2
Medium	0.5	0.5
Low	0.3	0.7

Dexterity (D)

High	Medium	Low
0.1	0.5	0.4

Number of Collisions (X)

Confidence	Dexterity	High	Low
Average+	High	0.1	0.9
Average+	Medium	0.3	0.7
Average+	Low	0.7	0.3
Low	High	0.5	0.5
Low	Medium	0.8	0.2
Low	Low	0.9	0.1

Done in Time? (T)

Confidence	Yes	No
Average+	0.7	0.3
Low	0.4	0.6

Figure 6.3: A sample Bayesian network topology.

Suppose that a patient's confidence level is also known to impact performance and that confidence is affected by the patient's dexterity level as well. The network layout in Figure 6.3 shows how a rehabilitation specialist might model this. As shown in the figure, the latent variable "Dexterity" increases the measured number of collisions when it is low, but low values for dexterity also increase the probability of a patient having low confidence. Although confidence is another latent variable that cannot be directly measured, we can see that it has an effect on both the length of time it takes for a patient to complete the exercise, and also the number of collisions that will occur along the way.

Once the network topology is defined, specialists can infer the value of the hidden latent variables such as "Dexterity" and "Confidence" based on the values of the measured variables. For example, in order to determine the probability that the patient has low dexterity given that they did not finish the exercise on time, and that the number of collisions was high, we could compute this as shown below.

$$\boldsymbol{P(D = Low|T = No, X = High) = \frac{P(D = Low, T = No, X = High)}{P(T = No, X = High)}}$$

$$\boldsymbol{= \frac{\sum_{C \in \{Average+, Low\}} P(C, D = Low, T = No, X = High)}{\sum_{C,D \in \{High, Medium, Low\}} P(C, D, T = No, X = High)} \approx 0.539}$$

Although the example shown uses a simple network with only two measured variables and two latent variables, this framework can be easily scaled to include a large number of latent and measured variables with complex interactions between them. Additional measured variables might include the optimality of the path a patient took through an environment, the ratio of collisions that occurred with objects out-of-sight compared to objects that were in plain view, or the number of collisions that occur while turning rather than simply driving straight, etc. Additional latent variables could include the ability of patients to judge distances, the patient's spatial awareness in regards to objects that are behind them, or the patient's spatial awareness on one side of their field of view compared to another in the case of brain injury, etc.

Likewise, the values of the variables in the network need not have only two or three possible values. Clinicians can define as many values as necessary for nodes in the Bayesian network, and express continuous variables as meaningful discrete ranges with as much granularity as necessary. Based on the inferred values of the latent variables, clinicians define which virtual game or training environment should automatically be loaded next to offer the appropriate difficultly level and type of training needed to continue to improve the patient's skills.

6.4 System Evaluation

6.4.1 Method

In order to help determine the effectiveness of the implemented system, a small-scale user evaluation was performed. The objective of the experiment was to determine if the virtual reality training system implemented was effective in teaching test participants to navigate around obstacles in an indoor environment any faster than those that did not receive the virtual training. A group of 8 graduate students between the ages of 22 and 28, both male and female, who had never used a power wheelchair before were randomly split into two evenly-sized test groups: the control group and the experimental group. A simple indoor power wheelchair obstacle course was then set up using office chairs and tables that was designed to require approximately 90 seconds to complete without collisions.

