PACE: A Framework for Personalized Visualization and Scalable Human Computer Interaction

BY

XUN LUO
B.S. UNIVERSITY OF ELECTRONIC SCIENCE
AND TECHNOLOGY, CHINA, 1999
M.S. UNIVERSITY OF ELECTRONIC SCIENCE
AND TECHNOLOGY, CHINA, 2002

THESIS
Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Sciences
in the Graduate College of the
University of Illinois at Chicago, 2008

Chicago, Illinois

To my parents and Ran

AKNOWLEDGEMENT

Above all, I would like to express sincere thanks to my advisor Professor Robert V. Kenyon for his insightful guidance, enlightening advice, and endless encouragement throughout my Ph.D. study, which has given me a great opportunity to explore various difficult but interesting problems. I was lucky and am proud of being a student of him, a great man with extraordinary vision and wisdom. Especially, I would like to thank my mentors in Northwestern Sensory and Motor Performance Program (SMPP), Qualcomm R&D and Motorola Labs, Dr. Derek Kamper, Dr. Sumit Sur and Dr. Weimin Xiao for their selfless discussions and suggestions, without which I could have not made this work possible. I would also like to thank my Ph.D. advisory committee members Professor Peter Nelson, Professor Ashfaq Khokhar, and Professor Jason Leigh, for their inspiring and constructive discussions during my study.

I also would like to thank all my colleagues in the Electronic Visualization Laboratory and many friends in SMPP, Qualcomm R&D and Motorola Labs. In particular, I would like to thank Dr. Dan Sandin, Dr. Javier Girado, Dr. Qunli Sun, Wei Xie, Dr. Julia Yu and Dr. Xiaohua Wu.

I wish to thank my family for all their endless love, support and encouragement though all the time of my study abroad. Finally, but not least, I would like to express my deep thanks to my dear wife, Ran, for all her love, sacrifice, understanding and help, which could be felt in every word in this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Virtual reality (VR) is a technology which allows a user to interact with a computer-simulated virtual environment (VE). Since its inception it has created an impact in many fields, but has not yet gained wide acceptance to be used in a personal context. At the same time, personal mobile devices (MDs) have evolved tremendously during recent years. The deployment of MDs is becoming pervasive; the hardware configurations are enhanced to be comparable to low-end desktop systems; and there are drastic improvements in the infrastructures supporting wireless inter-device connectivity and collaboration. These trends make novel applications and business models of VEs by the individual users promising and desirable. However, to make the key elements of a VE, i.e. visualization and interaction, work as smoothly for MD-based systems as for desktop-based systems, three research challenges need to be addressed, for which the technical solution portfolio is far from complete: 1) which visual factors of a VE make the user's performance similar to that in the real world? 2) How to effectively allow a user to collect his/her own biometrical data to build personalized profiles? 3) How to efficiently make use of the processing resources in the infrastructure for the compute-heavy tasks for a individual user, such as intelligent human computer interaction?

This work, referred to as Personal Augmented Computing Environment (PACE), sets its goal for *personalized* visualization and *scalable* human-computer interaction. It attacks the three challenging research problems with corresponding solutions. First, to understand the visual factors that make VE more like the real world, a set of controlled experiments are designed and conducted in a CAVE system. In the experiments three visual factors, namely scene complexity, stereovision and motion parallax are examined. The results suggest that scene complexity and stereovision significantly affect users' size perception, while motion parallax does not exhibit a significant effect. Being conducted by no other researchers before, this study helps us to better understand the role of size constancy in VE performance.

Next, an effective process to collect a user's hand reference images for posture profile building is proposed and implemented. The novelty of the proposed process is that it takes advantage of the synergies between mobile device and computing infrastructure, and uses proactive measure instead of post-processing techniques to improve sample image quality. While most hand posture recognizers' performance are very sensitive to hand sample image quality and require them to be taken under strictly controlled laboratory setting, by employing the process proposed and implemented by this work, the mobile device user can collect hand sample images by themselves with relative ease, and still be able to obtain satisfying posture profiles.

The third contribution of this work is a set of scalable computing techniques to speed up the tasks of a vision-based hand detection and recognition using computer clusters. These techniques include a novel data structure, called a scanning node tree which is used to manage cluster nodes; a unique load balancing algorithm to evenly distribute workload across nodes; and a node-to-node messaging protocol for parallel processing. The set of scalable computing techniques has been proved to be efficient by various evaluation metrics. Enabled by these techniques, a sample application, a hand tracker/controller named Hand

Wand is implemented and integrated with a large scale tiled display instrument as a human computer interaction device.

This work also introduces one preliminary and early work for personal VE: a reach-and-grasp training environment for in-home rehabilitation of stroke survivors. This pilot study achieved certain results and were successful to some extent.

1. INTRODUCTION

Virtual reality (VR) is a technology which allows a user to interact with a computer-simulated environment, be it a real or imagined one. Since the inception of VR, it has been used for applications in various fields. Just to name a few: military bases use VR to simulate battle field or air combat [7]; manufacturing factories use VR for mocking up of assembly process [6]; medical and health care facilities use VR to plan life-critical surgeries before they are carried out on real patients [1]; and scientists use VR to display complex or large scale data sets [5].

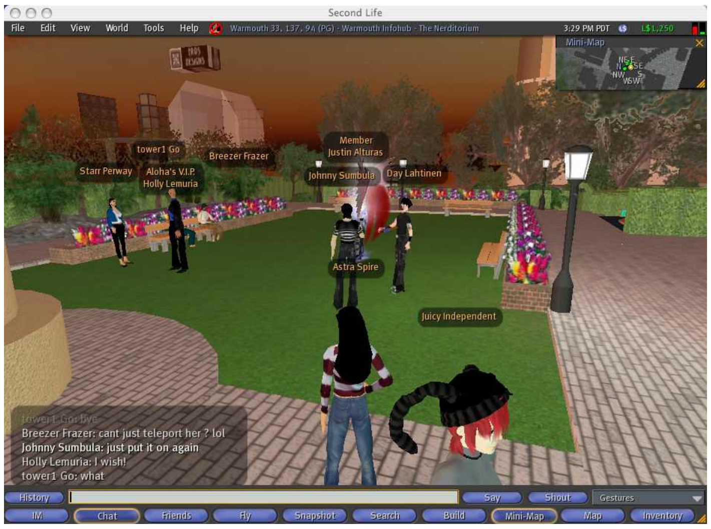
Although VR has achieved many successes, people's prediction that VR will be integrated into daily life and activity has not been obtained yet. To make the wide acceptance of VR for individuals possible, the VR system should be able to provide personalized and close-to-real-life visualization and interactivity for its user. On the visualization aspect, not only photo-realistic virtual scenes need to be rendered, presence of these virtual scenes also needs to have a high-fidelity for virtual-real world registration. On the interaction side, the VE should be able to be interacted with using the modalities a user interacts in the real world, to blend itself into the user's daily life seamlessly. This means that a VE system should maximize itself in sensing the presence and status of the user; the user's freedom of movement should be as unconstrained as possible; and the display, interaction and computation components of the VR system should satisfy the size, price and power consumption requirements for portable use. As described in the following sections, technology advances in the past decades have created the consumer needs and laid the foundation for personal VEs. In the following text of this chapter I firstly review the background of recent technology advances in VEs, mobile devices and infrastructures. Then the motivations of this work are explained and research goals to be achieved are stated.

1.1. <u>Trend of Virtual Environments</u>

Networked VEs for personal entertainment purpose emerge and quickly become popular in the past decades. In a networked VE, users are shown as 3D avatars and the virtual universe is composed of such avatars, as well as the virtual objects and scenes the avatars reside in. Two representative virtual universes are *Second Life* [25] and *There* [26]. Figure 1 illustrates a typical scene in *Second Life*, where multiple avatars of the users are socializing in a patio. Because of the potential business opportunities, integration of networked environments and location-based applications has now drawn the attention of wearable system vendors. Location-based VE provides computer generated visual and audio content based on the location context of the user, thus calling for more seamless registration between the virtual world and the real world. The popularity of applications fosters the volume of current and future VE systems. For example, *Second Life* has a reported user population of 2 million as of 2006.

Besides virtual universe applications, many traditional information sources are now explored by the users through modalities similar to what used to be categorized as virtual reality. For example, new geographical information systems that allow people to see a location using street map, satellite view, "hybrid" view that integrates the former two, and even 3D view of the panoramic camera images of the streets have been offered by major web service companies and is now used by millions of users, over a broad spectrum of computing platforms, including desktops and mobile devices.

The practical needs for networked VE applications also make the needs for wearable displays to increase. Compare to the immersive HMDs in traditional VEs, augmented displays suit this purpose better because the real world and VE can be optically blended to facilitate a user's experience.



tower1 Go: bye
Breezer Frazer: cant just teleport her ? lol
Johnny Sumbula: just put it on again
Holly Lemuria: I wish!
tower1 Go: what

Figure 1 the Second Life Virtual World

1.2. <u>Contemporary Personal Mobile Devices</u>

The definition for a mobile device was and till now, is still quite fluid. A broad consensus now is that mobile devices are pocket-sized computing devices, typically comprising a small visual display screen for user output and an interactive gadget for user input. Today, the category of mobile devices encompass mobile phones, personal digital assistants (PDAs), portable media players, information appliances, personal communicator, handheld game consoles, and ultra-mobile PCs (UMPCs), just name a few. The use of mobile devices nowadays is virtually pervasive. Take mobile phones alone, as of end of 2007, about half of the world population owns one[1]. Due to frequent feature and configuration overlaps, it is not always easy to differentiate one type of mobile device from another (e.g., a mobile phone and a portable media player). Also, nearly all these devices exhibit the common trends towards more powerful processing units, standardized and interoperable application platforms, and natural human computer interfaces.

Processing Power

Because of the high requirements for power saving features, a majority of mobile devices use ARM (Advanced RISC Machine) architecture processors. The ARM architecture is a 32-bit RISC processor architecture developed by ARM Ltd. that is widely used in a number of embedded designs. Power-saving as they are, computing power of the ARM processors are rich and also follow Moore's Law (i.e. doubled every two years) as the counterparts in desktop, workstation and sever fields do.

Typical ARM processors on mobile devices today are running at 300 – 800 million instructions per second (MIPS) range. Such processing speeds are still not comparable to main stream desktop computer processors, but close to the one two years ago. In other words, it could be assumed that mobile device's computing power is one level lower on the ladder of Moore's Law with desktops, and is closely tracking desktop systems at this pace. This fact makes migration of desktop application to mobile devices a natural choice and in practice, is already underway.

Application Platforms

[1] Source from industry analyst Informa Telecoms & Media: http://www.informatm.com/

One challenge in making mobile devices interoperable is that applications developed for one brand and/or model of mobile device were generally not able to run on other brands and/or models. Also, applications on mobile devices were largely in the embedded fashion, where the application itself is taking the responsibility of scheduling, resource management and communication with other processes. Embedded applications are difficult to develop, maintain and interoperate. As a consequence, most applications only run on proprietary systems and universal deployment among various mobile devices is not common. These situations were true until early 2000's. After that, application platforms that facilitate interoperability became widely deployed. Nowadays there are three major platforms: Sun's Java Micro Edition (J2ME), Qualcomm's Binary Runtime Environment for Wireless (BREW) and Microsoft's compact .NET framework. The driving forces for these platforms are strong and these platforms are evolving rapidly. For example, J2ME provides a reduced version of Java libraries to mobile device application developers. Java application written in J2ME could run on any mobile devices that are J2ME compatible, be it a mobile phone, a portable media player or a PDA. J2ME libraries are standardized through the form of Java Specification Requests (JSRs). As of 2007, 927 JSRs have been proposed and standardized[2], making a large variety of application features available through J2ME-compatible mobile devices.

Standardize and interoperable application platforms make possible the flourishing of mobile device applications. Among these prospective applications are personal VE systems.

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

Human Computer Interfaces

The main human-computer interfaces on mobile device phones today are QWERTY style keyboards and touch screens. There are also devices that make use of inertial sensing (Nintendo Wii) and touch-based gestures (Apple iPhone).

In 1997, the first complete integrated cellular phone and camera solution was demonstrated in public. Since then, the advent of the CMOS sensor manufacturing process enabled the camera phone technology for mass production. As of 2007, all major mobile phone manufacturers offer camera phone in their product lines. Besides merely taking a picture, storing locally or sending out, multiple mobile devices manufacturers also provide computer vision software libraries that facilitate image and video processing, such as [24]. There are still plenty of spaces for novel human-computer interaction methods for mobile devices.

1.3. Emerging Infrastructures

Personal visualization and interaction tasks are performed in the ecosystem of infrastructures (communication and computing services that are not with a specific user), which in the past decades have undergone enormous transitions. These transitions could be summarized from communication and computation perspectives. From a communications perspective, short range wireless protocols enable data to be transferred at higher throughput and lower latency; while last mile wireless gap is largely filled by third generation mobile phone technology, making mobile applications more convenient to be networked smoothly to internet backbones. From a computation perspective, processing power and storage become standardized resources that can be consumed by personal users, introducing the concept of

Utility Computing or more contemporary, Software as a Service and Infrastructure as a Service.

Short Range Wireless Communications Infrastructures

Till late 1990's, mobile devices were connected to base stations either through star link or bee-hive networks. Local wireless interconnectivities were mostly low-bandwidth, such as Infrared Data Association (IrDA). With the increasing need for high-bandwidth and low-latency local wireless link, several technologies have now gained popularity by hardware manufactures. The most widely deployed one is Bluetooth [1]. Bluetooth is an industrial specification for wireless personal area networks and provides a way to connect and exchange information between devices such as mobile phones, laptops, PCs, printers, digital cameras, and video game consoles over a secure, globally unlicensed short-range radio frequency. At the time of this work, the mainstream Bluetooth specification version is 1.2, which supports 2.1M bit/s of transmission speed.

The new Bluetooth specification, version 3.0, plans to incorporate ultra-wideband (UWB). UWB is radio communications technology that can be used for short-range high-bandwidth communications by using a large portion of the radio spectrum in a way that does not interfere with other more traditional "narrow band" uses. UWB could achieve data transmission rate at 480M bit/s. UWB integration will create a new version of Bluetooth wireless technology with a high-speed/high-data-rate option. This new version of the Bluetooth technology will meet the high-speed demands of synchronizing and transferring large amounts of data, as well as enabling high-quality video and audio applications for a variety of mobile devices.

Besides radio technologies, high-speed, high-bandwidth short range optical communication is also promising to be practically deployed for mobile devices. Optical signal transmitted through dedicated fiber link have already proved to be able to carry digitized information at gigabit or even terabit per second speed, depending on the number of fibers in optical cable housing. Optical transmission in free space is technically more challenging than in fiber media, but has gained substantial improvements in component miniaturization, optical path stability and optical-electronic conversion on mobile devices. There are quite a few industrial-quality mobile device components that are able to transmit data at gigabit per second rate. Although the price and power consumption are still prohibitive, reduction of both are confidently expected in the near future.

The improvements in short-range wireless communication infrastructures greatly facilitate device interoperability.

Last-Mile Wireless Communications Infrastructures

One trend of last-mile wireless communications is the increasing adoption of the third generation (3G) mobile phone standards and technology [2]. 3G technologies enable network operators to offer users a wider range of more advanced services while achieving network capacity through improved spectral efficiency. The most significant feature of 3G is that it supports greater numbers of voice and data customers – especially in urban areas – and higher data rates at lower incremental costs than 2G mobile systems. A typical 3G deployment allows the transmission of 384k bits per second for mobile systems and 2M bits per second for stationary systems. Novel services include wide-area wireless voice telephony and broadband wireless data can be carried out in harmony in a mobile environment. As of 2007, 200 million 3G subscriber have been connected[3].

Another emerging last-mile wireless technology is the Worldwide Interoperability for Microwave Access (WiMAX) [3]. WiMAX aimed at providing wireless data over long distances in a variety of ways, from point-to-point links to full mobile cellular type access. WiMAX is based on the IEEE 802.16 standard, which is also called WirelessMAN. A typical WiMAX deployment allows the transmission of 70M bits per second over a 2 kilometers distance. At the transmission rate as low as 2M bits per second, WiMAX systems are able to operate over a 50 kilometers distance.

The improvements in last-mile wireless communication fill the gap between mobile end users and the main internet trunk, which used to be limited by carrier offerings. The difference between mobile communication units and internet terminals are largely morphed, if not diminished completely yet.

Computation Infrastructures

In the past, corporate or academic mainframes were jealously guarded as strategic resources. With the reduction in price of commodity PC-based clusters, as well as proliferation of open source software, the new concept *Utility Computing* starts to be accepted by the computer users. Utility computing is the packaging of computing resources, such as computation and storage, as a metered service similar to a physical public utility (such as water or natural gas). This kind of system has the advantage of a low or no initial

[3] Source: http://en.wikipedia.org/wiki/3G

cost to acquire hardware; instead, computational resources are essentially rented. Customers with very large computations or a sudden peak in demand can also avoid the delays that would result from physically acquiring and assembling a large number of computers. Utility computing was initiated by IBM and now both Sun and Amazon have commercial offerings.

Two more modern terms similar to utility computing are Software as a Service (SaaS) and Infrastructure as a Service (IaaS). Both are targeting at flexibly allocated hardware and software resources to be used by individual users. In a ubiquitous computing environment, the existence of such kind of resources is very important for the user to be able to fulfill visualization and interaction tasks.

1.4. Motivation

The previous sections of this chapter briefly summarize the trend of VEs, features of contemporary mobile devices and the emerging infrastructures. These facts justify the practical need of introducing VEs for personal use, as well as the feasibility provided by the fast improving device and infrastructure capabilities. However, research portfolio to archive the smooth migration of today's VEs for personal use is far from complete. Specifically, the following challenges need to be concretely addressed:

1.4.1. Limitation in Understanding of Size Perception Performance

Being different from traditional personal computers' two-dimensional visual interfaces, three-dimensional (3D) user interface is one of the most important hallmarks of VEs and should be well implemented for personal use. Correct perception of the virtual object size is one of the key metrics of efficient 3D user interface implementation, and worth being carefully studied. This is especially true in the personal use context, where high-fidelity registration between the virtual environment and the physical world is desired. Here I present two fictitious scenarios where the correct size perception is the key to the application's user experience:

Scenario 1:

A stock broking agency provides its customers with wearable 3D HMDs and wireless broadband receivers to get the real-time stock price information. Stock prices are plotted in interactive 3D graphics as bars, spheres, and cubes, etc. In a HMD, 200 stocks are displayed in the same screen and each are tagged with relative review articles to be read from. A busy user browses the 3D visualization every morning and skim only these stocks that he feels at a higher price than others, based on the <u>*size*</u> *of the bar, sphere or cube he perceived.*

Scenario 2:

A driver uses a portable GPS mounted on the car to navigate routes while driving. Using a VE, the GPS displays all the buildings along the road as 3D models. The GPS also communicate with other GPS devices in the nearby vehicles and display these vehicles in 3D models on the screen. The driver counts on his perception of the building and car <u>*sizes*</u> *to adjust driving speed.*

Both applications illustrated are common for personal VE systems (similar services are available now but have not yet been used extensively), and sensitive to the size perception performance of their users. In Chapter 3 I describe in detail a deployed personal VE that trains post-stroke patients of reach-and-grasp tasks, which is dependent on its user's size perception performance.

Important as it is, to my best knowledge, there is no literature or systematic human study that addresses all major VE factor which affect user's size perception yet. Three of these factors are scene complexity, stereovision and motion parallax. Thus, studying to identify the significant factors that affect people's size perception in VE is a desirable research task.

1.4.2. The Need for Natural Interaction for VR Tracking

Being able to track the user is an important feature of VEs, because of the high requirement for interactivity. To make a VE ready for personal and mobile use, tracking module should ideally be tetherless, and able to track in 3D space. At the same time, current user interfaces are not fully suited for human-centered VEs. Traditional devices as keyboards and mice offer only the capability of working on a planar surface. Novel devices could be categorized in three classes: 1) Joystick-like devices, such as FlyBox (BG Systems) and Cubic Mouse (FakeSpace Systems). These devices are either mounted on the desk or manipulated with both hands. 2) Wearable devices, such as the PINCH gloves (FakeSpace System). These devices track the wearer's body movement and trigger respective commands. 3) Single hand-held wands, such as the Wand (Intersense). Through integration of 3D position tracking, these devices allow single hand operation, as well as certain degrees of body movement. Although these devices provide great convenience in traditional VEs, they are not suitable for personal VEs. First, they put high requirements for the user's operation skills, which is hard for VE novices. Second, users are burdened with device carrying, either explicitly or implicitly. In fact, the limitations of current human-computer interfaces hinder expansion of the VEs to serve individual uses.

1.4.3. Lack of Scalable Computing Techniques for Mobile Devices

The pervasiveness of mobile devices and the abundance of infrastructure processing power do not automatically make the effective synergy between these two of entities. In fact, most exchanges between today's mobile devices and the infrastructure are application data: streaming of multimedia contents, for example, video and audio; download or upload personal files to web servers, such as social networking or self-publishing sites; and, synchronizing personal information (email, office documents) with desktop systems. The ideal exchange between mobile devices and computer systems in the infrastructure should be both application data as well as user profiles. Mobile devices should be able to transfer its user profile to the infrastructure in a nearly transparent and online way that minimizes the user's awareness. At the same time, accesses to the infrastructure processing power, including discovery of the computation resource in the infrastructures, initialization of collaboration sessions, as well as quality-of-service control should all be automated and effectively managed.

Here is yet another scenario that shows the usefulness of scalable computing techniques for mobile devices[4]:

An English-speaking traveler is having a business trip in Tokyo and walked into a Japanese sushi bar. Unfortunately, all the menus and store signs in the sushi bar are in Japanese and the traveler cannot read them at all. Luckily, powerful computer system has been deployed in the sushi bar, and available for customer access through wireless link. So the traveler use his mobile device to take continuous video stream of Japanese menus and store signs and offloads the computer-vision based processing to the computer system in the

[4] Personal communication with Professor Jason Leigh, one of my thesis committee members.

sushi bar. The traveler's mobile device already stored his native language (English) and dietary preferences as personal profiles. Such information personalizes the image processing process. When the processed result is rendered to the user, menus and store signs are translated into English and with a "recommendation" ranking displayed next to each of them.

Although this scenario is fictitious, it clearly shows why scalable computing for mobile devices is still a unsolved research problem today and how big the impact will be if this research problem is gracefully solved.

1.5. Thesis Goals

Motivated by the research problems described in the above section, this work sets three goals to accomplish. 1) to study the visual factors that affect people's size perception performance in a VE. 2) to develop an effective solution to construct user profile for hand posture recognition. 3) to investigate scalable computing techniques to facilitate the synergy between mobile devices and infrastructure computer systems to implement a high-performance hand posture based tracker and controller. Grouping these three goals together, this work is aiming at implement a prototypical framework for personalized visualization and scalable human computer interaction, which is named personal augmented computing environment (PACE). The balance between advanced research and practical implementation is carefully planned when PACE is envisioned. And this balance is reflected in the planning of all the three goals described as below.

1.5.1. Study of Size Constancy

Size constancy refers to the capability of a person to correctly perceive object sizes in the environment, independent of the objects' relative location from the viewer. As size constancy in the physical world is natural, users' size constancy performance is an important metric of the quality of a VR system. This research sets its goal to study three visual factors that might have an effect on the users' size constancy performance in the VE: scene complexity, stereoscopy and motion parallax. To achieve this goal, controlled experiments that configure a VE with difference combination of these three visual factors are to be conducted, data from VR-experienced and VR-naive subjects are to be collected, and statistical analysis methods are to be applied to get quantified conclusion of the significance of difference visual factors in affecting size constancy performance.

The VR system that is chosen to conduct the control experiments is the CAVE Automatic Virtual Environment (CAVE®[5]), a projection based virtual reality system developed at the Electronic Visualization Lab at the University of Illinois of Chicago (EVL-UIC). The reason to choose a projection-based VR system and not more contemporary LCD-based VR systems or mobile device based VR systems is that CAVE is a mature, widely deployed, and representative VR system. In the mean time, CAVE system is less prone to the interference of other "noisy" environment factors, such as motion artifacts in mobile VR systems caused by human body movement and perspective skew in LCD based VR systems caused by imprecise calibration of LCD screen mountings.

1.5.2. <u>Personalized Hand Tracker and Posture Recognizer</u>

Using the most dexterous part of human body – the hand – for human computer interaction in the VE conforms to users' daily life habit, and has other important advantages such as tetherless interaction and expanded degrees of freedom. Making the users' hand

[5] The CAVE is trademark of the Board of Trustees of the University of Illinois at Chicago

capable of tracking and controlling objects in the VE will greatly facilitate the wide deployment of such systems, and push them closer towards portability and mobility. This work proposes to realize a hand tracker using computer vision techniques. Hand images are to be captured, processed, and mapped to 2D coordinates as well as control commands at the rate of VR frame update. A set of hand postures are defined as symbol for control commands and when they are performed by the VE user, corresponding commands are issued to the VE.

To personalize the proposed tracker, user profile needs to be effectively constructed. This work aims at implementation of a process that mobile device user are able to take the hand sample images by themselves and achieve comparable profile accuracies as what used to be done in a strictly controlled environment. The profiles to be constructed include the user's hand shape and skin color models.

1.5.3. Scalable Computing for Hand Tracking

To investigate solutions that effectively create the synergy between the mobile device and computer systems in the infrastructure, this work aims at designing and implementing a prototypical paradigm that use the computation clusters in the infrastructure to boost the computation tasks performed by the hand tracker as described in section 1.5.2. Discovery, initiation and process offloading to the cluster system in the infrastructure are to be automated for the mobile device user, thus to achieve minimal user involvement and awareness of the scalable computing process. Two components of the hand tracker, namely hand detection and posture recognition are to be accelerated through collaboration among nodes of the cluster.

The computation cluster to be used in this experimental work is a 32 node cluster which serves as a large-scale display node on the global Grid. The cluster system drives a 55-tile LCD display wall and is an exemplary supercomputing virtual organization (VO) on the Grid which has underused computation capabilities. Without loss of generality, this system could be regarded as representative of the computer systems ubiquitously existing in a mobile device user's surroundings.

1.6. Chapter Organization

This thesis is organized as follows. Following the background introduced in Chapter 1, Chapter 2 gives a review of relevant research in addressing similar problems. In Chapter 3, a pilot study conducted by the author to deploy personal VE for stroke survivors, named VR Hand Trainer is presented. Being an import preliminary work preceding the final approaches chosen by the thesis research, VR Hand Trainer provides valuable lessons to for the final approaches of this work. Chapter 4, 5, 6 compose the main body of this thesis. A human factors study to identify visual factors that significantly determine VE users' size constancy performance is described in Chapter 4. In Chapter 5, extension work from tracking a rigid human part, i.e. the head to an articulated human body part, i.e. the hand, are proposed and the implementation details are given. Chapter 6 presents the scalable computing techniques employed to solve this computation-intensive problem. These techniques are then evaluated by a real-world application for interaction with a large scale tiled display system in Chapter 7. At last, Chapter 8 draws the conclusions for this research work, and gives discussion on lessons learned and possible future works.

2. RELATED WORK

Making use of mobility and hand based interaction techniques has been a long-standing problem in mobile computing, ubiquitous computing, location-aware computing and virtual reality communities. Along with the various researches on this topic, a large number of approaches have been proposed and a variety of prototypes have been constructed based on the approaches. Some of them emphasize the use of external cameras, while the others prefer wearable cameras and other portable devices. In this chapter, several representative approaches and systems are briefly reviewed to set the reference baseline of this work.

The Finger Mouse

In [28], Quek and colleagues presented Finger Mouse, which is a freehand pointing alternative to the ubiquitous mouse. In this system, the user merely performs a pointing gesture above the keyboard. A down-looking camera is trained on the keyboard. A user typing at the keyboard may switch into the “pointing mode” by simply assuming a hand pointing configuration above the keyboard. A vision system constantly monitors the hand and tracks the fingertip of the pointing hand. As the user gestures in a horizontal plane just above the keyboard, the screen cursor moves accordingly. The user depresses the SHIFT key on the keyboard with the non-pointing hand to register a ’mouse button press’.

Although Finger Mouse has real-time performance, it only tracks fingertip rather than the hand posture. It is also not personalized. The limitations of Finger Mouse from the viewpoint of nowadays can not only be attributed to the technology restrictions in the 1990s’, but also the design goal of the system. The design goal for Finger Mouse is clearly for 2D desktop use, rather than immersive and mobile systems: camera is passive and mounted in the desktop workspace; usage of mouse is in conjunction with the keyboard; strict constraints are imposed for background noise. However, Finger Mouse's concept of using the hands as a mixture of tracking and interaction devices, of which one hand performs the state switch task and the other serves the tracking task, could shed light on this research. The further research by Xiong and Quek [29] about unconstrained hand gesture extraction by analyzing oscillation cues is related to endeavors towards minimized user involvement and awareness.

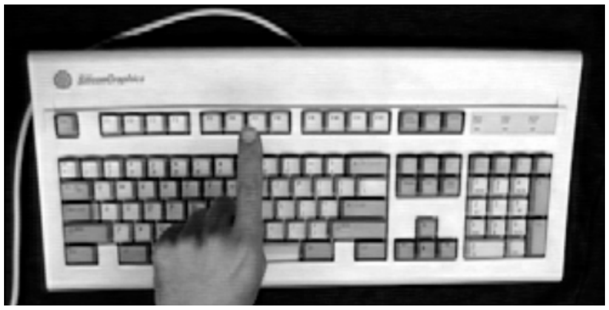


Figure 2 Finger Mouse by University of Illinois at Chicago

Incorporating Dynamic Real Objects with VE

In [27] Lok presented algorithms to generate virtual representations, avatars, of dynamic real objects at interactive rates. Further, he presented algorithms to allow virtual objects to interact with and respond to the real-object avatars. That allowed dynamic real objects, such as the user, tools, and parts, to be visually and physically incorporated into the VE. The system uses image-based object reconstruction and a volume-querying mechanism to detect collisions and to determine plausible collision responses between virtual objects and the real-time avatars. Those techniques allowed their system to provide the user natural interactions with the VE and visually faithful avatars.

Lok's research, in line with a few others, emphasizes the reconstruction of real objects in immersive VE. The purpose is not to create human computer interfaces that are suitable for mobile users, but rather enhance the users' experience in fixed setting VEs like the CAVE. This design principle is reflected by the using of both mounted and wearable cameras and the high requirement on the calibration of mounted cameras.

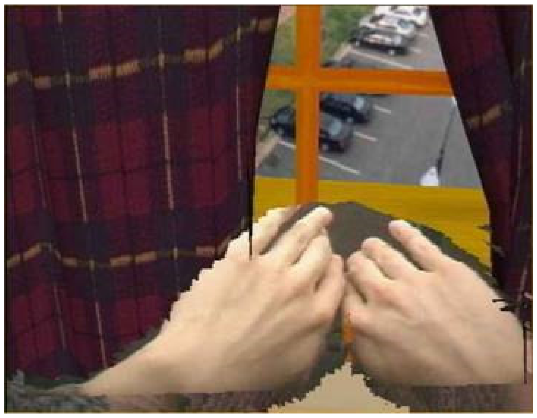


Figure 3 Real Object – VE integration by University of North Carolina

Gesture Pendant

At the Georgia Institute of Technology, Starner and fellow researchers did extensive work [30, 31] on real-time gesture recognition. One of their earlier applications was for American Sign Language (ASL) recognition [31]. The technology was later been extended to recognize user-defined gestures. With the aid of a 5-word vocabulary, the Gesture Pendant allows ordinary household devices to be controlled, literally, with the wave of a hand. The user wears a small pendant that contains a wireless camera. The user makes gestures in front of the pendant that control anything from a home theater system, to lighting, to the kitchen sink. Therefore, hard to use, hard to understand remotes can be replaced with simple hand gestures. The research is still active and evolving. In [30] they used both acoustic sensor and accelerometers for ASL recognition. The restriction of the gesture pendant is that it requires active illumination and hand distance to be optimally set. At the same time, gesture recognition is based on Hidden Markov Model trained from grey-level images. These features put barriers on the application of more advanced computer vision techniques. In the mean time, the Gesture Pendant is designed to be a control device only and not a tracking device, neither 2D nor 3D.

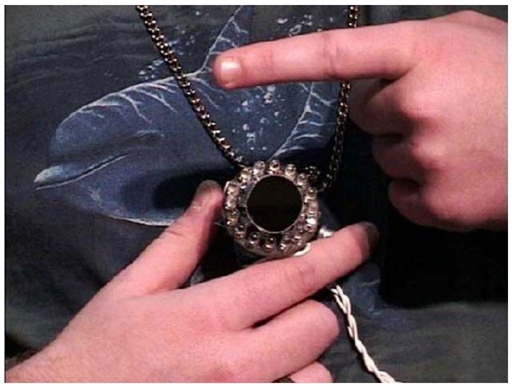


Figure 4 Gesture Pendant by Georgia Tech

Hand Mouse

In the Hand Mouse project by Kurata and colleagues [32], the user carries a wearable computer, wear HMD and use hand as the sole input for HCI. The system uses color-based segmentation to classify hand from the background, and a Gaussian Mixed Model (GMM) to dynamically adjust color histogram models. Video captured from wearable camera are streamed to a desktop system (Dual-Processor Pentium III) via wireless LAN for all processing and the results are sent back to the user. At the same time a video combiner feeds the captured video back to the HMD to create augmented reality.

The color-based segmentation technique used by Hand Mouse system is sensitive to background noise, especially static objects that are of similar color and shape of the human hand. Although the authors claim cluster computing techniques are used for parallel processing, the processing computer they used is merely a high-availability desktop system rather than a real computer cluster. The wearable computer is used as a video terminal only. This solution puts a large amount of load on the wireless link and makes the system prone to quality-of-service problems.

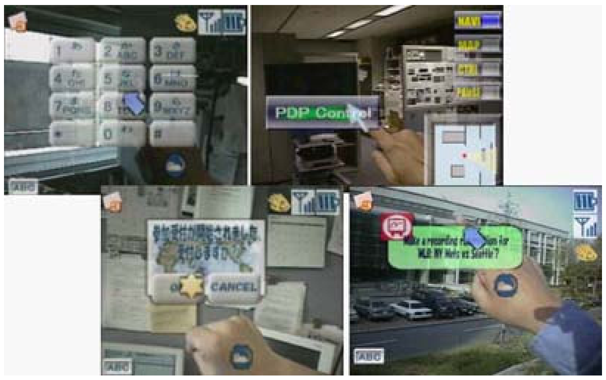


Figure 5 Hand Mouse by JAIST

HandVu

HandVu [33] is a software/hardware combination developed by Kolsch at the University of California at Santa Barbara. It achieved robust hand tracking, in a real-time manner (10 – 45ms for per frame processing). Hand tracking is the main concentration of HandVu, and it assigned gesture recognition to application. Based on texture and color features of the video image, HandVu is able to detect “Flock of Features” and is more robust to background noise interference.

HandVu does not give its users the freedom to train hand shape and skin colors of their own. It uses a pre-computed general hand shape model for posture recognition. It also uses a predefined skin color model for hand detection, and relearns the color profile after the hand is detected. Thus, hand detection is not personalized either. At the same time, HandVu solely counts on a wearable laptop system rather than scalable computing resources to fulfill all the computation tasks, include hand detection, tracking and recognition.

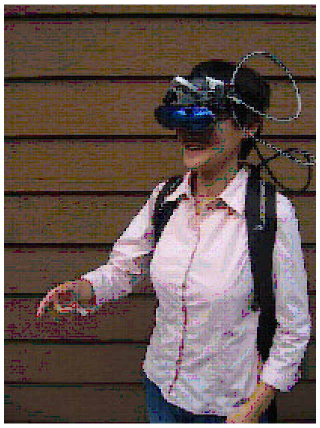


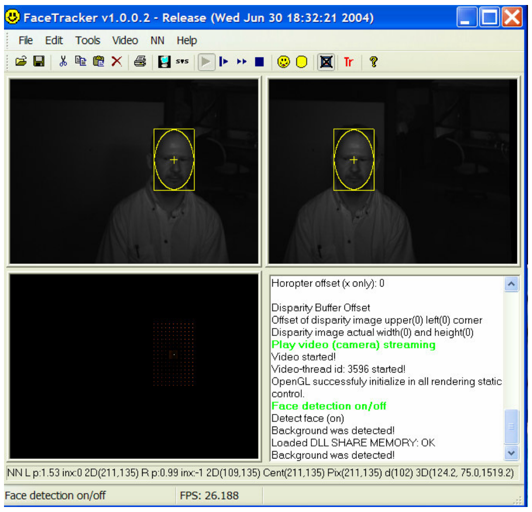
Figure 6 HandVu by University of California at Santa Babara

VR Face Tracker

Girado at the University of Illinois at Chicago designed and implemented a VR face tracker [34] using neural network based techniques. By using dual mounted cameras and active IR illumination, the tracker is capable to detect and track human faces in 3D space at interactive rate and acceptable latency. Girado's tracker firstly trains the face model as a grey-scale blob using neural network, and then detect this trained blob in the captured image to detect the head. Disparity in two camera images is used to calculate depth of the face, and in turn determine its 3D space location.

The use of neural network for face tracking is both an advantage and a limitation. For the pros, neural networks are intuitive to train because they are pixel based and rather than feature based; the classification process is embedded in the neurons and thus very generic instances can be classified. The cons are that because this embedded classification process, tracking of articulated objects such as the human hands is virtually impossible to be fulfilled by neural network. Girado's system is basically tracking a "face-like" grayscale blob in the captured image, as human hands are highly deformable, using a similar approach for hands will be very sensitive to background noise.

Scalability is also a key issue that restricts the performance of Girado's system. Because the whole image processing needs to be performed at satisfying frame rate and latency, several important components of the image processing are forced to be simplified. For example, scaling of the captured image to match the pre-trained template is limited to be at only 3 levels. This restricts the freedom of movement of the user's head and hinders the system's mobility if it is ported for mobile device use. Because the processing time is a critical factor for the tracker, recognition is an expensive task to perform. This is acceptable for face tracking but not for hand tracking because gesture recognition is a highly preferred feature when a hand tracker is deployed.


FaceTracker v1.0.0.2 - Release (Wed Jun 30 18:32:21 2004)
File Edit Tools Video NN Help

```
Horopter offset (x only): 0

Disparity Buffer Offset
Offset of disparity image upper(0) left(0) corner
Disparity image actual width(0) and height(0)
Play video (camera) streaming
Video started!
Video-thread id: 3596 started!
OpenGL successfuly initialize in all rendering static
control.
Face detection on/off
Detect face (on)
Background was detected!
Loaded DLL SHARE MEMORY: OK
Background was detected!
NN L p:1.53 inx:0 2D(211,135) R p:0.99 inx:-1 2D(109,135) Cent(211,135) Pix(211,135) d(102) 3D(124.2, 75.0,1519.2)
Face detection on/off          FPS: 26.188
```

Figure 7 VR Face Tracker by University of Illinois at Chicago

Based on reviewing of the related projects, Table 1 presents a summarized feature chart between them and the proposed research.