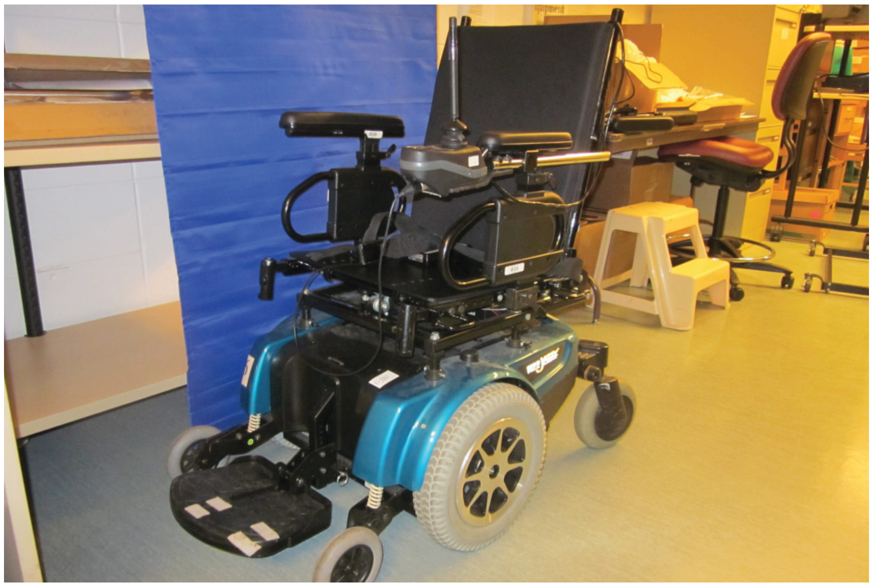


Figure 6.4: Real-world power wheelchair used in experimental setup.

Participants in both the control and experimental groups were tasked with completing the course as fast as possible using a “Jazzy 1122” mid-wheel drive type electric power wheelchair with a standard joystick as its control device as shown in Figure 6.4. Participants in the experimental group first completed a virtual obstacle course in the proposed virtual reality training environment prior to their real-world run, whereas individuals in the control group received no virtual training beforehand. The time taken by each test participant to complete the real-world obstacle course was then recorded.

6.4.2 Results

Participants who used the proposed virtual reality wheelchair training system completed the real world obstacle course on average faster, with a mean time of 81.5 seconds for the experimental group compared to 104.5 seconds for the control group. While the results seem to suggest that the proposed system is effective in training individuals on the use of power wheelchairs, due to the high level of variance between individual completion times (i.e. a standard deviation of 26.51 for the response group, and 43.16 for the control) and the fact that the system was tested on non-disabled individuals, more research is needed before more conclusive results can be drawn.

The completed prototype system was also demonstrated to a group of rehabilitation specialists and clinicians present at the collaborating rehabilitation hospital. Feedback was positive to the point that plans are now currently in motion to begin integration of the proposed system into a part of the rehabilitation training practices used at the hospital.

6.4.3 Discussion

Although our specific experiment was focused on controlling powered electric wheelchairs, our framework is applicable to a much wider range of applications. As an example, a common rehabilitation task in the case of stroke patients is to perform hand and arm exercises to restore the full range of hand motion and dexterity that may have been partially lost. By using our hand-tracking frame-work described in Chapters 3 and 4 clinicians can quantitatively measure several different characteristics of the user's hand motions. Characteristics that could be measured include variability in hand motions, movement speed, or accuracy in performing specific motions. These measured characteristics can subsequently be used as observed variables in the framework proposed in this chapter. One can then use these observations to attempt to determine hidden characteristics of a user's performance such as dexterity, confidence, and level of control. Clincians can then use the predicted values of these hidden variables to dynamically adjust the rehabilitation training exercises appropriately.

For increased hand pose estimation accuracy, clinicians can apply the techniques from Chapter 4 (in the case of a single sensor setup) or alternatively, use the techniques from Chapter 3 (for systems with multiple sensors). In cases where the rehabilitation tasks require the manipulation of tools, the techniques outlined in Chapter 5 are directly applicable.

6.5 Summary

In this work, we demonstrated a means by which to adaptively train users on the control of a challenging user interface. This is accomplished by a proposed novel Bayesian network-based adaptive training framework that uses measured variables from a completed training simulation in order to probabilistically determine unknown user skill levels in a variety of key areas. We then demonstrated how our model could be applied to the analog hand-joystick interface of a novel virtual reality rehabilitation training system. The same model is also completely applicable to other interfaces that use continuous motions such as hand or full body gesture interfaces.

Chapter 7

Conclusion and Future Directions

In this work, we explored several techniques for improving the efficiency and robustness of interaction with hand gesture interfaces. The techniques involved not only improving the underlying hand pose estimation and gesture recognition, but also techniques for making the interfaces more adaptive to user interactions.

We first explored a novel real-time approach to increase hand pose estimation accuracy through an intelligent analysis of estimations from multiple state-of-the-art sensors at extremely high sampling rates. Experimental results demonstrate that our approach can reduce the pose estimation error by over 31% (in comparison to using a single sensor) when applied to gestures commonly used in display interfaces (tap, pinch gestures, etc.). The key contribution of our work is that it is the first to demonstrate the viability of integrating different pose estimations in this manner while in the absence of any depth or color image data. Our technique can also be expected to scale better than techniques that use image/depth data because of reduced computational complexity and data bandwidth requirements.

Next we demonstrated an approach that uses kinematic constraints to reconstruct a partial pose estimation and improve real-time gesture recognition from a single sensor system without image data. A user study was conducted which showed that our approach significantly improved the gesture recognition rate while still producing pose estimations at over 120 times per second. The key contribution of this work is that our approach can be applied as a pre-processing step to improve the gesture recognition rate in any sensor systems that produce partial pose estimations in a similar manner.

We then proposed a novel gesture interface for controlling medical displays that was specifically designed to adapt appropriately to whether or not the user was holding a small handheld instrument. Our experimental results showed that there was no observable loss in user performance when users transitioned between different hand configurations, including configurations where the users were holding a small instrument. This framework allows for more streamlined interface interactions because it does not require users to remove tools from their hands when switching tasks.

Finally, we examine techniques to make the training of challenging interfaces more efficient. This is accomplished through a Bayesian network model that measures various hidden parameters of a user's skill set based on measurable properties of their hand movements. We show how our model is suitable to the training of an electric wheelchair interface in a virtual reality rehabilitation environment.

7.1 Future Directions

The techniques presented in this document have thus far primarily focused on building models for hand motions, however, with some modifications most of the techniques explored should also be effective when applied to full body human pose tracking and gesture recognition. For example, it should be possible to use a multi-sensor set-up for full human skeletal pose recognition that makes use of the same general approach from Chapter 3 but with a modified feature vector that uses the orientation of the human body to determine which sensor is most likely to give the best data on limb poses. Additional future directions for work our include investigating the effectiveness of our approaches when used with larger arrays of sensors to help increase the size of the interaction space.

More generally, there are several interesting directions for the field of human pose recognition in the future. With new types of low-cost, highly accurate depth sensors now appearing commercial markets, there is a great deal of potential work to be done. One can likely expect a continuation of the current trend whereby different sensors are likely to be more or less optimized for specific applications. Future research challenges will likely involve combing data streams from multiple sensors each with different resolutions, noise characteristics, sampling rates, and likely even data content (e.g. colour images vs. depth images).

An additional trend to consider is the steady shift of low-cost hardware sensors to mobile devices. One can expect that within the next few years, many mobile devices may carry depth/image sensors which may be used for human hand gesture recognition in order to expand the interaction space beyond the confines of a small touch-screen. This is likely to introduce a new suite of research challenges due to the tight processing power constraints imposed by mobile hardware. Furthermore, additional challenges will likely stem from a wider range of uncontrolled illumination and backgrounds, and additional motion caused by unstable movement of the mobile device which may make it difficult to isolate hand motions. Further work could focus on integrating data from multiple mobile devices arranged in an ad-hoc configuration. Besides the challenges involved in recognizing human hand poses for mobile devices, more investigation will likely be necessary to determine which gestures are best-suited for interacting with a small, hand-held mobile device.

Finally, as hardware and software for recognizing gestures in new application domains evolves, it will become increasingly important to the research field to provide open datasets to allow for more direct comparisons between approaches that address similar challenges.

Bibliography

[1] Invisible touchcontrol of a dicom viewer with finger gestures using the kinect depth camera. *Journal of Forensic Radiology and Imaging*, 1(1):10 – 14, 2013.