Table 1 A Comparison Chart of Proposed Solution and Related Projects

	Mobility	Body Part	Tracking Dimension and Device	Personalization	Interaction	Human-factor study	Use of Infrastructure Computation Resource
Quek	No	Hand	2D: Mounted Camera	No	Fingertip	None	No
Lok	No	Hand	3D: Mounted and Wearable Cameras	No	Hand Convex Hull	None	No
Starner	Yes	Hand	No Tracking, Interaction Only	No	Gesture	None	No
Kurata	Yes	Hand	2D: Wearable Camera	No	Gesture	None	Yes Desktop System
Kolsch	Yes	Hand	2D: Wearable Camera	No	Gesture	Posture Comfort	No
Girado	No	Face	3D: Mounted Camera	Yes, face look recognition	No Interaction, Tracking Only	None	No
Proposed	Yes	Hand	2D: Mobile and Mounted Cameras	Yes, skin color and hand shape profile construction	Gesture	Size Constancy	Yes Computer Cluster

3. VR HAND TRAINER: A PILOT STUDY

This chapter presents a preliminary work in which a hand-based interaction between users and the personal VE is used for rehabilitation. This work is conducted in collaboration with field experts in rehabilitation engineering and produces a working system that is currently deployed in one of the nation's top rank rehabilitation hospitals ([35], [36]). The personal aspect of this work is that it features a wearable system equipped with HMD and EMG triggered control device to be activated only by the wearer's muscle contraction attempts. Also, the user hand is tracked and acts as the mouse cursor in the VE, with a magnetic 3D sensor. The VE, which is used for reach-and-grasp training for stroke survivors, has the advantage over current hand rehabilitation devices, in its relative low cost and small size and potential to be used at home. The networked feature of the VE also allows application in tele-rehabilitation.

This work also provides a real-life motivation for the development of the final approaches of this work. It reveals the gap between the needed features for personal profile construction, tetherless interaction and scalable computing, and the limitations in present day's mainstream VE systems.

3.1. Introduction

Stroke is among the leading causes of adult disability in the United States [8] [9] [12]. Chronic impairment of the upper extremity occurs in roughly one-third of all stroke survivors [12]. While finger flexion often appears spontaneously within weeks after a cerebrovascular accident, finger extension is less likely to exhibit recovery [16] and creates difficulty for voluntary hand opening. The resulting distal limb impairment is especially problematic, since proper hand function is crucial to carrying out activities of daily living. A study from the UK reported that over half of the subjects studied depended on others for assistance in ADLs six months post-stroke [14].

Thus, a great need for hand rehabilitation therapies exists. None of the current therapies, however, has been wholly successful. For example, the effectiveness of electrical stimulation may be reduced by hypertonia. Usage of Botulinum toxin [13] further weakens already paretic muscles. Participation in constraint-induced training [19] requires some initial voluntary extension, thereby limiting eligible stroke survivors.

A combination of two different technologies, however, may be beneficial. The first is virtual reality (VR), which is able to present pre-set or online computed rehabilitation tasks with minimized setup and breakdown time. VR also provides many important possibilities that are not possible in real-world applications. For example, with precise hand position tracking and kinetic calculations, a stroke survivor user ("user" for short in the text hereafter) can manipulate virtual objects that are free of mass but can still provide force feedback. The second technology entails assistive devices. Research has already shown that devices which permit the active production of repetitive movements are helpful for arm rehabilitation after stroke. Therapeutic straight-line reaching assisted by the ARM Guide [15] resulted in improved active range of motion and peak velocity. In another experiment, assisted unilateral training with a PUMA robot led to increased Fugl-Meyer Motor Assessment scores [17].

Similar results may be achievable with the hand. As far as I know two systems which can readily provide assistance to finger extension in coordination with VR: the Rutgers-II ND Hand Master haptic device [21] and the CyberGrasp glove (Immersion Inc.) [22]. There are some drawbacks with each. The Rutgers-II ND is a point-based system, with it, the visual feedback is provided to the user through VR displayed on a non-stereo desktop monitor (“Fish Tank”). Thus, the user is unable to see his real hand together with the virtual scene. Additionally, the size of the virtual display is quite limited. Due to the use of pneumatic pistons residing within the palmar space, the maximal PIP flexion angle the glove allows is 45º, thereby limiting grasp simulation. CyberGrasp is designed more for haptic application purposes; its price and weight (over 500g) are relatively prohibitive for clinical use.

Thus, I have developed a training environment that integrates augmented reality (AR) and assistive devices. This environment addresses the limitations of Rutgers-II ND and CyberGrasp. AR allows the user to move objects with no mass while seeing his own hand overlaid with the virtual scene simultaneously. This experience suggests that see-through AR is much less disorienting to stroke survivors than fully immersive VR. Also, by incorporating head-tracking and stereoscopy, the virtual scene is made panoramic rather than flat, as that of Fish Tank VR. Assistance for finger extension is provided through either a body-powered orthosis (BPO), with cables acting on the dorsal side of the hand to pull the fingers, or a pneumatic-powered device (PPD) with an air bladder on the palmar side of the hand to push the fingers into extension. The two assistive devices share some common favorable characteristics: the pieces attached to the hand are lightweight (less than 100g) gloves; they work with AR in a coordinated manner; assistance is provided in accordance with a user’s voluntary attempt and under the ultimate monitoring and control of the therapist. This design diminishes the potential for excessive assistance. Lastly, the monitoring/control interface presented to the therapist incorporates visual, audio and force feedback using commercial hardware.

In pilot experiments, two stoke survivors participated in training under AR-with-BPO and AR-with-PPD conditions, respectively. Another stroke survivor, acting as a control subject, was trained with AR but no device assistance was provided. While the control subject showed little post-training improvement, both subjects under the integrated environment showed some signs of quantitative and qualitative improvements in hand function.

3.2. <u>Methodology</u>

3.2.1. <u>Overview of The Training Environment</u>

In the environment, the user is seated, wearing both head mounted display (HMD) goggles and either the BPO or PPD. The HMD shows 3D stereo virtual objects and contextual environment. The user is then trained to perform reach-and-grasp tasks of virtual objects. Dynamic assistance of finger extension is provided through the assistive device. For BPO, the assistance is controlled by the voluntary movement of the user's unaffected arm; for PPD, assistance is controlled by a combination of electromyography (EMG) signal along with the difference between present hand opening angle and desired hand opening angle. A therapist, who can be either on-site with the user or watching off-site through a video camera feed, supervises the user's movement. The therapist can modify the virtual scene dynamically to best meet the needs of the user. An example on-site setup with BPO is shown in Figure 8.

The experiment environment is made up of four main components: AR element, the BPO/PPD element, therapist monitor/control element and a networking element interfacing the therapist side and the user side.

3.2.2. <u>The AR Element</u>

Individual VR applications utilize one of four display strategies: HMD, augmented display, Fish Tank and projection-based display. The user environment uses an HMD display, namely, a SONY PLM-S700 Glasstron. The Glasstron provides a horizontal view angle of 28º, simulates a virtual 30” screen at 1.2 meters away from the viewer, and has adjustable see-through using an LCD shutter system. It is lightweight (120g for head device) and can be worn comfortably by the user. By adjusting the see-through level, the amount of the actual environment visible through the goggles is altered. This allows the user to see his hand along with the virtual object.

The scene, as shown in Figure 9, shows the surroundings as well as the object to grasp. Proper perception of depth and object size is achieved by both rich visual cues (e.g., table, floor, stationary objects) and field stereo. Objects are specially designed to have certain sizes and shapes. These instruct the user as to the proper hand posture and opening width needed for grasping. Also, objects can only be grasped when the user’s hand contacts the virtual object’s surface at “hotspots”. Hotspots are points predefined on the object’s surface, at the location of normal grasping. They are invisible, so the constraint they introduce is implicit to the user.

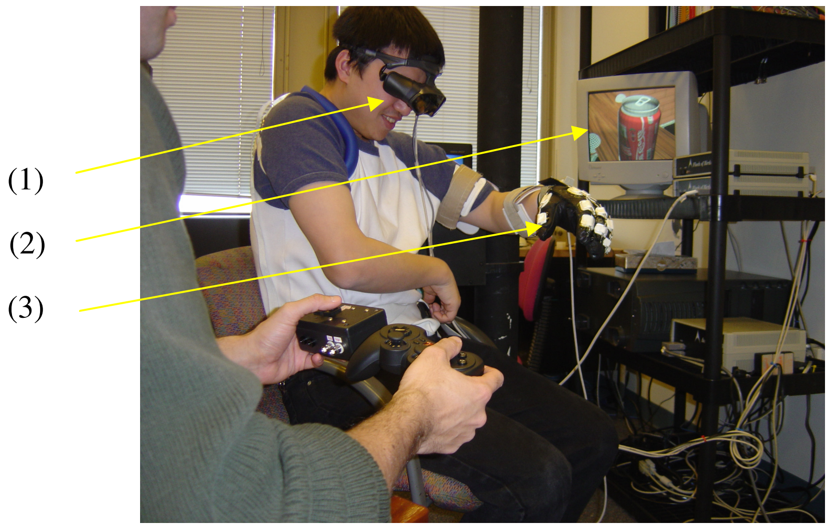


Figure 8 On-site Setup of the Training Environment Using BPO. The Therapist Is Holding Both the Joystick and Control Switch in His Hands. (1): HMD, (2): Fish Tank, (3): BPO

Several software packages are used for building the AR element. The Coin3D [Systems In Motion] library implements scene graphs, and it provides a comprehensive range of graphics and interactive objects. The CAVE Library [VRCO, Inc.] manages display parameters to establish the sense of depth and scale. The Trackd tool [VRCO, Inc.] reads the magnetic head and hand trackers' [Flock of Birds, Ascension Tech] positions and orientations, and provides these data to the rendering thread transparently.

VR objects persist on hard disk in VRML format and map one-to-one to files. I implement two levels of object management to achieve scalability and flexibility. The first level is "object library". Each folder that contains object files is scanned and an XML-format index file is generated for the folder. The index file contains entries of each object's size, location, hotspot numbers, and other information like suggested hand opening width. In this step, a "sanity check" for the files is also done to ensure the index contains only valid objects. The second level is "library view", which integrates all the libraries so that all objects appears to be in one large repository, thus the details of individual libraries are hidden and dynamic remapping is possible. Another functionality the second level provides, as its name suggests, is that the therapist can use his own definition file to create a local "view" of the whole repository. These definition files are plain text format and need only contain object names.

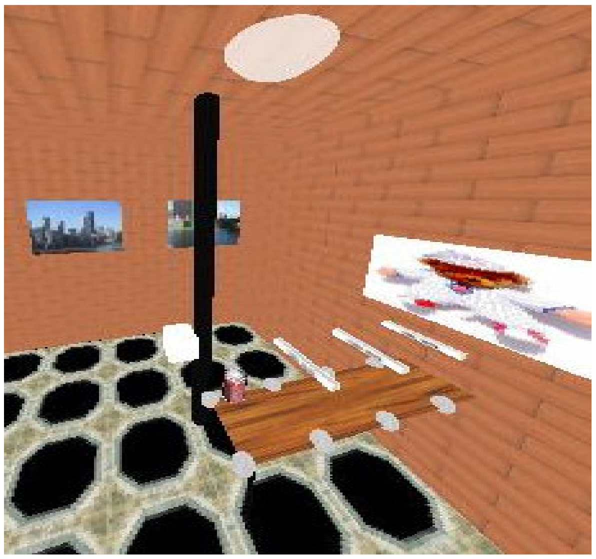


Figure 9 Overview of Virtual Object and Surroundings Displayed in HMD: Training Room, Coke Can Virtual Object and Markers to Provide 3D Cues

3.2.3. <u>The Assistive Device Element</u>

The BPO

The BPO, as shown in Figure 10, is based on prosthetics technology. A glove covers the paretic hand, and cables from the glove travel up to a standard figure-of-8 shoulder harness through metal cable housing. The cables actuate the finger joints. Namely, biscapular abduction and glenohumeral flexion pull on the cables, thereby forcing the fingers to extend. This single control moves all fingers simultaneously in a manner akin to that of control of the prehensor in arm prostheses. Alternatively, the cable can be run to a handle held by the unimpaired hand; extension of the unimpaired arm extends the fingers on the impaired side. In either manner, the user controls the amount of assistance provided to finger extension. The cable housing over the MCP and PIP joints also serves to prevent hyperextension of these joints.

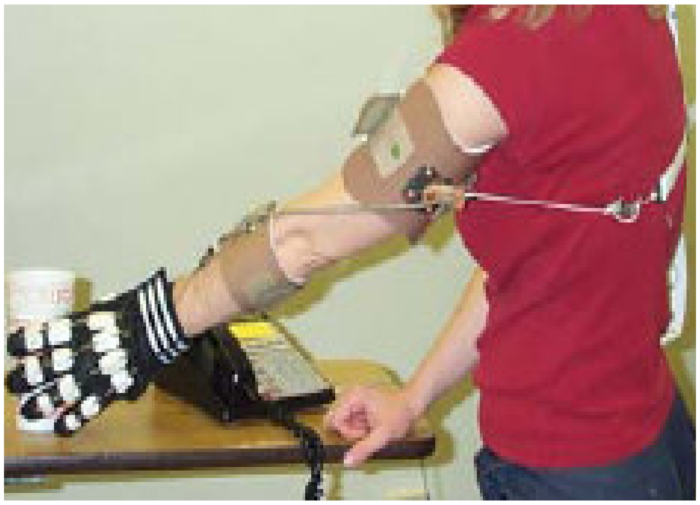


Figure 10 The BPO. A Zipper Sewn into the Palmar Side of the Glove Facilitates Donning.

The orthosis is light (450g) and easy to wear. The part of the device that directly acts on the impaired hand resides entirely on the dorsal surface so there is no interference with palmar grasp. Finger movement space is also maximized (90º PIP flexion angle). The amount of assistance utilized to extend the fingers is quantified by an in-line force sensor [Sensotec Inc.]. The sensor, spliced into the cable between the cuff and harness, detects the amount of force in the cable; this force serves as an estimate of the degree of assistance provided. Force is also encoded into sound pitch to provide audio feedback for the subject, as well as being sampled and stored for subsequent analysis.

One practical issue for body-powered therapy is that when the user becomes familiar with the device, they tend to overly rely on the device rather than using their own hand. This design addresses this issue by two means: one, as mentioned, is through the cable housings, proper adjustment of their lengths can regulate cables' maximum free movement distance, thus limit maximum assistances that could be provided; the other is through force feedback to the therapist (see section D), when excessive assistance is noticed by the therapist, he/she will give proper instructions to prevent user from doing so.

The PPD

The PPD as shown in Figure 11 is a polyester glove placed on the subject's hand. The glove contains an air bladder situated on the inner surface of the glove such that it contacts the palmar surface of the hand. Inflation of the air bladder forces straightening of the palmar surface, and consequently extends the fingers. The bladder is connected through a servo valve [Pressure Control Valve, QB02005, Proportion-Air] to a pressure reservoir [1104360, Jun-Air]. The servo valve allows pressures between 0-5 PSI to inflate the glove. Another port on the bladder is connected to a pressure relief valve [check valve w/ 6.1 PSI spring, 246301000, Halkey-Roberts] that opens at 6.1 PSI to avoid over-inflation. The electro-goniometers are attached with Velcro to the velar surface of the glove over the PIP and MCP joints.

Angle measurements from the proximal interphalangeal (PIP) joint of the index finger and metacarpophalangeal (MCP) joint of the middle finger are recorded using electro-goniometers [F35, Biometrics]. Muscle activity is recorded using active surface EMG electrodes [Delsys Inc.]. Electrodes are placed on the flexor digitorum superficialis (FDS) and extensor digitorum communis (EDC) muscles of the gloved arm to sample muscle activity. Each EMG signal is passed through the DelSys amplifier, full-wave rectified, and low-pass filtered before sampling.

Feedback control of the PPD uses an EMG signal and PIP/MCP angles as input and air pressure as output. Actual joint angles are compared with the desired trajectories necessary for opening the hand sufficiently to grasp the object. These desired finger trajectories are derived from the stereotypical spiral trajectories (Equation 1) observed in a study by Kamper et al. [23] examining fingertip trajectories during grasp in neurologically healthy subjects. The spiral trajectory may be expressed in Cartesian coordinates (Equation 2). With the addition of a constraint relating DIP angle to PIP angle, inverse kinematics may be used to translate fingertip location into MCP, PIP and DIP joint angles.

$$\begin{aligned} r &= A * (\exp(\theta * \cos(b) / \sin(b))) \\ A &= 1.3394 * (ld + lm + lp) - 23.255 \\ b &= 1.633 \end{aligned}$$

r and θ represent the position in polar coordinates where r represents the distance from the origin and θ represents the angle or rotation. ld, lm, and lp represent lengths of the distal, middle and proximal phalange respectively measured on the index finger of each subject. Units are in millimeters.

Equation 1 Stereotypical Spiral Trajectories

The equations used to represent the finger end-point in terms of joint angles are:

$$\begin{aligned} x &= ld * \sin(\theta_1 + \theta_2 + \theta_3) + lm * \sin(\theta_1 + \theta_2) + lp * \sin(\theta_1) \\ y &= ld * \cos(\theta_1 + \theta_2 + \theta_3) + lm * \cos(\theta_1 + \theta_2) + lp * \cos(\theta_1) \\ \theta_3 &= 0.7\theta_2 \end{aligned}$$

θ_1, θ_2, and θ_3 are MCP, PIP, and DIP joints respectively. The locations of x and y are shown in Figure 12. The origin is at the center of the MCP joint.

Equation 2 2D Finger Endpoint Calculation with Joint Angles and Finger Segment Lengths

A computer controlled proportional-derivative controller regulates the pressure necessary to maintain the required angle for both the PIP and MCP joints during reaching. The control regulates pressure to the glove based on the greatest angular flexion error. When the fingers are extended further then the set-point, pressure to the glove is reduced to maintain the necessary joint angles.

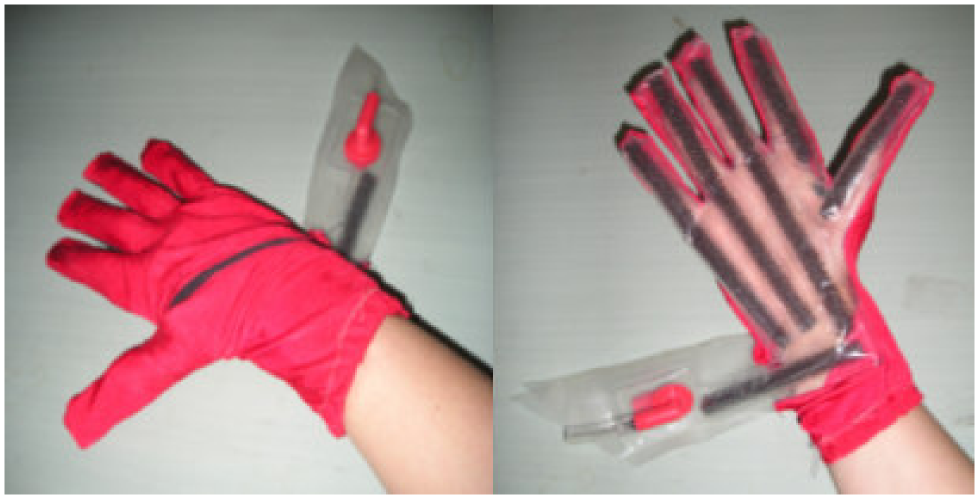


Figure 11 Picture of the Glove that Contains the Bladder on the Palm of the Hand

EMG feedback is incorporated to ensure active participation of the user. The system senses muscle activity through the electrodes; air pressure is only provided to assist extension when EDC activity exceeds a predetermined threshold.