[2] K. Abe, H. Saito, and S. Ozawa. Virtual 3-d interface system via hand motion recognition from two cameras. *Trans. Sys. Man Cyber. Part A*, 32(4):536–540, July 2002.

[3] Mark Shelley Qi Jiang Abhishek Seth, Shana S. Smith. A low cost virtual reality human computer interface for cad model manipulation. *Engineering Design Graphics Journal*, 69(2), 2005.

[4] S. Ahmad. A usable real-time 3d hand tracker. In *Conference Record of the Twenty-Eighth Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1257–1261 vol.2, Oct 1994.

[5] Jonathan Alon, Vassilis Athitsos, Quan Yuan, and Stan Sclaroff. Simultaneous localization and recognition of dynamic hand gestures. In *Proceedings of the IEEE Workshop on Motion and Video Computing (WACV/MOTION'05) - Volume 2 - Volume 02*, WACV-MOTION '05, pages 254–260, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Xiang Bai and Longin Jan Latecki. Path similarity skeleton graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(7):1282–1292, July 2008.

[7] Tomasz P. Bednarz, Con Caris, Jeremy Thompson, Chris Wesner, and Mark Dunn. Human-computer interaction experiments immersive virtual reality applications for the mining industry. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications*, AINA '10, pages 1323–1327, Washington, DC, USA, 2010. IEEE Computer Society.

[8] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, April 2002.

[9] Hrvoje Benko and Andrew D. Wilson. Multi-point interactions with immersive omnidirectional visualizations in a dome. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 19–28, New York, NY, USA, 2010. ACM.

[10] Paul J. Besl and Neil D. McKay. Method for registration of 3-d shapes, 1992.

[11] Mark Billinghurst. Put that where? voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 32(4):60–63, November 1998.

[12] M. J. Black and A. D. Jepson. Recognizing temporal trajectories using the condensation algorithm. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, FG '98, pages 16–, Washington, DC, USA, 1998. IEEE Computer Society.

[13] Richard A. Bolt. Put-that-there: Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 14(3):262–270, July 1980.

[14] Matthieu Bray, Esther Koller-Meier, and Luc Van Gool. Smart particle filtering for 3d hand tracking. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, FGR' 04, pages 675–680, Washington, DC, USA, 2004. IEEE Computer Society.

[15] Bas Brederode, Panos Markopoulos, Mathieu Gielen, Arnold Vermeeren, and Huib de Ridder. powerball: the design of a novel mixed-reality game for children with mixed abilities. In *Proceedings of the 2005 conference on Interaction design and children*, IDC '05, pages 32–39, New York, NY, USA, 2005. ACM.

[16] Xiang Cao and Ravin Balakrishnan. Visionwand: interaction techniques for large displays using a passive wand tracked in 3d. In *UIST*, pages 173–182, 2003.

[17] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21(8):745 – 758, 2003.

[18] Jian Chen. A hybrid direct visual editing method for architectural massing study in virtual environments. In Xiangyu Wang and JerryJen-Hung Tsai, editors, *Collaborative Design in Virtual Environments*, volume 48 of *Intelligent Systems, Control and Automation: Science and Engineering*, pages 131–140. Springer Netherlands, 2011.

[19] I. Cheng, S. Rodriguez, and A. Basu. Multimedia and games incorporating student modeling for education. In *Technology for Education, 2009. T4E '09. International Workshop on*, pages 91 –94, aug. 2009.

[20] Rory A. Cooper, Donald M. Spaeth, Daniel K. Jones, Michael L. Boninger, Shirley G. Fitzgerald, and Songfeng Guo. Comparison of virtual and real electric powered wheelchair driving using a position sensing joystick and an isometric joystick. *Medical Engineering & Physics*, 24(10):703 – 708, 2002.

[21] Andrea Corradini. Dynamic time warping for off-line recognition of a small gesture vocabulary. In *Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01)*, RATFG-RTS '01, pages 82–, Washington, DC, USA, 2001. IEEE Computer Society.

[22] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J.G. Taylor. Emotion recognition in human-computer interaction. *Signal Processing Magazine, IEEE*, 18(1):32–80, Jan 2001.

[23] R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, FG '98, pages 416–, Washington, DC, USA, 1998. IEEE Computer Society.

[24] Barney Dalgarno and Mark J. W. Lee. What are the learning affordances of 3-d virtual environments? *British Journal of Educational Technology*, 41(1):10–32, 2010.

[25] M. de La Gorce, N. Paragios, and D.J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[26] Guillaume Dewaele, Frdric Devernay, and Radu Horaud. Hand motion from 3d point trajectories and a smooth surface model. In Toms Pajdla and Ji Matas, editors, *Computer Vision - ECCV 2004*, volume 3021 of *Lecture Notes in Computer Science*, pages 495–507. Springer Berlin Heidelberg, 2004.

[27] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 726–, Washington, DC, USA, 2003. IEEE Computer Society.

[28] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *CVIU*, 108(12):52 – 73, 2007.

[29] Catelijne Victorien Erren-Wolters, Henk van Dijk, Alexander C. de Kort, Maarten J. IJzerman, and Michiel J. Jannink. Virtual reality for mobility devices: training applications and clinical results: a review. *International Journal of Rehabilitation Research*, 30(2):91 – 96, 2007.

[30] Wim Fikkert, Paul Vet, Gerrit Veer, and Anton Nijholt. Gestures for large display control. In *Gesture in Embodied Communication and Human-Computer Interaction*, volume 5934 of *LNCS*, pages 245–256. 2010.

[31] William T Freeman and Michal Roth. Orientation histograms for hand gesture recognition. pages 296–301, 1994.

[32] L. Gallo and M. Ciampi. Wii remote-enhanced hand-computer interaction for 3d medical image analysis. In *Intl. Conf. Current Trends in Information Technology (CTIT)*, pages 1–6, 2009.

[33] L. Gallo, A.P. Placitelli, and M. Ciampi. Controller-free exploration of medical image data: Experiencing the kinect. In *24th Intl. Symp. Computer-Based Medical Systems*, pages 1–6, 2011.

[34] D.M Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82 – 98, 1999.

[35] Sébastien Grange, Terrence Fong, and Charles Baur. M/oris: A medical/operating room interaction system. In *Proceedings of the 6th International Conference on Multimodal Interfaces*, ICMI, pages 159–166, New York, NY, USA, 2004. ACM.

[36] M. Grant, C. Harrison, and B. Conway. Wheelchair simulation. In *Cambridge Workshop Series on Universal Access and Assistive Technology*. Springer Verlag, 2004.

[37] C. Grätzel, T. Fong, S. Grange, and C. Baur. A non-contact mouse for surgeon-computer interaction. *Technol. Health Care*, 12(3):245–257, August 2004.

[38] Feng Guo, D. Kimber, and E. Rieffel. Featured wand for 3d interaction. In *ICME*, pages 2230–2233, 2007.

[39] M. Hasanuzzaman, V. Ampornaramveth, Tao Zhang, M.A. Bhuiyan, Y. Shirai, and H. Ueno. Real-time vision-based gesture recognition for human robot interaction. In *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, pages 413–418, Aug 2004.

[40] Alexander G. Hauptmann and Paul McAvinney. Gestures with speech for graphic manipulation. *Int. J. Man-Mach. Stud.*, 38(2):231–249, February 1993.

[41] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the 4th European Conference on Computer Vision-Volume I - Volume I*, ECCV '96, pages 343–356, London, UK, UK, 1996. Springer-Verlag.

[42] D. Jack, R. Boian, A.S. Merians, M. Tremaine, G.C. Burdea, S.V. Adamovich, M. Recce, and H. Poizner. Virtual reality-enhanced stroke rehabilitation. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 9(3):308 –318, sept. 2001.