Two different control strategies are employed for the grasp portion of the grasp-and-release training dependent on whether virtual or actual objects are used. When virtual objects are displayed to the user, the system monitors the point at which the hand is sufficiently extended to hold the object. When this is true, a signal is sent to the AR element to allow grasp of the object. The object is then attached to the user's hand when the hand is properly positioned in space over the displayed object. When the virtual object is held, the glove control system continues to regulate pressure to maintain the desired joint angles in order to simulate holding a real object. When real objects are displayed to the user, the therapist is responsible for determining when the hand is in position to grasp the object. This is detailed in section D.

The release portion of the therapy session is accomplished by monitoring EDC activity. A threshold based on the subject's maximum recorded EDC activity is set. When the object is held, and activity greater then this threshold is recorded from the EDC muscle, a pressure of 5 PSI is used to inflate the glove in order to assist the subject in object release.

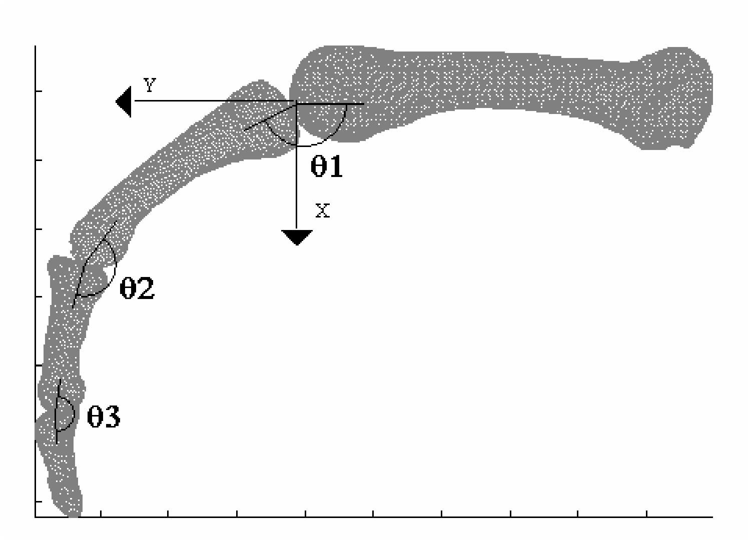


Figure 12 Image of Finger Representing Location of (x, y) Origin

3.2.4. <u>The Therapist Monitor/Control Element</u>

The therapist-side element serves two functions: monitoring and control. During training sessions, the user's hand movement is supervised by the therapist. This can be done by either the therapist staying on-site with the user, or watching through a camera link. Under both circumstances, the therapist is also shown the exact scene that the user views, but in Fish Tank display. This display for the therapist is especially useful when the user has problems with distance and depth perception, as the therapist can guide the user. When the therapist determines that the user's hand is sufficiently opened (dependent on impairment level of the hand and the current task), she/he flips a switch to set the hand state to be "ready", which means that the user's hand is in the correct posture to grasp the object once the hand reaches the proper location in space, as determined by the hand tracker. Once the hand contacts a hot spot on the object, the object now moves with the user's hand. After manipulation of the object, the therapist instructs the user to let go of the object. When the therapist determines that the hand has been sufficiently opened, she triggers "release" of the virtual object with the toggle switch.

A Logitech RumblePad2 force feedback joystick is used by the therapist to dynamically control the virtual scene. Online modifiable parameters of the virtual scene are the position and orientation of the object in 3D space, as well as its size. This makes configuration of the environment convenient as no thorough pre-calculations are needed for these parameters.

The therapist is provided with dynamic feedback of subject performance. For BPO, the assistive force recorded by the in-line sensor is displayed as a running waveform on a computer screen, in addition to the audio feedback. For PPD, the waveforms are for EMG, MCP/PCP angles and air pressure, companied by audio prompt for air pressure as well. Under both BPO and PPD, it is possible to encode for the assistance provided to extend the impaired hand by providing force feedback to the therapist through the joystick. The force magnitude is represented by the intensity of joystick vibration.

3.2.5. Therapist Side and User Side Communication

Successful coordination of the user-side element and therapist-side element requires inter-communication between them. Three kinds of data comprise the traffic stream: 1) force sensor data (for BPO) or EMG/angle difference/servo pump control combination data (for PPD) from user side to therapist side, with bandwidth consumption of about 10kbps; 2) head and hand tracker positions and orientations, from user side to therapist side; bandwidth consumption is also about 10kbps; 3) control commands issued by the therapist, from therapist side to user side; this traffic is random (every one or more seconds) and has negligible bandwidth consumption. To meet the need for tele-rehabilitation, the bandwidth and response time requirements must be able to be satisfied by the network. The environment's overall bandwidth requirement is about 20-30kbps, and response requirement is about 8-10ms each way (to meet the 100Hz sampling/control rate). These are all within the capability of today's broadband network services: LAN, DSL and T1.

3.3. Preliminary Experiment

Three male stroke survivors, rated between stages 2-3 of the Stage of Hand portion of the Chedoke-McMaster Stroke Assessment scale [20] respectively, participated in training sessions using the environment for 6 weeks. One of them was using AR with BPO, one was using AR with PPD and one was using only AR with no assistive device provided. The 30-minute training sessions were held three times per week. In each session, the subject tried to grasp 15 virtual objects followed by 15 real objects. The therapy was performed on-site.

3.4. Results

Subjects underwent standard functional tests, i.e. box & blocks [18] and Rancho [11], before and after the six-week training sessions. Both tests map better performance to higher scores. Table 2 indicates that the subject undergoing AR-with-BPO slightly improved scores on both box & blocks and Rancho; the subject undergoing AR-with-PPD slightly improved the Rancho score, but actually performed worse on the box & blocks; the subject undergoing AR-only showed no change in these scores.

Table 2 Pre- and Post-training Functional Test Results for Three Stroke Survivors

Treatment Used	Chedoke Level	Box&Blocks		Rancho	
		Pre	Post	Pre	Post
AR w/BPO	2	1	4	5	6
AR w/PPD	3	3	1	3	4
AR only	2	0	0	4	4

Besides the functional tests, I used a custom-developed apparatus to further assess the change of voluntary MCP extension. Speed and maximum displacement were measured for voluntary extension against no load. A servomotor system, described in [13], maintained zero-load through servo-control of the motor about zero torque.

The test results are shown in Table 3. The subject undergoing AR-with-PPD showed some improvement in both peak angular speed and angular displacement toward extension. The subject undergoing AR-only exhibited increase in peak velocity, but the amount of extension was so small as to render it of little functional consequence. The subject undergoing AR-with-BPO showed no change in either measure.

Table 3 Pre- and Post-training Biomechanical Metrics for Voluntary Extension

Treatment Used	Peak Angular Velocity (degree/second)		MCP Maximum Displacement (degrees)	
	Pre	Post	Pre	Post
AR w/BPO	11.055	11.21	9.3388	10.5125
AR w/PPD	115.94	124.19	39.066	45.996
AR only	12.924	31.327	2.3323	4.7353

An analysis was also performed on the assistive force data collected from BPO over time. Figure 13 shows the normalized force during each training session. Assistive force first increased largely from session 4 to session 6. This increase may have arisen from greater patient familiarity with the orthosis which allowed the patient to make greater use of it. Starting from session 6, the needed assistive force started to decrease, revealing a significant descending slope (p = 0.03). The overall decrease is 14.5% from pre- to post-training.

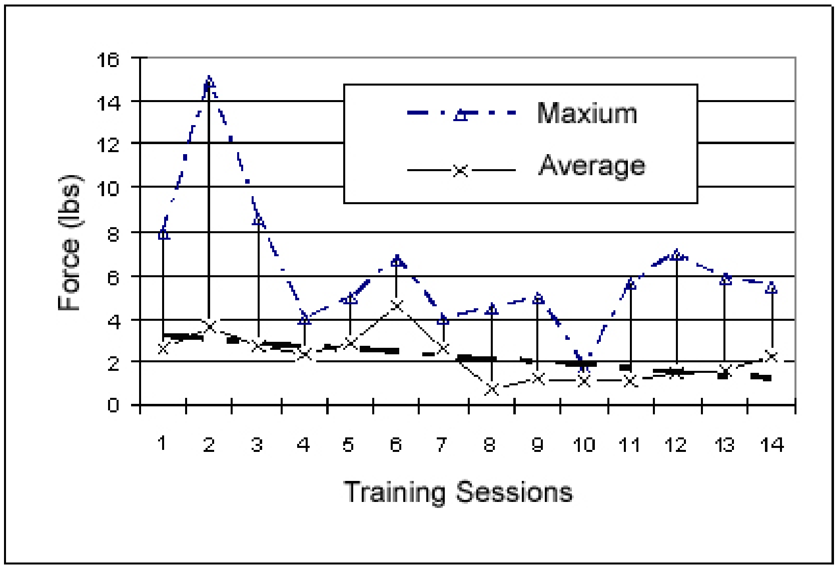


Figure 13 Assistive Forces Recorded During Each Training Session for AR-with-BPO, with Fitted Trend Line

3.5. <u>Discussion And Conclusions</u>

In this chapter, a training environment for rehabilitation of hand opening in stroke survivors is presented. This environment integrates augmented reality, assistive devices and the process of repetitive training of grasp-and-release tasks. Compared with current hand rehabilitation robotic devices, it is relatively low-cost and small in size, thus has the potential for use in clinics and even at home. The networked feature also allows application in tele-rehabilitation.

The preliminary experimental results, functional tests scores, peak angular MCP extension speed, MCP maximal displacement, and BPO assistive force, show that after 6 weeks of training, there was an encouraging trend of modest improvement of finger extension capability in the impaired hand. Both the user and therapist reported that the environment was user friendly due to the lightness of the assistive devices and the simple steps needed for set up of the environment. I believe that therapies using this environment are promising. Being part of an interdisciplinary research, the training system has been deployed in one of the nation's top ranked rehabilitation hospitals, and is now in daily use to train stroke survivors.

The lessons learned from this work, for a computer scientist, are three-folds: the first lesson is that this work shows that advanced visualization application, such as the VE used in VR hand trainer, can be massively deployed for personal users. The stroke survivor population is a good example of the potential beneficiaries. According to the data from American Stoke Association (ASA) in 2005, there were 4.8 millions of stroke survivors and the population was increasing at a 700,000/year rate. Even if VR trainers are used by 10% of the stroke survivors, the number of deployed systems will outnumber all the VEs deployed so far in universities and industries.

The second lesson is that understanding of human factors is an indispensable part of in-depth research in the computer science field of virtual reality. Users of the VR Hand Trainer take advantage of the VEs to simulate real life. The ultimate purpose of the usage is not to gain familiarity of the computer itself, but to gain from the transfer of VE to physical world. This is quite different from many other fields of computer science research, where the computer users explicitly perceive the presence of the computer entity, and the purpose is to better use a “computing machine” rather than body parts of the user himself. Insufficient knowledge of the physical, physiological and cognitive human factors when the virtual reality technology is used will either impede wide adoption of the technology itself, or make the introduction of this technology less meaningful or even negative.

The third lesson is that efficient computation must be conducted in a mixed composition of the wearable system and the infrastructures to achieve satisfactorily transparent human computer interaction. In this work, a key element of the training environment is the tracking system, which continuously monitors the position of the user’s head and hand. Compared with the other human-monitoring components used in the system, such as the electrodes that monitors muscle contraction EMG or the bend sensors that measures hand openness, tetherness of the electromagnetic tracking system is obvious. A “smart” environment needs to be introduced and its intelligence heavily relies on the environmental computation power. In another words, more natural interaction needs to be implemented through scalable computing techniques, for example, utility computing power provided in the rehabilitation setting.

VR Hand Trainer is a preliminary work, and reflects the spiral learning path when conducting research in visualization and interaction problems for the individuals. By working with and listening to domain experts, the three strong motivations of this research work emerged and in the following chapters, are addressed in detail.

4. SIZE CONSTANCY EXPERIMENTS IN THE CAVE

This chapter presents the work conducted to address one of the research questions identified by this thesis: what are the visual factors of a VE that affect users' performance in correct size perception in a VE? To answer this, a set of controlled experiments are designed and conducted over a 18-subject test population. Three VE visual factors are examined: scene complexity, stereovision and motion parallax. Results suggest that the first two exhibit statistical significance, while the last one does not in both of its forms, i.e. active motion parallax and passive motion parallax.

4.1. Introduction

VEs are used for a variety of research and commercial purposes, such as medical diagnosis, scientific data mining and industry manufacturing ([44], [35]). The effectiveness of VE in its applications relies heavily on its ability to create perceptions within the environment that faithfully replicate those in the physical world. However, due to limitations the VE can have a number of flaws that adversely affect its use and the credibility of the environments that it offers. One of the more significant aspects of this problem is whether the perceived size of an object in the VE is equivalent to that perceived in the physical world when object distance from the observer changes.

The recent work of Kenyon et al. [51] demonstrated size-constancy behavior in subjects using a CAVE. For a majority of their population monocular cues were needed to accompany the persistent stereovision of the object to fortify its true size. Without these monocular cues, a majority of their subjects failed to exhibit size-constancy and adopted a visual angle performance. Although subjects in their study could move their head or body during the experiments, which would have produced motion parallax, they did not do so. In this study I exposed subjects to both active and passive motion parallax conditions in addition to changes in scene complexity and stereovision. Results were similar to those performed in the physical world where size-constancy was more prevalent when a rich scene environment was used with stereovision. When the richness of environment was turned off and stereovision was removed most of the subjects adopted a visual angle performance. Results of these experiments also suggested that motion parallax, either created by the VE or the observers, had a mix effect on the perception of size constancy. Some subjects benefited from motion parallax while others showed no effects at all.

4.2. <u>Related Work</u>

Huber et al. [37] did experiments under the applied contexts of minimal access surgery (MAS) tasks, and studied the effects of stereoscopy and observer-produced motion parallax for distant judgment. Results indicated that stereoscopy confers a considerable performance advantage, while providing motion parallax information was not beneficial. Experiments by Beall et al. [38] where subjects judged the size of objects' whose visual dimension varied four-fold, concluded that absolute motion parallax only weakly determined the visual scale of nearby objects. Rondot et al. [39] studied distance perception during a tele-operation task. Their results suggested that stereoscopy and motion parallax were of equal significance in distance judgment, and users' performance varied largely between HMD and projected screen settings.

Additional studies showed inconsistent effect of motion parallax. Ikehara et al., [40] compared the results of different experimental methodologies for size-distance perception tests. Their results argued that for size and distance perception studies that used point light sources and rods could produce different results from each other, but these differences were not significant enough to change their conclusions. Watt et al. [41] raised the question of whether enhanced motion parallax, i.e. visually magnified motion parallax would alter the result found when using standard motion parallax stimuli. They found no significant effect of augmentation on motion parallax effect. Rosen et al. [42] showed that subjects made symmetry judgments in VE under different view conditions, and argued that motion parallax was not a significant factor in determining such capabilities. Effects of multi-modal interaction factors in determining size and distance perception were analyzed in Hirose et al. [43], and the authors emphasized the effectiveness of haptic interface in improving distance perception accuracy.

4.3. Methods

4.3.1. Subjects

Eighteen subjects were tested (EC1-EC18). Nine were experienced in VE and had a minimum of 6 months of using immersive VEs. For the other inexperienced subjects, this was their first exposure to an immersive VE. All subjects were tested for visual acuity and stereo acuity. All subjects had corrected vision of 20/20 and normal stereovision.

4.3.2. Apparatus

All tests were performed using a single wall CAVE – the C-Wall (Configurable Wall) [52]. The C-Wall is a high-quality, head-tracked, active stereo wall, that displays an image before the viewer by means of a 10x10ft rear-projection screen. The back projector pointed to a mirror, which reflected the images onto the screen. To create stereoscopic objects, two off–axis perspective images are consecutively displayed; one visible to the right eye, the next to the left eye. The visibility of images by each eye is controlled by the stereo glasses (Stereographics, Inc. Beverly Hills, CA) which rapidly turn each lens on and off in synchrony with the corresponding images on the screen. A Pentium IV PC created the images for the C-Wall. The image resolution was 1024x768 pixels with a refresh rate of 120 Hz and an update rate of 60 stereo images per second. Each subject's interpupillary distance (IPD) was measured (R.H. Burton Digital P.D. Meter, R.H. Burton LLC, Drive Grove City, OH) and incorporated into the CAVE program to generate the stereo images for each subject. A six-degrees-of-freedom camera tracking system (Eagle Digital System, Motion Analysis Corp., Santa Rosa, CA) provided real-time head position which was used to calculate the correct stereoscopic perspective projections for the C-Wall as the viewer moved his/her head. The head tracking system had a latency of 65 ms and was calibrated to an accuracy of ±0.1 inches for the tracking distances used in these experiments. A cordless joystick (RamPad, Logitech Inc., Fremont, CA) held by the viewer provided interaction with the VE.

A virtual Coke bottle textured with the image of a physical 2-liter Coke bottle was drawn to test size perception [51]. Characteristics of VE scene were manipulated in order to test the effects of scene complexity, motion parallax, and stereovision on perception of virtual object size.

Scene Complexity

Two scene environments were provided, either a rich environment (ENV), with monocular and stereo cues to depth in addition to those confined to the bottle in the scene or a sparse environment (No-ENV) with cues to depth confined to the bottle in the scene [51]. The ENV consisted of a gray-green checkered floor with a wooden textured table in the scene; the Coke bottle sat on top of the table. The table's height above the floor was randomly set at one of the three possible heights (30, 33 and 36 inches). For the No-ENV case, the environment consisted solely of a gray background. The virtual Coke bottle was presented as being suspended in mid air at different heights from the floor (corresponding to the table heights) and at a number of different distances from the user as described above. The head was tracked identically to that described above.

Stereovision

Two viewing conditions were examined: monocular vision (MONO) and stereovision (STEREO). For the STEREO condition, disparate images were presented to the two eyes. Interpupillary distance (IPD) was measured for each subject, and the images for the two eyes were created to reflect the different vantage points in order to evoke a stereo image. For the MONO condition, the IPD was set to zero in the CAVE program therefore the same image was presented to each eye. Consequently, the subjects continued to view the scene through the Stereographics headset producing the same visual conditions as when stereovision was present.

Motion Parallax

Three different motion parallax conditions were tested: no motion parallax (No-MP), motion parallax generated by the VE (Passive-MP), and motion parallax generated by the lateral movement of the viewer (Active-MP).

For the No-MP condition the subject was instructed to hold his/her head still and look straight ahead with no lateral head movement. To ensure the subject was not moving, the experimenter monitored the lateral head movements from the tracker, and prompted the subject whenever there were head movements greater than 1 inch, the minimum value needed to incur motion parallax.