[43] Alejandro Jaimes and Nicu Sebe. Multimodal human-computer interaction: A survey. *Comput. Vis. Image Underst.*, 108(1-2):116–134, October 2007.

[44] Vijay John, Spela Ivekovic, and Emanuele Trucco. Articulated human motion tracking with hpso.

[45] MichaelJ. Jones and JamesM. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, 2002.

[46] Younbo Jung, Shih-Ching Yeh, and Jill Stewart. Tailoring virtual reality technology for stroke rehabilitation: a human factors design. In *CHI '06 extended abstracts on Human factors in computing systems*, CHI EA '06, pages 929–934, New York, NY, USA, 2006. ACM.

[47] Cem Keskin, Furkan Kra, YunusEmre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. In Andrea Fossati, Juergen Gall, Helmut Grabner, Xiaofeng Ren, and Kurt Konolige, editors, *Consumer Depth Cameras for Computer Vision*, Advances in Computer Vision and Pattern Recognition, pages 119–137. Springer London, 2013.

[48] Hyosun Kim, Georgia Albuquerque, Sven Havemann, and Dieter W. Fellner. Tangible 3d: Hand gesture interaction for immersive 3d modeling. In *Proceedings of the 11th Eurographics Conference on Virtual Environments*, EGVE'05, pages 191–199, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[49] Hyosun Kim and Dieter W. Fellner. Interaction with hand gesture for a back-projection wall. In *Proceedings of the Computer Graphics International*, CGI '04, pages 395–402, Washington, DC, USA, 2004. IEEE Computer Society.

[50] Marcus King, Juha Hijmans, Michael Sampson, Jessica Satherley, Nicole McMillan, and Leigh Hale. Bilateral movement training with computer games for stroke rehabilitation. In *Proceedings of the 4th International Convention on Rehabilitation Engineering & Assistive Technology*, iCREATe '10, pages 20:1–20:4, Kaki Bukit TechPark II,, Singapore, 2010. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre.

[51] Paul G. Kry, Adeline Pihuit, Adrien Bernhardt, and Marie-Paule Cani. Handnavigator: Hands-on interaction for desktop virtual reality. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, pages 53–60, New York, NY, USA, 2008. ACM.

[52] C. Kwok, D. Fox, and M. MEILA. Real-time particle filters. *Proceedings of the IEEE*, 92(3):469–484, Mar 2004.

[53] Byungsung Lee and Junchul Chun. Manipulation of virtual objects in marker-less ar system by fingertip tracking and hand gesture recognition. In *Proceedings of the 2Nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ICIS '09, pages 1110–1115, New York, NY, USA, 2009. ACM.

[54] Hyeon-Kyu Lee and Jin H. Kim. An hmm-based threshold model approach for gesture recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(10):961–973, October 1999.

[55] Bastian Leibe, Thad Starner, William Ribarsky, Zachary Wartell, David Krum, Brad Singletary, and Larry Hodges. The perceptive workbench: Toward spontaneous and natural interaction in semi-immersive virtual environments. In *Proceedings of the IEEE Virtual Reality 2000 Conference*, VR '00, pages 13–, Washington, DC, USA, 2000. IEEE Computer Society.

[56] John Y. Lin, Ying Wu, and Thomas S. Huang. 3d model-based hand tracking using stochastic direct search method. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, FGR' 04, pages 693–698, Washington, DC, USA, 2004. IEEE Computer Society.

[57] Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):286–299, February 2007.

[58] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(5):441–450, May 1991.

[59] Gan Lu, Lik-Kwan Shark, Geoff Hall, and Ulrike Zeshan. Dynamic hand gesture tracking and recognition for real-time immersive virtual object manipulation. In *Proceedings of the 2009 International Conference on CyberWorlds*, CW '09, pages 29–35, Washington, DC, USA, 2009. IEEE Computer Society.