For the Passive-MP condition, the whole scene displayed on the C-Wall moved in a sinusoidal fashion at 0.25 Hz. Peak scene displacement was 1 ft and peak velocity was 4 ft/sec. These parameter values were chosen to conform to natural human lateral movement in order to facilitate comparisons with active motion parallax ([38], [39]).

For the Active-MP condition the subject was instructed to move his/her head laterally from side to side at 0.25 Hz with a minimum displacement of 1 ft. The subject was provided with audio cues for proper movement frequency from an electronic metronome. The experimenter monitored lateral head movement through the tracker and prompted the subject whenever lateral movement amplitude fell below the desired level.

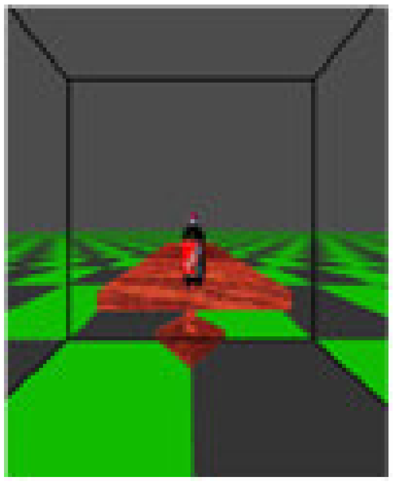


Figure 14 the Virtual Coke Bottle with Rich Scene Environment

4.3.3. <u>Experimental Protocol</u>

Subjects were instructed to adjust the size of the virtual object (2-liter Coke bottle) so that they perceived the virtual object's size as being identical to that of a physical Coke bottle if placed the same distance from the subject. To aid in this task, a physical 2-liter Coke bottle was visible to the subjects for comparison to the virtual object. The 2-liter Coke bottle was placed on a wooden stand covered with black cloth at a height of 3 ft. The stand was positioned at the front left side of the C-Wall at a distance of 3.5 ft. from the subject. Both the physical and virtual Coke bottles were 12 inches tall and 5.5 inches (maximum) wide. The physical Coke bottle, lit by a standing spotlight, was visible to the subjects by simply turning their head 40° to the left.

The virtual Coke bottle was displayed randomly at one of the five distances from the subject: 3.5, 5.0, 6.5, 8 and 9.5 ft. The subject sat 5 ft. from the C-Wall screen; thus, the virtual object could be located in front of, on, or behind the C-Wall screen. The computer randomly set the initial size of the virtual Coke bottle from 0.2 to 3.0 times the normal size (12 inches) of the bottle. Subjects used the cordless joystick to increase and decrease the size of the virtual Coke bottle to what they perceived to be the appropriate size for each trial. The head was tracked so the scene was updated appropriately to the position of the subject's head.

The independent variables of scene complexity, motion parallax, and stereovision had 2, 3, and 2 possible states, respectively. Each condition was repeated 6 times for each bottle location for a total of 360 repetitions. To avoid ambiguity hereafter, I call each repetition of size judgments that was performed under the same configuration of the independent variables a run, and the consecutive block of runs a trial. Additionally, subjects performed an initial trial to familiarize themselves with the process. It could be seen that except for the initial trial, trials and visual factor configurations mapped one-to-one to each other. Table 4 shows this mapping relationship between trial IDs and visual factor configurations.

Table 4 Mapping between Trial IDs and Visual Factor Configurations

Trial ID	Scene Complexity	Stereovision	Motion Parallax
T0	Initial trial for familiarization		
T1	No-ENV	MONO	No-MP
T2	No-ENV	MONO	Passive-MP
T3	No-ENV	MONO	Active-MP
T4	No-ENV	STEREO	No-MP
T5	No-ENV	STEREO	Passive-MP
T6	No-ENV	STEREO	Active-MP
T7	ENV	MONO	No-MP
T8	ENV	MONO	Passive-MP
T9	ENV	MONO	Active-MP
T10	ENV	STEREO	No-MP
T11	ENV	STEREO	Passive-MP
T12	ENV	STEREO	Active-MP

Subjects were encouraged to take 5 minute breaks between runs or as often as they needed to avoid fatigue. The total experiment time varied among subjects, from 45 to 60 minutes.

4.3.4. Data Analysis

Subject performance was evaluated quantitatively using several measures based on the selected size of the virtual bottle as described in [51]. In brief, SizeRatio represented the relative size of the virtual bottle compared to the proper size of the physical bottle:

$$SizeRatio = \frac{BottleSizeSetBySubject}{CorrectBottleSize}$$

Equation 3 Definition of Size Ratio

The numerator in Equation 3 corresponds to the size of the virtual bottle set by the subject in each run and the denominator was fixed at 12 inches (height of the physical 2-liter Coke bottle).

Linear regression of resulting SizeRatio values versus the distances of the virtual bottle from subject was then calculated. Since with projection-based VE everything is drawn on the CAVE wall, I calculated the visual angle (VA) setting that would result if subjects perceived their distance to the bottle as being the distance they were from the CAVE wall regardless of the bottle's virtual distance from the subject. If the subjects' performance is purely determined by visual angle, the SizeRatios will theoretically form a fixed slope, α, using the following formula:

$$\alpha = \frac{CorrectBottleSizeOnCAVEWall}{DistToCAVEWall}$$

Equation 4 the Visual Angle Constant in Size Constancy Experiment

In the experiment, α was fixed at 0.2 given a bottle size of 12 inches, and a distance between the subject and the CAVE wall of 5 ft. While SizeRatio measured subject's performance in a given run, the relationship between the regression slopes and α indicated the consistency of how well the subject performed across all the runs in a given trial. This percentage relationship between the subjects' SizeRatio data regression slopes to that of the predicted VA performance was calculated using the equation:

$$PercentVASlope = \left[\frac{FittedSlope}{\alpha}\right] * 100\%$$

Equation 5 Definition of PercentVASlope

For example, if the regression slopes of the subject's data were identical to α, then the "Percent VA slope" would be 100%, implying that the subject was showing no size-constancy. On the contrary, if the subject regression data showed perfect size-constancy, the regression slope would be zero and the "Percent VA slope" would consequently also be zero.

Absolute error for each run and mean absolute error across a trial were calculated as another indicator to examine the differences between ideal performance and the SizeRatio data collected from population. Absolute error indicates the deviation of a judgment in a run to actual virtual bottle size. Mean absolute error averaged absolute errors within a given trial. They were computed using the following equations:

$$AbsoluteError = \left|SizeRatio - 1\right|$$

Equation 6 Definition of Absolute Error

$$MeanAbsoluteError = \frac{1}{n}\sum_{1}^{n} AbsoluteError(i)$$

Equation 7 Definition of Mean Absolute Error

Percent VA slope and AbsoluteError were both derived from SizeRatio values and as aforementioned, described these values from two separate perspectives.

For the VA slope percentage, I did repeated measures analysis of variance (ANOVA) using SPSS (SPSS, Inc), with the independent variables to be the three visual factors: scene complexity, stereovision and motion parallax. The purpose of using ANOVA was to discover the significance of each visual factor in affecting size-constancy performance. While for AbsoluteError, I investigated its mean and distribution in each trial. Comparison of these indicators was to reveal that in which trials, i.e. under which visual factor configurations did subjects had better size-constancy performance.

4.4. Results

For the population, size-constancy performance, as measured by percent VA, was better when viewing the ENV conditions than the NO-ENV conditions (similar to [51]) and better under STEREO conditions than MONO conditions (both single-factor ANOVA). Furthermore, there were no significant interactions among these three visual factors. All other models that used interactions did not explain the data well and all had $p > 0.188$ or more.

4.4.1. Effect of Scene Complexity

Comparing the percent visual angle slopes (Equation 5) from the population, for the ENV vs. No-ENV trials that had the same motion parallax and stereovision conditions, (i.e. T1 vs. T7, T2 vs.T8, T3 vs.T9, T4 vs.T10, T5 vs.T11 and T6 vs.T12), showed that subject size-constancy performance was significantly better under the ENV conditions rather than the No-ENV conditions ($p < 0.0001$). The percent VA slopes obtained in the ENV conditions (20%±15) more closely matched the slopes expected with size-constancy whereas the slopes in the No-ENV viewing conditions (140%±20) more closely matched those associated with visual angle performance, as shown in Figure 15.

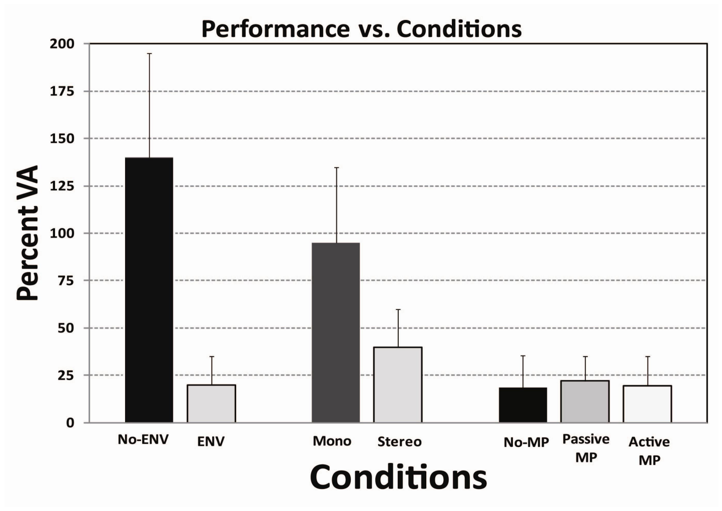


Figure 15 Percent VA Slope Means and Standard Deviations for Different Conditions, Motion Parallax (MP) for ENV Conditions Only

In addition, the ENV condition produced more consistent subject performance and the task was easier to perform according to subject reports. As seen in Figure 16, SizeRatio settings were consistently closer to 1 in ENV conditions than in No-ENV conditions for different bottle positions, especially for the bottles farther from the subject. In contrast, the mean SizeRatio for the No-ENV condition increased as the bottle positions receded from the subject and with a wider range of SizeRatio settings.

Improved performance of ENV over No-ENV can be seen with or without stereovision. With no stereovision (Figure 16 top), SizeRatio settings for the ENV condition ranged between 0.9-1.8 for the bottle distance of 3.5ft- 9.5ft from the subject, for the same visual conditions, No-ENV scene produced SizeRatio settings covered twice the range of the ENV data i.e., 0.62 – 2.46. With stereovision (Figure 16 lower) the SizeRatio settings under ENV condition spanned a smaller range from 0.96 – 1.53 compared to no-stereo. In No-ENV condition, the SizeRatio range was smaller i.e., 0.91 – 1.96, than its counterpart in the no-stereo case.

The absolute errors for size judgments made in all the six ENV and six No-ENV conditions for the population in Figure 17 shows a clear overall difference between ENV and No-ENV performances. The frequency distribution of absolute error for all judgments shows that 66.48% of the errors were 0.2 (or 2.4 inches if bottle height used) and below with the ENV condition while only 27.6% of the errors fell within this range with the No-ENV condition. The mean absolute error values calculated using Equation 7 were 0.53 for all six No-ENV conditions and 0.26 for all six ENV conditions.

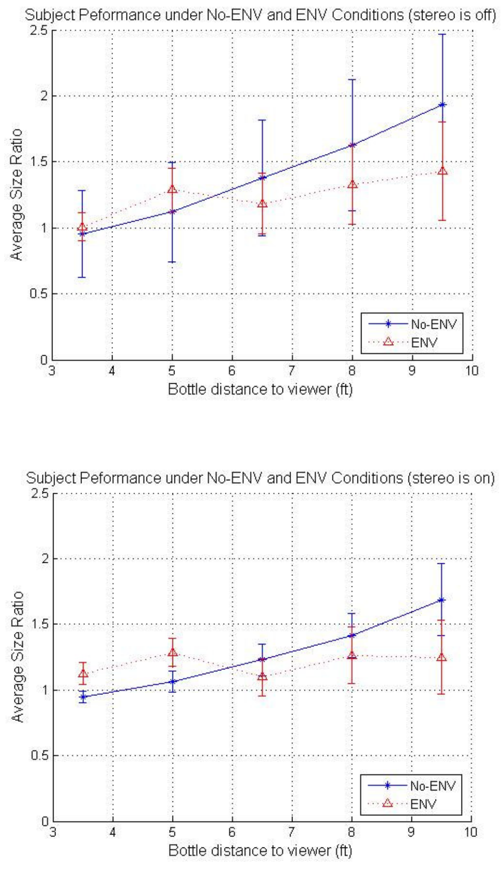


Figure 16 Population Performance with Stereovision (a) Off or (b) On, w/o Motion Parallax

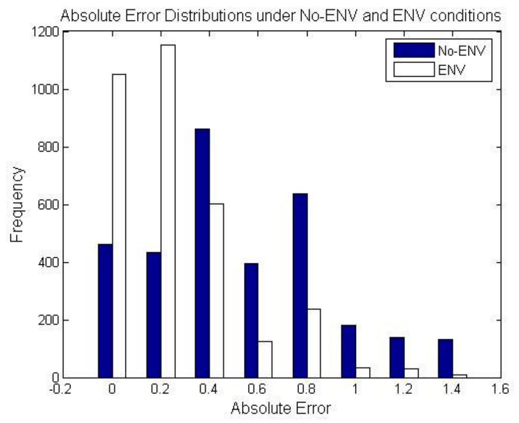


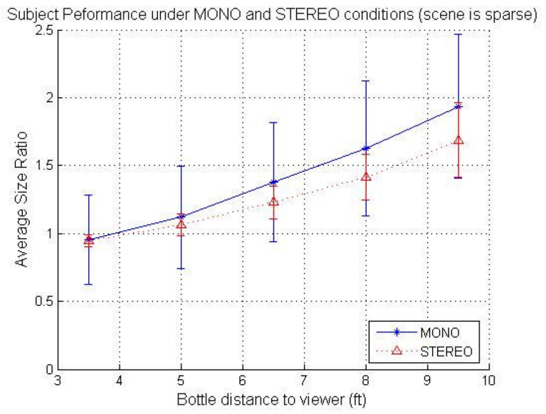
Figure 17 Absolute Error using No-ENV and ENV Conditions

4.4.2. <u>Effect of Stereovision</u>

Comparing the percent visual angle slopes from the population, for the STEREO vs. MONO trials that had the same scene complexity and motion parallax, (i.e., T1 vs.T4, T2 vs.T5, T3 vs.T6, T7 vs.T10, T8 vs.T11 and T9 vs.T12), showed that subject performance was significantly better in performing sizing task under the STEREO conditions rather than the MONO conditions. The VA slopes obtained in the STEREO conditions (40%±20) more closely matched the slopes expected with size-constancy and conversely the slopes in the MONO viewing conditions (95%±40) more closely matched those associated with visual angle performance, as shown in Figure 15.

This improved performance can be seen in Figure 18 where the mean SizeRatio for the MONO condition increased as the bottle's position receded from the subject. In contrast, for the STEREO condition although the mean SizeRatio also increased with bottle distance from viewer, it increased at a much lower rate. These observations were independent of the setting of scene complexity, a visual factor which has a significant effect as described above.

Under MONO conditions, subjects had a wider range of SizeRatio settings as well. The SizeRatio settings for the STEREO condition when scene was sparse in VE ranged between 0.91-1.96 for the bottle distance of 3.5ft- 9.5ft from the subject, for the MONO condition under same scene complexity configuration the SizeRatio settings ranged from 0.62 – 2.46. When scene was rich, the SizeRatio settings under STEREO condition ranged from 0.96 – 1.53. Under MONO condition, the SizeRatio ranged from 0.91 – 1.96.



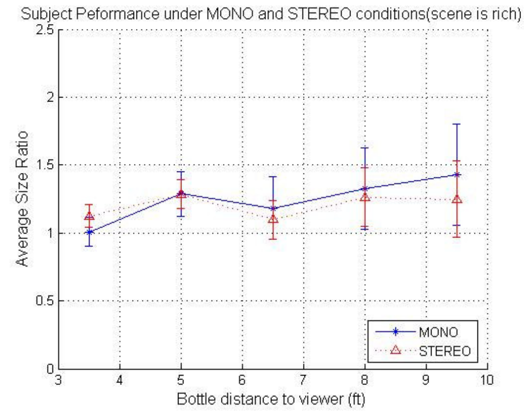


Figure 18 Population's average SizeRatio settings for trials T1 and T4, using sparse (a) or rich (b) scene, w/o motion parallax

The absolute errors for size judgments, Figure 19, used the six MONO and six STEREO conditions from the population. Examination of the absolute error for all judgments shows that 54.32% of the errors were 0.2 (or 2.4 inches if measured in the error of size judgment) and below with the STEREO condition while only 34.75% of the errors fell within this range with the MONO condition. The mean absolute error values calculated using Equation 7 were 0.46 for all six MONO conditions and 0.32 for all six STEREO conditions.

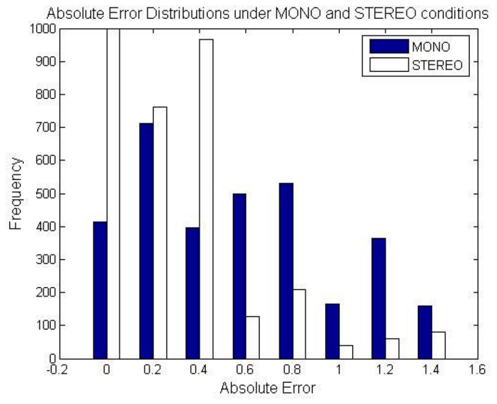


Figure 19 Absolute error value distributions under MONO and STEREO conditions

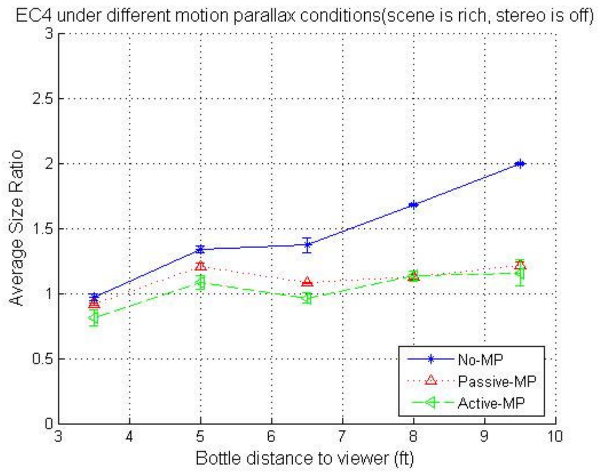
4.4.3. Effect of Motion Parallax

Comparing the different motion parallax conditions for trials that had the same scene complexity and stereovision, (i.e., T1, T2 and T3; T4, T5 and T6; T7, T8 and T9; T10, T11 and T12), showed no statistical difference under any of these conditions. The percent VA slope values for all three motion parallax settings overlapped in mean value, and standard deviations. These behaviors were found to be independent of the scene complexity and stereovision. When scene was sparse and stereovision was turned off, subjects showed a visual-angle performance. While when scene was rich and stereovision was turned on, they showed a uniform performance towards size-constancy, Figure 15. When scene was rich and stereovision was turned off, subjects' performance lay between the above two conditions.

There was no significant difference in the range of SizeRatio settings. When scene was sparse and stereovision was turned off in VE, range of SizeRatio settings under NO-MP was 0.62-2.46, under Passive-MP was 0.62-2.42 and under Active-MP was 0.63-2.53. When scene was rich and stereovision was turned off in VE, range of SizeRatio settings under NO-MP was 0.91-2.0, under Passive-MP was 0.9-1.8 and under Active-MP was 1.04-1.71. When scene was rich and stereovision was turned on in VE, range of SizeRatio settings under NO-MP was 0.96-1.53, under Passive-MP was 0.96-1.37 and under Active-MP was 1.01-1.35. This illustrates that the population's performance under the motion parallax conditions were not different from each other.

Although these result showed no significance analyzed as a population, I examined the performance of individual subjects under different MP conditions to better understand any effects on individual subjects. As stated above, the twelve triples could be grouped into four triples of trials, based on different settings of scene complexity and stereovision. I rank ordered them, based on a decreasing order of scene richness, as: ENV-STEREO, ENV-MONO, NOENV-STEREO and NOENV-MONO. Previous statistics told us that within each group, how the subject group performed. Here I am interested in that *across* groups, how *each* subject performed consistently among the three settings of motion parallax. Three notations are used to quantify the closeness of the same subject's performance shown in Table 5. For example, under two motion parallax settings M_1 and M_2, let S_1 and S_2 be the subject's percent visual angle slopes under M_1 and M_2 respectively: if S_1 is greater than S_2 by at least 10%, I denote M_1 performs worst than (<) M2; if S_1 is less than S_2 by at least 10% I denote M_1 performs better than (>) M2; under all other cases, I denote M_1 is the same as (=)

M_2. I further use the abbreviations A, P and N to represent Active-MP, Passive-MP and No-MP respectively.



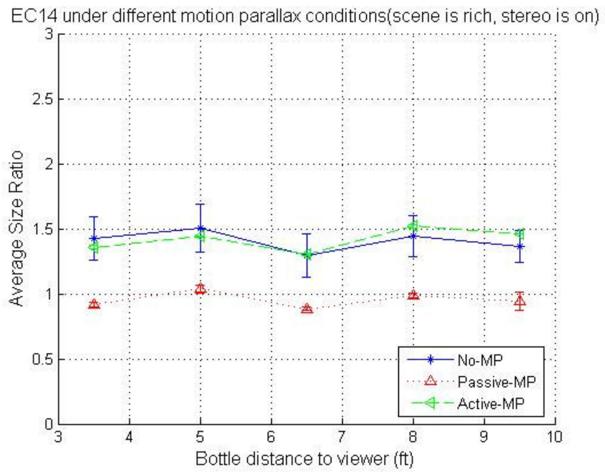


Figure 20 Individual SizeRatio Settings for Trials using a Rich Scene, without (a) and with (b) Stereo.

These results revealed that the eighteen subjects could be categorized into four groups, based on their consistency in size-constancy performance across the scene-richness groups. Eight subjects (EC1, 2, 5, 10, 13, 15, 16, and 18) exhibit no significant difference in size-constancy across all three motion parallax conditions, regardless of the variation in scene-richness. Four subjects (EC3, 4, 7 and 12) performed relatively better under Active-

MP settings than Passive-MP in certain triples. Three subjects (EC9, 11 and 14) performed relatively better under Passive-MP settings than Active-MP settings in certain triples. Two subjects (EC6 and 8) performed better both under Active-MP settings and Passive-MP settings. An instance of improved slope with MP is shown in Figure **20**a where there is a significant change in the slope with MP conditions.

There were some subjects where the slope did not give the entire picture of their performance. As shown in Figure **20**b, this subject shows the same slope for all conditions but the motion parallax condition shows an improvement in accuracy of the size setting behavior since this data is below the other 2 curves and hovers about a size ratio of 1.

Table 5 Individual Subjects' Performance across Scene Richness Groups, With Regard to Motion Parallax Settings

	ENV-STEREO	ENV-MONO	NOENV-STEREO	NOENV-MONO
EC1	same	same	same	same
EC2	same	same	same	same
EC3	A>N>P	A=N>P	same	same
EC4	A>N>P	N>A=P	same	same
EC5	same	same	same	same
EC6	P>N=A	P=N>A	same	same
EC7	A>N=P	same	same	same
EC8	P>A>N	A=P>N	same	same
EC9	P>N=A	same	same	same
EC10	same	same	same	same
EC11	P>N=A	same	same	same
EC12	A>N=P	same	same	same
EC13	same	same	same	same
EC14	same	same	N>A=P	P=N>A
EC15	same	same	same	same
EC16	same	same	same	same
EC17	A=P>N	same	same	same
EC18	same	same	same	same

4.5. Conclusions and Discussion

These experiments illuminate several important issues regarding size-constancy in projection based VE systems (the C-Wall is a CAVE variation). This work first verifies the findings of [51] that users can obtain useful size constancy performance in an immersive projection-based VE, at view distances and screen resolutions that represent mainstream VE systems (1-9 ft., 1024x768 pixels screen). This confirmation supports the need for complex scenes and monocular cues in addition to stereovision if wider deployment of VE system in size and distance perception sensitive applications, such as visual scientific data analysis and virtual metropolitan building planning are to be successful.

Unexpectedly, I found that motion parallax, produced by the VE or by the observer alone, might not be a significant factor in determining size-constancy performance for a given population. However, when I reexamined the data for individual subjects, I found that the effect of motion parallax in the experiment seems to vary from one subject to the next. As might be expected MP was dependent on the richness of the scene. The small amount of movement that occurs using a sparse scene was generally not sufficient to improve performance. The largest effect can be seen in the ENV:Stereo condition followed by the ENV:Mono condition. As seen in Figure **20**, I found that some subjects either improved their performance by changing the slope of their response (shallower) or produced more veridical bottle sizes (SizeRatio $\approx$ 1). Thus I can see that in some subjects the increase in the number or type of visual cue can produce an improvement in performance. Furthermore, there has been much published about the errors in distance judgments in VE. If I assume that size constancy has a measure of distance judgment included in its process then I might infer that in some cases that I might expect improved distance judgments with MP.

The results are similar to those in the physical world [50] that have shown that a subject's performance lies on continuum between size-constancy and visual-angle and that this performance is a function of the cues that are present in the scene. In Figure 21 I show where the subjects' performance lie given the cues presented[6]. This is reminiscent of the Figure 22 in [49] where they plot their subjects' performance as the field of view was narrowed and subjects moved from size-constancy to VA performance. One might expect a population of subjects' performance to follow the graph as the cues to depth are manipulated. This graph shows that the dominant condition for size constancy is a rich scene with stereovision (ENV:Stereo). That is followed by a rich scene and monocular condition (ENV:Mono). Notice that under sparse visual conditions that stereo (No-ENV:Stereo) modestly improves performance compared to monocular viewing (No-ENV:Mono).

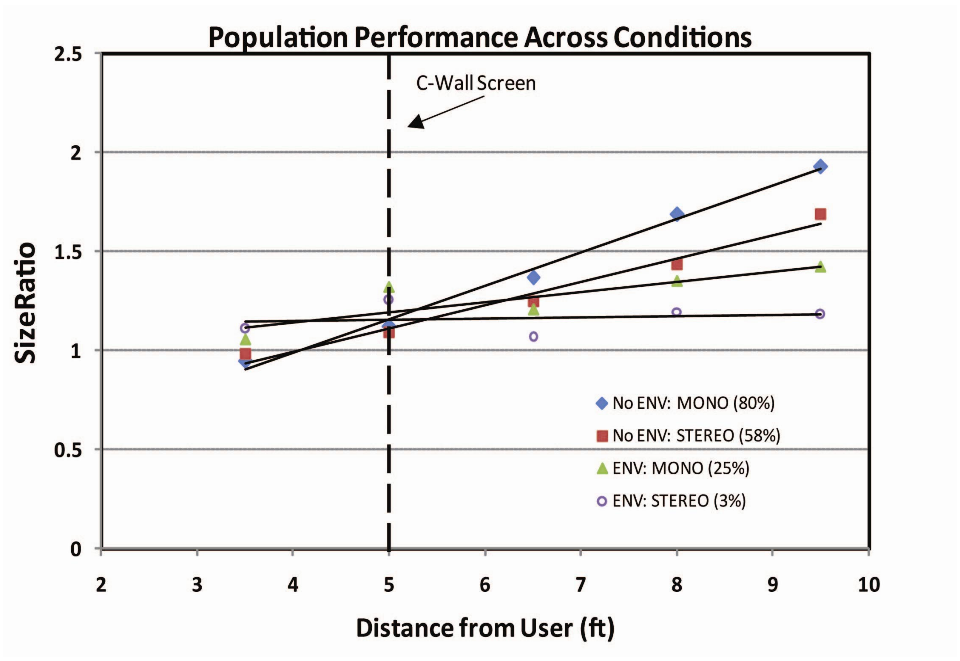


Figure 21 Averaged Fitted Slopes across Four Combinations of Scene Complexity and Stereovision Conditions with their associated percent VA slopes for each condition.

These conclusions could be helpful in decision making, for VR system designers who build the systems and for users who utilize the systems for specific applications.

[6] As motion parallax was not a significant factor in statistical test, I grouped all subjects' fitted visual angle into the categories No-ENV:MONO, No-ENV:STEREO, ENV:MONO and ENV:STEREO and averaged fitted slope values within each category.

It is worth mentioning that in the physical world 2D cues to depth are natural and straightforward. In fact, it takes effort to arrange a situation that would diminish these cues to the subject. In VE, displaying less complex scenes is easier than showing more complex ones. A VE that has numerous cues to depth (2D and stereovision) takes time to program and computer-time to generate. Thus, it is more expensive to generate a complex world compared to a sparse world in terms of cost, programming time, and display time. By understanding the relationships that exist between the physical and VEs will help us better utilize this extraordinary technology by supplying the most important information to the user.

In these experiments I only analyzed three major visual factors due to the hypothesis that they might be of most importance in determining size constancy performance. However with the enrichment of VE, multi-modal interaction between the user and VE is getting more popular and it could be interesting to examine the effect of other factors, e.g. display resolution, haptics, 3D audio etc.

5. DESIGN AND IMPLEMENTATION OF A HAND TRACKER

This chapter gives the technical details about the design and implementation of a computer program which does computer vision based hand tracking and posture recognition[7]. This work is a natural extension of and has the same principle as the personalized control system in Chapter 5. Extensions are reflected in two aspects: first, computer vision techniques are used to replace tethered measuring devices for hand movement; second, users are free to construct the personal hand profiles by themselves using mobile devices. As described in Chapter 1 of the thesis, the design requirements for the hand tracker are: efficient construction and incorporation of user's personal hand profile; robust detection of hand occurrence; real-time conversion of hand movements to 2-D coordinates; and lastly, high-accuracy mapping of posture to control commands.

Comparing with head tracking and face detection, hand tracking and posture recognition are more difficult problems. Unlike the human head, which could be abstracted as a elliptical blob and processed accordingly ([34, 57]), the human hand is highly deformable and difficult to be described by low dimensional parametric models. The hand tracker implemented in this work takes the approach of statistical learning for hand characterization. In other words, it learns from a database of "hand" and "non-hand" images what a hand should look like. In this work, the knowledge obtained by the hand tracker during the learning process are two statistical models: a histogram for hand color distribution and a decision-tree based classifier for hand shape identification. With the two learned

[7] If not specially explained, I use the short name "hand tracker" to state the hand tracker and posture recognizer implemented in the rest of this thesis.

statistical models, the hand tracker is able to make judgments about hand presence whenever a new frame comes in from video streams captured by a camera.

Based on the hand tracker design, its components are divided into two groups: an off-line preprocessing components group which collects training samples and builds statistical models of the human hand; and an interactive computing components group which processes the captured video stream from a camera frame by frame, detects/tracks the presence of a human hand, and recognizes the gestures performed if there are any. The main advantage of this hand tracker over related work is that the off-line preprocessing components group is mainly implemented on a mobile device, and makes use of motion sensors to control training sample quality. While in most related works, samples are to be collected from a public library or in strictly controlled laboratory environment. This improvement makes sample collection by the user themselves possible, and has the benefit of being able to incorporate individual biometric characteristics. In section 5.4, I present the evaluation results that reflect the advantages of the sample collection component.

To incorporate the personal profile of a specific user into the hand tracking algorithms, the first task is to identify what are the human factor invariants of a hand. High-level features of a hand include its relaxed shape, texture, posture configuration and skin color. Among these four high-level features, relaxed shape and posture are not invariants because they differ with different degrees of muscle contraction. Texture and skin color are invariants within an individual. Between these two, texture information is automatically reflected by low-level haar-like image features in shape-based statistical models [53, 54, and 55]. Thus, besides the shape profile, I chose skin color as the other main feature to construct a personal profile of the hand. In this sense, the hand tracker implemented is multi-modal because it makes use of both shape and color features.

This chapter is organized as follows. Section 5.1 gives overview of the tracker design. Components in the off-line preprocessing group are described in section 5.2, while components in the interactive computing group are illustrated in section 5.3. Conclusions and discussions complete this chapter.

5.1. Hand Tracker Design Overview

Figure 22 illustrates the composition of the off-line preprocessing components group. In off-line preprocessing, the user captures two image sets with the camera on a mobile device: hand images and non-hand images. The set of non-hand images is used directly as negative samples, i.e. it contributes to the "what is not a hand" knowledge in the trained statistical models. The hand image set is further processed by an image selector. The purpose of the image selector is to improve the quality of the positive samples by filtering, because sample quality will greatly affect the accuracy of the trained hand shape statistical model. Criteria for the selection process are introduced in detail in section 5.2.2.

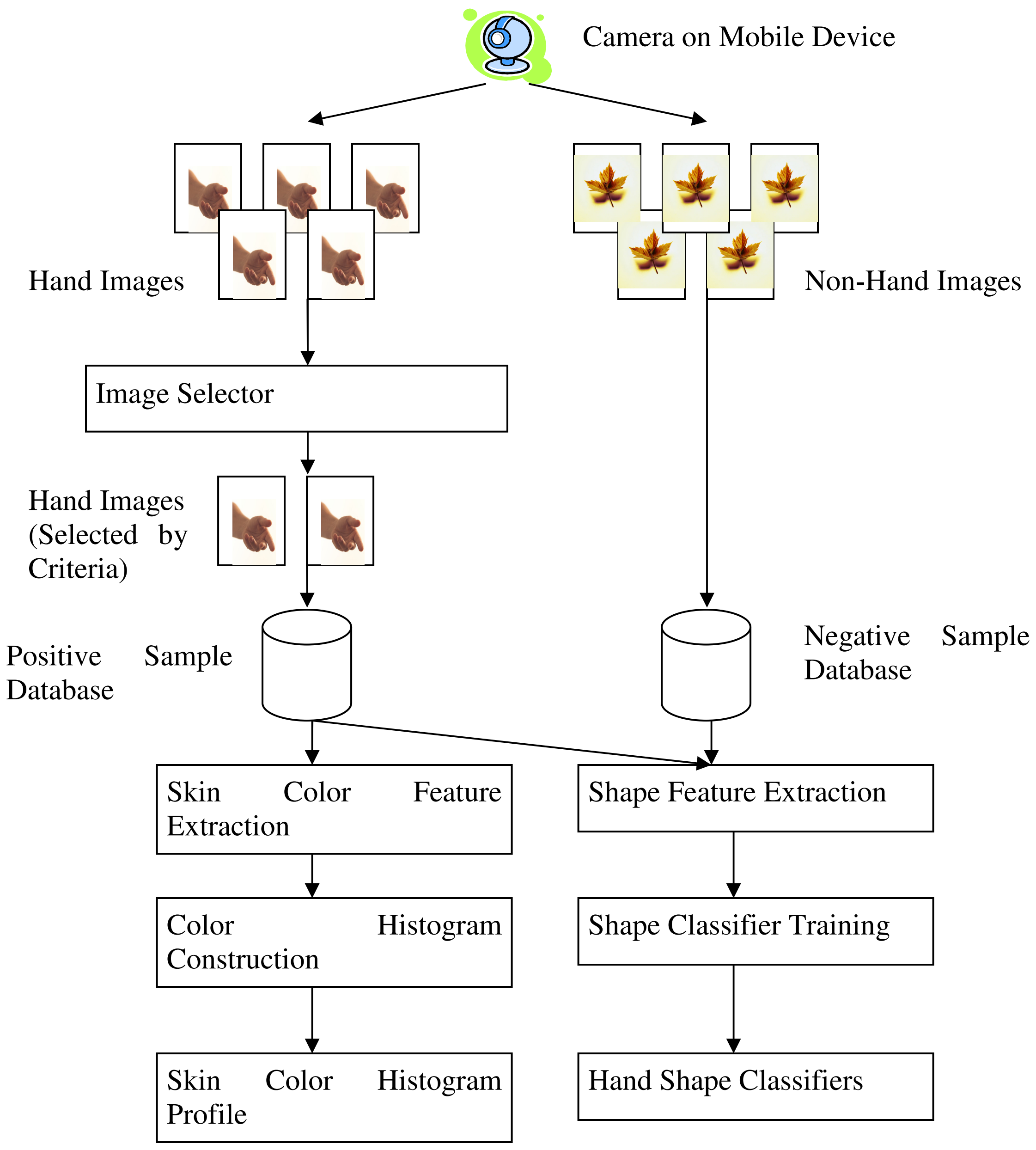


Figure 22 Block Diagram of the Off-line Preprocessing Components Group

After image selection, a positive sample database is obtained. This database, together with the negative sample database derived from non-hand samples, is supplied to a feature extraction component to extract haar-like shape features. The shape classifier training component subsequently trains classifiers based on these feature-identified samples. In the interactive computing components group, the classifiers are going to be used to identify hands from captured video frames.

The shape-based features are color-independent. That is, they are extracted from gray-scale only images and one important property of the positive images: skin color of the hand is not used during classifier building. To capture the skin color characteristics, the color features from the positive samples need to be extracted and converted into corresponding statistical models. This task is fulfilled by the color histogram map construction component. This component maps all the color points of the hand in positive samples into a color histogram. The histogram is then leveraged for hand detection.

The key thoughts of the off-line preprocessing component groups design are three folds: Firstly, a hand classifier of high quality, i.e. low false positive rate and high detection rate, should be trained from high quality training samples. This fact is determined by the nature of the statistical learning methods. Thus the image selector is introduced; Secondly, use of multi-modal features for hand tracking will bring not only the profit of lower the false positive rate but also the loss of lower detection rate. However, for the interaction purpose application uses, the penalty of lower detection rate is much less than the penalty of false positives, which harms the user experience of such a system. Finally, the sample collection process should be semi-automated to provide satisfying usability.