[60] Minhua Ma, Michael McNeill, Darryl Charles, Suzanne McDonough, Jacqui Crosbie, Louise Oliver, and Clare McGoldrick. Adaptive virtual reality games for rehabilitation of motor disorders. In Constantine Stephanidis, editor, *Universal Access in Human-Computer Interaction. Ambient Interaction*, volume 4555 of *Lecture Notes in Computer Science*, pages 681–690. Springer Berlin / Heidelberg, 2007.

[61] Corey Manders, Farzam Farbiz, Tang Ka Yin, Yuan Miaolong, and Chua Gim Guan. A gesture control system for intuitive 3d interaction with virtual objects. *Comput. Animat. Virtual Worlds*, 21(2):117–129, March 2010.

[62] S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324, May 2007.

[63] Mathieu Nancel, Julie Wagner, Emmanuel Pietriga, Olivier Chapuis, and Wendy Mackay. Mid-air pan-and-zoom on wall-sized displays. In *CHI*, pages 177–186, 2011.

[64] Chan Wah Ng and Surendra Ranganath. Real-time gesture recognition system and application. *Image and Vision Computing*, 20(1314):993 – 1007, 2002.

[65] H. Niniss and T. Inoue. Electric wheelchair simulator for rehabilitation of persons with motor disability. In *Simposio Brasileiro de Realidade Virtual*, Balm, Brazil, 2005.

[66] R.G. O'Hagan, A. Zelinsky, and S. Rougeaux. Visual gesture interfaces for virtual environments. *Interacting with Computers*, 14(3):231 – 250, 2002.

[67] R.G. O'Hagan, A. Zelinsky, and S. Rougeaux. Visual gesture interfaces for virtual environments. *Interacting with Computers*, 14(3):231 – 250, 2002.

[68] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, volume 1, page 3, 2011.

[69] Sharon Oviatt, Antonella DeAngeli, and Karen Kuhn. Integration and synchronization of input modes during multimodal human-computer interaction. In *Referring Phenomena in a Multimedia Context and Their Computational Treatment*, ReferringPhenomena '97, pages 1–13, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[70] Vladimir I. Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):677–695, July 1997.

[71] John C. Platt. Advances in kernel methods. chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[72] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.

[73] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[74] Aditya Ramamoorthy, Namrata Vaswani, Santanu Chaudhury, and Subhashis Banerjee. Recognition of dynamic hand gestures. *Pattern Recognition*, 36(9):2069 – 2081, 2003. Kernel and Subspace Methods for Computer Vision.

[75] Zhou Ren, Junsong Yuan, Jingjing Meng, and Zhengyou Zhang. Robust part-based hand gesture recognition using kinect sensor. *Multimedia, IEEE Transactions on*, 15(5):1110–1120, Aug 2013.

[76] J. Romero, H. Kjellstrom, and D. Kragic. Monocular real-time 3d articulated hand pose estimation. In *IEEE-RAS International Conference on Humanoid Robots*, pages 87–92, Dec 2009.

[77] N. Rossol, I. Cheng, Rui Shen, and A. Basu. Touchfree medical interfaces. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 6597–6600, Aug 2014.

[78] Anara Sandygulova, Abraham G. Campbell, Mauro Dragone, and G.M.P O'Hare. Immersive human-robot interaction. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, pages 227–228, New York, NY, USA, 2012. ACM.

[79] Y. Sato, M. Saito, and H. Koike. Real-time input of 3d pose and gestures of a user's hand and its applications for hci. In *Virtual Reality, Proceedings of IEEE*, pages 79–86, March 2001.

[80] Albrecht Schmidt. Implicit human computer interaction through context. *Personal and Ubiquitous Computing*, 4(2-3):191–199, June 2000.

[81] Matthias Schwaller, Simon Brunner, and Denis Lalanne. Two handed mid-air gestural hci: Point + command. In *Human-Computer Interaction. Interaction Modalities and Techniques*, volume 8007 of *LNCS*, pages 388–397. 2013.

[82] Jakub Segen and Senthil Kumar. Look ma, no mouse! *Commun. ACM*, 43(7):102–109, July 2000.