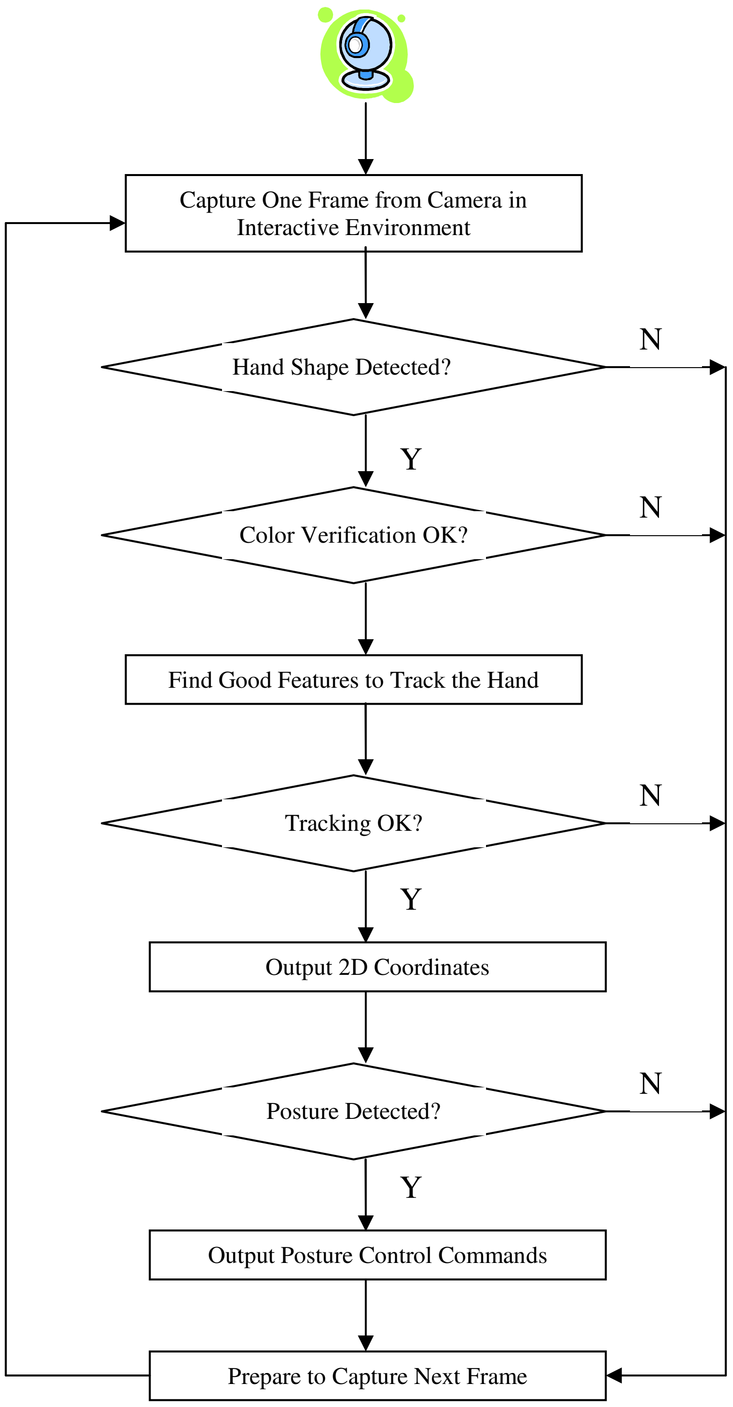

Capture One Frame from Camera in Interactive Environment
Hand Shape Detected?
N
Y
Color Verification OK?
N
Find Good Features to Track the Hand
Tracking OK?
N
Y
Output 2D Coordinates
Posture Detected?
N
Y
Output Posture Control Commands
Prepare to Capture Next Frame

Figure 23 Block Diagram of the Interactive Computing Component Group

Figure 23 illustrates the block diagram of the interactive computing component group of the hand tracker. The interactive computing component group makes use of the statistical models built by the off-line preprocessing component group. In addition, this component group is in charge of processing live video in the interactive environment and mapping hand motion to 2D coordinates and postures to control commands.

Whenever a video frame is captured by the camera in the interactive environment, it is scanned to detect if a hand exists in the scene. The criteria to justify if a hand exists are based on the statistical models built on shape features and color features. If both features are verified, a tracking feature extraction module is engaged to identify the "good features to track", i.e. KLT features as described in ([58]). The weighed centroid of the tracked features is used as the location of the hand and this is used as the 2D coordinates in the interaction with the VE.

In a "tracked" frame, the area where the hand is located is further scanned to test if the configuration of the hand satisfies a predefined posture. This is done by comparing the hand image in the area to predefined posture templates, and choosing the template that has the highest resemblance to captured hand image. The identified posture is then mapped to a control command as part of the tracker output.

After all the processing of one video frame, the tracker prepares itself for the next captured video frame. As video frames are captured at an interactive rate, the computation conducted by all the components in the interactive computing component group needs to be within a strict time limit (no larger than 100milliseconds). This is a major difference from the off-line preprocessing component group, where computation has no interactive rate requirement constraint. In section 5.3, I illustrate several techniques used to achieve the interactive rate goal.

5.2. Off-line Preprocessing Components

The off-line preprocessing components collect positive and negative training samples and generate color and shape models. Components in this group include the image capture component, which collects hand (positive) and non-hand (negative) image samples; camera motion capture component, which uses acceleration sensing to proactively capture camera motion during sample collection process; color histogram computation component, which constructs the skin color profile of a specific user's hand based on the samples collected; and shape classifier training component, which builds the hand shape profile of a specific user based on the samples collected. The first three components fit in a mobile device while the fourth component is preferably undertaken by the infrastructure.

5.2.1. Training Sample Collection Component

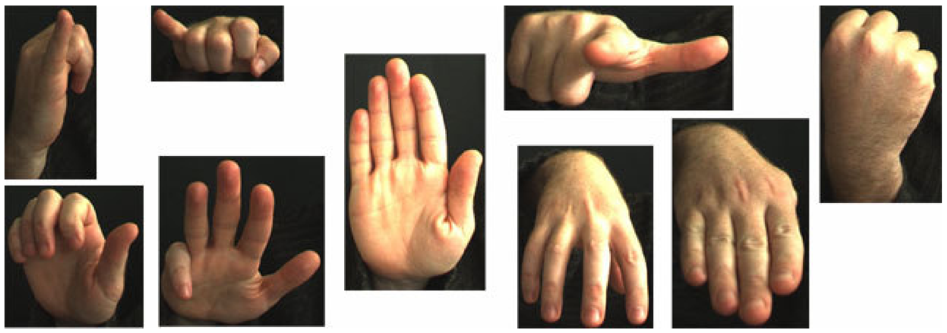


Figure 24 an Example of Hand Images Database: Hand Gesture DB

One common approach to collect training samples for a learning-based computer vision system is to get these samples from a public database. Figure 24 shows the Massey Hand Gesture Database maintained at the Massey University of New Zealand. The database contains 6 image sets collected from 5 individuals. The total number of images in this dataset is about 1600. The advantage of using a public hand images database is that user does not need to be involved in the sample collection process, and the trained classifiers will be user-independent, i.e. generalized enough to fit a wide range of hands.

Despite the above arguments, the advantages of using a public hand images database are not necessarily true when applied in a personalized computing context. Firstly, the skin color of humans varies in a great range. Asians, Africans and Americans, for example, could have drastically different skin color pigmentations. Even within the same demographic group, skin colors could still differ to a large extent. For example, African Americans and Caucasians in the United States, in general, have different skin color tones. Figure 25 shows the hand image samples I collected from two subjects, one is a southwest Asian and the other is a southeast Asian, luminance coverage and light exposure are carefully calibrated so that lighting conditions are ensured to be same for the collection of both sample sets. It could be clearly observed that the two subjects' hand differs with regard to skin color to a large extent. Secondly, the semantically identical postures, when performed by different individual, could be quite different in shape. Reasons for these differences could be finger/palm shape difference and habit of presenting a certain posture. Thus, collecting personalized hand image database for an individual user is desirable.

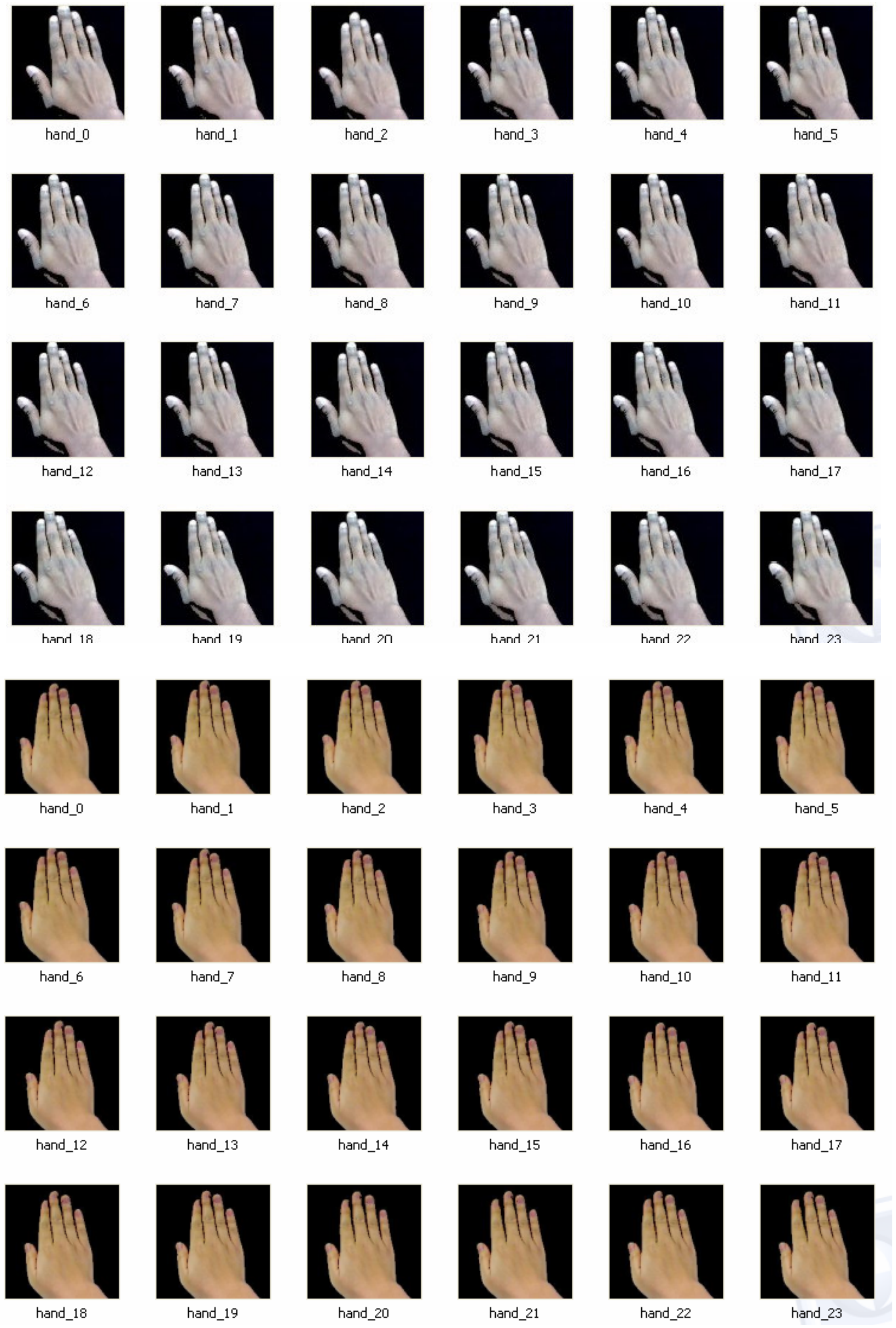


Figure 25 Hand Image Samples from Two Subjects Who Have Different Skin Colors (Top) Southwest Asian (Bottom) Southeast Asian

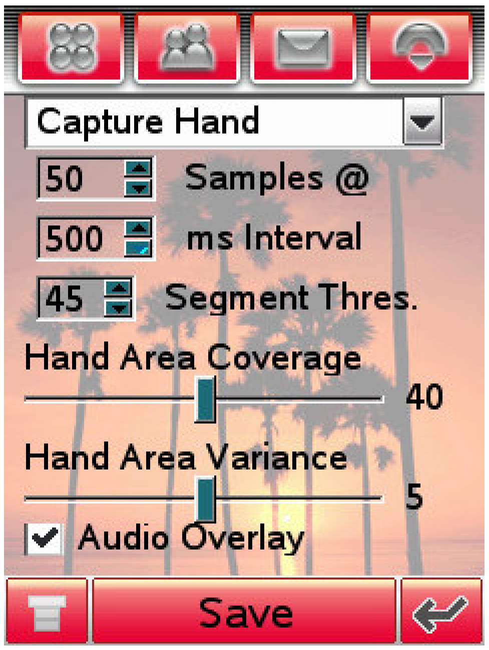
Before collecting a personalized hand image database, it is important to realize the technical challenges of this task. Unlike the pubic databases, which are collected by technology-savvy personnel and under controlled conditions, personalized database are to be collected by the users themselves. It is not uncommon that the collector themselves are not computer vision or photographing experts at all, and the collecting environment are far less controlled than a laboratory setting. To make this task even more difficult, number of images needed in a hand image database is not a small amount: it should be at least couple of hundreds if not any more.

Because of these reasons, the training sample collection component is implemented as a mixture of mobile device- and infrastructure- based applications. This component has the important features that make themselves capable of fulfilling the tasks they are assigned: 1) the collection process is semi-automated, the user needs only to specify number of hand images to be collected, and the application on a mobile device will initiate the camera at collection intervals, and performs the capturing task until the specified number of hand images has been collected. and 2) Lighting condition and hand depth condition of training sample images are automatically measured, used to judge acceptance/rejection of a hand sample and feedback to the user.

A Linux-based camera capture application was developed on a Motorola A1200 smart phone (Figure 26). The A1200 handset is a high-end mobile device equipped with MontaVista® Mobile Linux operating system, 2 mega-pixel Micron color imager, Intel XScale 312MHz processor and 1Gigabyte micro SD card.

Figure 26 the A1200 Smart Phone

The samples capture application starts with a configuration form as shown in Figure 27. In this form, the user specifies whether hand images or background images should be collected. The form also allows the user to adjust six parameters that controls the application behavior: number of samples to collect, interval between two consecutive sample collections, the segmentation intensity threshold, percentage of hand area coverage in the whole image, allowed variance in hand area coverage percentage, and whether or not audio prompts should be used. The application collects image samples at the specified interval, until the specified number of samples has been collected. By default, the hand area coverage is set to be 40% with a 5% variance allowed, and audio prompt is set to be true. More details of the hand area coverage and hand area variance values can be found in the sections below.


Capture Hand

50 Samples @

500 ms Interval

45 Segment Thres.

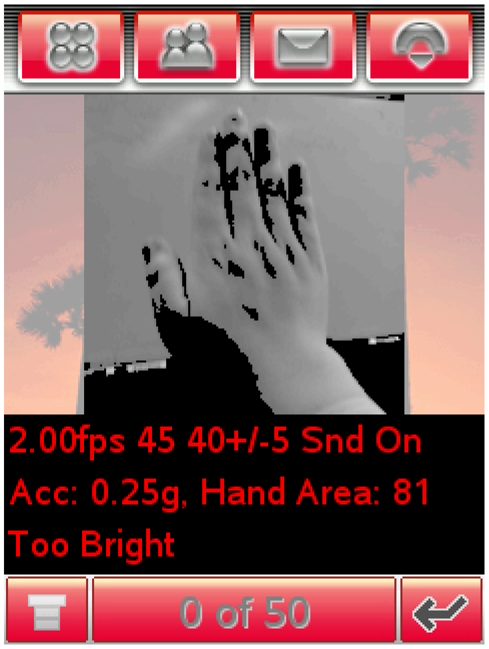
Hand Area Coverage 40

Hand Area Variance 5

[✓] Audio Overlay

Figure 27 Screenshot of the Configuration Form of the Sample Collection Application

There is no special preprocessing for background images since every pixel within such images belongs to the same category, i.e. the "non-hand" category. However this is not the case for hand image samples. This is because the hand is a highly articulated object and is very flexible and deformable. Thus, there are inevitably many non-hand pixels which belong to the background captured into a rectangular image that contains the hand. The approach to clean these background pixels is to perform preprocessing on the collected hand images. I use the steps described below to retrieve background-free hand images.


2.00fps 45 40+/-5 Snd On

Acc: 0.25g, Hand Area: 81

Too Bright

Figure 28 Screenshot of the Viewfinder of the Sample Collection Application

First, the user put his/her hand before a single dark colored background, e.g. a blue wall. The hand is captured by the sample collecting application into a raw hand image I_{raw}, with mixed hand and background pixels.

Then, the application saves a grayscale image I_{gray} from I_{raw}. It applies intensity threshold S on the I_{gray} in the following fashion: If Intensity(I_{gray}, X, Y) < S, then $I_{gray}(X, Y) = 0$; otherwise, set $I_{gray}(X, Y) = 1$. After this, a binary mask is obtained over the raw hand image, where pixels of low intensity are set to logical false while pixels of high intensity are set to logical true. Because the user is asked to take the hand image before a dark colored background. The application can assume that hand pixels are set to logical true while background pixels are set to logical false.

The application verifies if the threshold segmentation is correct by calculating the percentage P of logically true pixels in the mask:

$$P = \frac{NumberOfLogicalTruePixels}{TotalNumberOfPixels} * 100\%$$

Equation 8 Logical True Pixel Percentage

It only uses I_{raw} if P is greater than the segmentation intensity threshold. Lower than segmentation intensity threshold could indicate that the segmentation is not successful. In the case that I_{raw} is not to be used, the application does not increase the “collected” samples counter in this run and gives the user a corresponding audio prompt through text-to-speech engine, if the audio prompt option is set.

If I_{raw} is to be used, then the application uses the mask generated from I_{gray} to mask out the background pixels. After this step, a color image of the hand should be obtained which is background-free. The application saves this image as a positive sample and prepares for the next capture.

Figure 29 shows examples of the raw image, binary mask and the preprocessed positive sample, respectively. In the sample collection phase, 500 images of each category of positive samples are collected, at 2 fps rate. That is, it normally takes 5 minutes to collect positive samples for one hand posture.

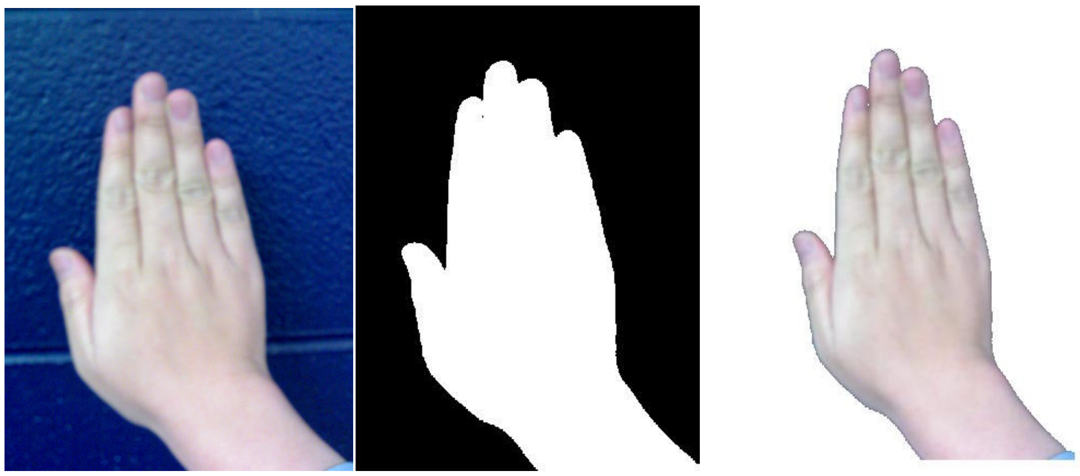


Figure 29 Raw Hand Image (Left), Binary Mask (Center) and Segmented Color Hand Image (Right)

As mentioned before, pixels are not the only data collected by the sample collection application. Each training image sample (hand or non-hand) is collected together with the mobile device's motion data with wireless accelerometer/tilt controller affixed to the mobile device (Sparkfun WiTilt, Figure 30). How the acceleration data is used for image selection is described in detail in section 5.2.2.

Figure 30 the SparkFun WiTilt Wireless Accelerometer/Tilt Controller

5.2.2. Positive Image Selection Component

Following sample collection, I am able to obtain a set of positive and negative sample images. Among them, all the positive samples satisfy the exposure and depth thresholds requirements and have the hand image segmented. These images are ready for color model training, but not ready for shape model training yet. The main reason is image blur caused by camera movements during sample collection.