[83] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, January 2013.

[84] Peng Song, Wooi Boon Goh, William Hutama, Chi-Wing Fu, and Xiaopei Liu. A handle bar metaphor for virtual object manipulation with mid-air interaction. In *CHI*, pages 1297–1306, 2012.

[85] Donald M. Spaeth, Harshal Mahajan, Amol Karmarkar, Diane Collins, Rory A. Cooper, and Michael L. Boninger. Development of a wheelchair virtual driving environment: Trials with subjects with traumatic brain injury. *Archives of Physical Medicine and Rehabilitation*, 89(5):996 – 1003, 2008.

[86] Thad Starner, Alex Pentland, and Joshua Weaver. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1371–1375, December 1998.

[87] B. Stenger, P.R.S. Mendonca, and R. Cipolla. Model-based 3d tracking of an articulated hand. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–310–II–315 vol.2, 2001.

[88] Mu-Chun Su. A fuzzy rule-based approach to spatio-temporal hand gesture recognition. *Trans. Sys. Man Cyber Part C*, 30(2):276–281, May 2000.

[89] Shu-Li Sun and Zi-Li Deng. Multi-sensor optimal information fusion kalman filter. *Automatica*, 40(6):1017 – 1023, 2004.

[90] Jochen Triesch and Christoph von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(12):1449–1453, December 2001.

[91] Daniel Vogel and Ravin Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *UIST*, pages 33–42, 2005.

[92] Christian von Hardenberg and François Bérard. Bare-hand human-computer interaction. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, PUI, pages 1–8, New York, NY, USA, 2001. ACM.

[93] Juan P. Wachs, Helman I. Stern, Yael Edan, Michael Gillam, Jon Handler, Craig Feied, and Mark Smith. A gesture-based tool for sterile browsing of radiology images. *Journal of the American Medical Informatics Association*, 15(3):321 – 323, 2008.

[94] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Commun. ACM*, 54(2):60–71, 2011.

[95] Frank Weichert, Daniel Bachmann, Bartholomus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393, 2013.

[96] Alan Wexelblat. An approach to natural gesture in virtual environments. *ACM Trans. Comput.-Hum. Interact.*, 2(3):179–200, September 1995.

[97] Andrew D. Wilson and Aaron F. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):884–900, September 1999.

[98] Andrew D. Wilson and Aaron F. Bobick. Hidden markov models. chapter Hidden Markov Models for Modeling and Recognizing Gesture Under Variation, pages 123–160. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2002.

[99] Ying Wu, J.Y. Lin, and T.S. Huang. Capturing natural hand articulation. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 426–432 vol.2, 2001.

[100] Yingen Xiong, Francis Quek, and David McNeill. Hand gesture symmetric behavior detection and analysis in natural conversation. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, ICMI '02, pages 179–, Washington, DC, USA, 2002. IEEE Computer Society.

[101] Zhaohong Xu, Hongliu Yu, and Shiju Yan. Motor rehabilitation training after stroke using haptic handwriting and games. In *Proceedings of the 4th International Convention on Rehabilitation Engineering & Assistive Technology*, iCREATe '10, pages 31:1–31:4, Kaki Bukit TechPark II,, Singapore, 2010. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre.

[102] Xiaoming Yin and Ming Xie. Estimation of the fundamental matrix from uncalibrated stereo hand images for 3d hand gesture recognition. *Pattern Recognition*, 36(3):567 – 584, 2003.

[103] Quan Yuan, Stan Sclaroff, and Vassilis Athitsos. Automatic 2d hand tracking in video sequences. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1 - Volume 01*, WACV-MOTION '05, pages 250–256, Washington, DC, USA, 2005. IEEE Computer Society.

[104] Yuanxin Zhu, Guangyou Xu, and David J. Kriegman. A real-time approach to the spotting, representation, and recognition of hand gestures for humancomputer interaction. *Computer Vision and Image Understanding*, 85(3):189 – 208, 2002.