Although improper focus can also cause image blur, it is the secondary issue to address when dealing with hand image capture. The reason is that the training hand images are collected in a semi-controlled environment: the hand has to be within a certain distance from the mobile device camera due to field of view limitations; and the requirement for dark color background also helps the imager to easily focus on the hand. In contrast, the slim form factor of a mobile device makes its imager very sensitive to users' hand movements. The small field of view of the mobile device camera exacerbates this problem.

There have been off-the-shelf deblurring algorithms, such as the Lucy-Richardson algorithm [64] and Wiener filter [65] but they are post-processing techniques. These techniques reconstruct the camera motion from the captured image and have no knowledge of the actual motion data. In this thesis, I use an approach that integrates acceleration data to understand the camera motion. Unlike post-processing techniques, this approach captures the camera motion proactively and stores it together with the captured images. To reduce false alarms and make use of the user's knowledge for the cause of the blur, an arbitrary acceleration threshold is to be selected by the user over the collected image samples. The user adjusts the level of acceptable levels of motion, and images that satisfy the motion threshold are chosen to be incorporated into the model training process. Once the motion threshold is finalized, unqualified images are deleted and thus training with bad data is eliminated to the best of the user's knowledge.

The accelerometer on WiTilt outputs accelerations along X, Y and Z axis. Total acceleration of the board is measured as:

$$A_{total} = \sqrt{A_x^{\,2} + A_y^{\,2} + A_z^{\,2}}$$

Equation 9 Total Acceleration Calculation

As the WiTilt board has on-board Bluetooth transceiver, data is collected at 50 Hz rate and the total acceleration through the duration of one capture interval is recorded as the metric of mobile device motion during one sample capture.

5.2.3. Statistical Model Training Component

Following image collection and sample selection processes, the positive and negative sample databases are ready to be used to build statistical models. Two kinds of statistical models of the human hand are to be built, namely the color histogram model and the shape classifier model.

The argument for developing a color histogram model for the hand is that human skin usually differs from environment in color. The differences can be described by a distribution of skin color in color spaces, usually in the form of a histogram. I train the color histogram in normalized RGB color space, with 256 bins on each color dimension. In the color space I choose, all colors that can be displayed are specified by the red, green, and blue components. One color is presented as one point in a three-dimensional space whose axes are the red, green, and blue colors. As a result, a cube can contain all possible colors ([59]). This space and its corresponding color cube in this space can be seen in Figure 31 ([60]). The origin represents black and the opposite vertex of the cube represents white.

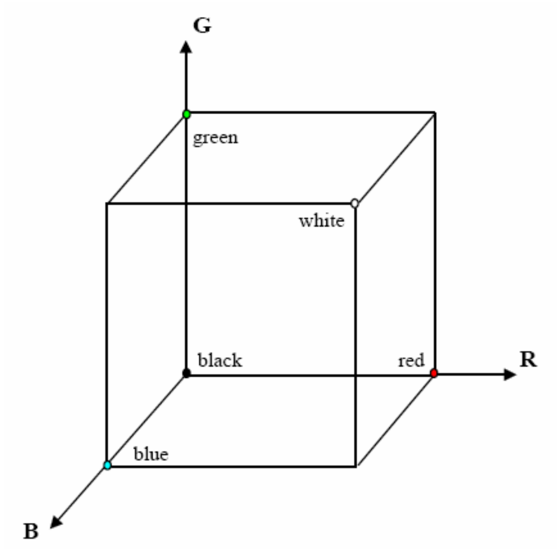


Figure 31 RGB Color Space and the Color Cube

There are two implementation details which should be noted. Due to the memory and bus bandwidth limitation of mobile device imagers, they normally do not save raw image directly in RGB color space. Instead, another color space YUV is used, where Y indicates the luminance of a pixel while U and V indicate the chrominance. The mapping between YUV format pixel and RGB format pixel can be described by Equation 10 ([61] [62] [63]). To be more specific, the A1200 device saves raw pixels in a compressed variant of YUV format called YUV422. The YUV422 format uses 4 bytes (U, Y_1, V, Y_2) to denote a pixel pair (Y_1, U, V) and (Y_2, U, V). The reason that compression of two neighboring pixels with same U and V value is feasible is because of human visual perception, which is sensitive to luminance variance but relatively insensitive to chrominance variances. These format conversions are implemented by the sample collection component so that the model training component can safely regard image collected are in RGB format.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

Equation 10 YUV to RGB conversion

Also, due to change of illumination, objects that appear to have the same color in human eyes might have quite different RGB values under different lighting conditions. To address this I use normalized RGB instead of RGB values. Normalized RGB values normalize each color component by an illumination factor, as shown in Equation 11:

$$R_n = \frac{R}{R+G+B}; G_n = \frac{G}{R+G+B}; B_n = \frac{B}{R+G+B}$$

Equation 11 Calculation of Normalized RGB

The training of skin color model is performed on the average image of all collected hand image samples and takes less than a second to process 100 images. The hand shape model training process follows the skin color model training process.

Shape-based statistical training methods map an image into one or more feature vectors, and store the knowledge retrieved from these vectors as the representative model for the presence or absence of a certain object. There are several choices for feature selection. The haar-like features, as shown in Figure 32, computes the intensity difference of neighboring rectangular areas as feature values. The advantages of haar-like features are that they are easy to compute if the image is pre-processed into an integral image ([53]), and captures texture characteristics of an object, rather than the contour lines. Figure 33 illustrates three instances of harr-like features when used for hand recognition from background, from left to right they are of type 1, 2 and 1 respectively. Another feature type that under consideration is SIFT (Scale Invariant Feature Transforms) proposed by Lowe et al ([66] [67] [68]). As the name implied, this type of features is robust to scale and rotation changes of a certain object in the image. It is also robust to partial occlusions. Both haar-like and SIFT features require only grayscale images.

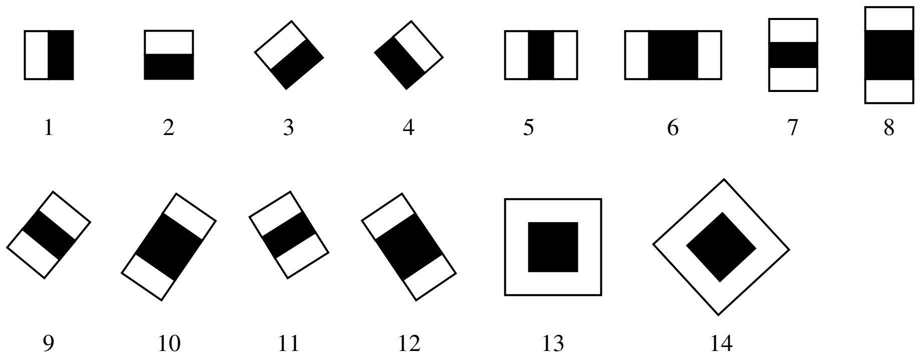


Figure 32 the 14 Haar-like Features used in Hand Tracker

In this work, I choose to use haar-like features to build hand shape models. This choice is mainly based on the availability of software package support. Haar-like features have been widely used in vision-based object recognition systems and well supported by publicly available software packages. Haar-like features computation, together with the pre-processing step of image integration proposed by Viola and Jones ([53]), are the de-facto object recognition methods in Intel's Open Computer Vision library. Alternatively, software packages that support SIFT features computation are relatively fewer in quantity.

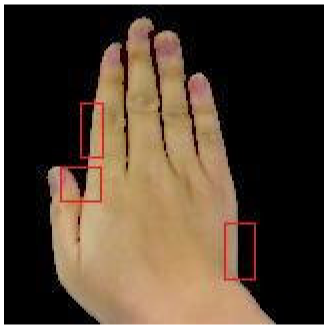


Figure 33 Instances of Haar-like Features in Hand Detection

5.3. <u>Interactive Computing Components</u>

The interactive computing components group works in the environment where live video is captured and processed in real-time. In the hand tracker context, I define ideal real-timeness as in synchronization with the VE display frame rate. A typical VE displays interactive image at 15 – 30 Hz. Converting to latency, the summed processing time of the hand tracker on a single camera frame has to be within 33-66ms. This range also fits the conclusion of the study results of Sheridan and Ferrell ([71]), which concludes that a maximum latency between event occurrence and system response of 45ms to be experienced as “no delay". In the same work Sheridan and Ferrell also state that the threshold of 300ms for when interfaces start to feel sluggish, might provoke oscillations, and cause the “move and wait" symptom.

Components in the interactive computing group include the detection component, which detects the hand presence in a video frame; the tracking component, which updates the hand location in the video frame once it is detected; and the recognition component, which identifies the posture of the hand. In the following sections, each component is described in detail.

5.3.1. <u>Detection Component</u>

From a high level perspective, hand detection is a pattern matching process (so is hand recognition), where the presence of hand is compared with the statistical models built earlier by the off-line pre-processing components group. As I have two kinds of statistical models pre-built - the skin color model and the hand shape model - there are two sequences to make use of them:

Sequence 1 is to use the skin color model to identify the hand from its background then obtain one or more hand-colored foreground blobs. Shape models are subsequently applied on these discovered blobs to verify if it is truly a hand posture in the pre-defined posture vocabulary.

Sequence 2 is to use the hand shape model to identify a specific hand posture from the captured image, and then use the skin color model to verify that the object retrieved is indeed a hand and not a hand-like object that does not have the skin color. In the personalized hand detection process, color verification is also helpful in filtering out the hands that do not belong to a specific user.

By comparing the two sequences, I choose to use the shape-first-color-second sequence for several reasons. Firstly, using a shape-based model to do the first level identification eliminates more false positives than using color-based model to do the same job. Foreground color blobs that are identical or close to the skin color are not rare occurrences, although most of such blobs can be rejected by shape-based models with relative ease, bringing these false positives into the second level identification stage is not necessary. Secondly, to identify the skin color foreground blobs calls for a scan of the whole image area, whereas identifying a specific posture could be conducted in one or more smaller predefined range-of-interests (ROIs) in VE space. Given that the computational load is linearly correlated with the size of the scan area, color-first-shape-second sequence pays a higher computation penalty than shape-first-color-second sequence. Lastly, using the color-based model to do the first level identification has the problem of scalability with respect to the number of predefined hand postures, i.e. the hand posture "vocabulary". All identified skin color foreground blobs need to be examined against all possible postures in the predefined vocabulary. Conversely, in shape-first-color-second sequence, there is only one comparison with the skin color model needed once the hand posture is identified from the ROI.

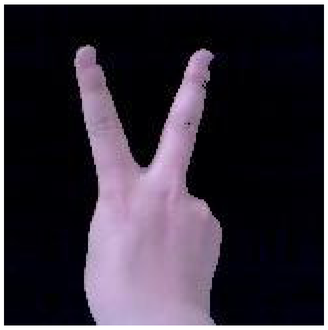


Figure 34 the "V-Shape" Posture Used for Hand Detection

Based on the above reasons, I choose to use the shape-first-color-second sequence. The other design consideration is whether to give the user complete freedom of exhibiting his/her hand at any posture and at any area of the camera's field-of-view to be detection, or to set up a certain protocol as an agreement between the user and the hand tracker, only when this protocol is followed then the hand will be detected? I believe that when used for interactive purpose, the detect component's reliability is more important than its detection rate. Missing one or two frames that contains the hand shape form is acceptable – after all, it takes much longer for the human hand to enter/exit a scene than the time of a video frame update. Conversely, reporting a hand presence while there is actually no hand there will trigger false responses of the system, and in turn frustrate the user who is using the interactive environment. Thus, I use the "V-shape" posture as shown in Figure 34 for hand shape detection. The protocol is as follows:

- The detector regulates one or more configurable sub-areas of the camera field-of-view that it constantly monitors for hand presence. These sub-areas are called "hotspots".

- When the user would like to start an interactive session, he/she places the hand in front of the camera, making the hand image fall into one of the "hotspots".

- Not only does the hand need to be within a "hotspot", it also needs to be of a specific posture configuration. Figure 34 shows the "V-shape" hand posture I used for the interactive system.

- Once the detector identifies that the hand image in the "hotspot" is related to the predefined posture, it then uses the user's skin color model to judge if the hand-shaped object is indeed of the user's skin color. If the verification is positive, then the detector deems that hand presence has been detected.

The method that the detector uses to detect a posture match is to scan the image area with the template scaled at various sizes. As illustrated by Figure 35. If the hand image is completely within the hotspot boundary and it could be covered by a scaled template image, then a shape match is returned. The accuracy of detection is independent of the hotspot area size. However, scanning time is tightly correlated with the hotspot size. I present these results in the evaluation section.

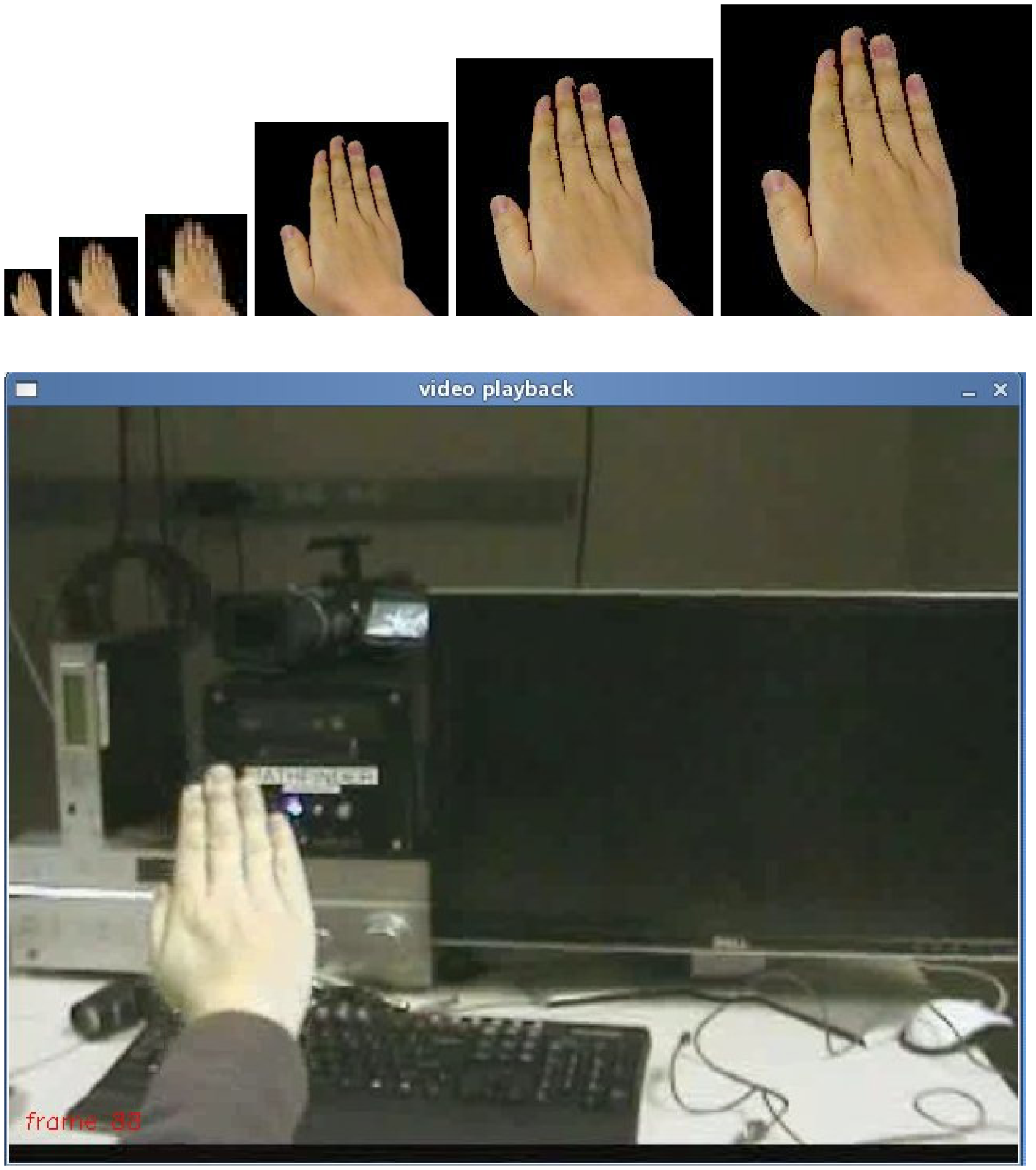


Figure 35 Template Matching at Different Scales on a Sample Video Frame, the Posture is Matched at Scale Level 4.

5.3.2. Tracking Component

The hand tracking process is different from the hand detection process as described above. In the hand detection process, whether the hand is in camera's field-of-view or not is not known a priori by the detection component. The detection component needs to constantly monitor all the ROIs to respond to the predefined hand posture presence. However, during the tracking process, the presence of the hand in camera's field-of-view is known a priori, what is to be determined is where the hand is located in the next video frame. Tracking starts right after the hand is detected and stops when no satisfying cues could be found to identify the hand existence in the next image frame.

The human hand is a very adept part of the body as it is highly deformable, thus could exhibit a large quantity of arbitrary shapes. At the same time, the multiple joints in upper limb give a high degree-of-freedom to the hand, making the combinations of in-plane rotation and 3D rotations a big set. Because of the forbidding numbers of possible hand configurations, tracking the hand by testing a classifier set against each video frame is not practical. To address this challenge, Kanade, Lucas and Tomasi proposed a method to find the steep brightness gradient along multiple directions. These gradients are called KLT features after the creators' initials ([58]). In this work I use multiple KLT features to track the hand once it is identified by the detection component and the heuristic rules described in [33] to manage these KLT features. The implementation scans two consecutive video frames to track 30 KLT features and makes use of off-the-shelf KLT feature tracking function from the OpenCV library [73].

5.3.3. <u>Recognition Component</u>

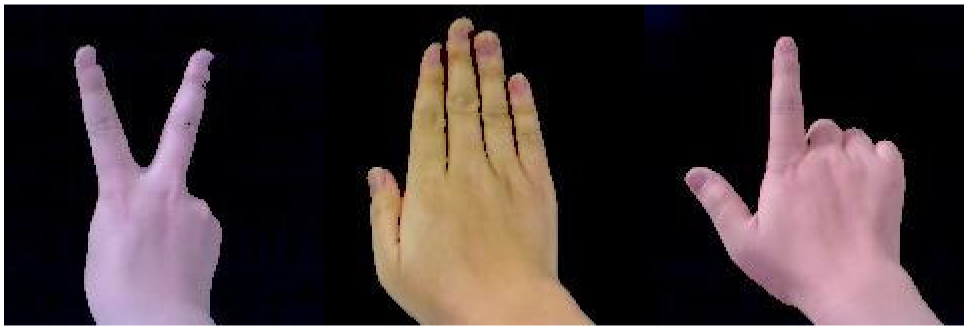


Figure 36 the Three Hand Postures Used for Posture Recognition

The recognition component fulfills a task that is different from the ones fulfilled by the detection and tracking components. Similar to the detection component, the recognition component also works on ROI. But the ROI here is not the “hotspots” handled by the detection component, but rather the image area identified by the tracking component of hand presence. The recognition component constantly monitors the tracked ROI and identifies possible presence of the predefined hand postures. In the implementation, the recognition ROI size is 320 by 240.

Figure 36 shows the three hand postures predefined in the posture vocabulary: “V-shape”, “flat” and “arrow”. Once the hand is tracked, the image area contains the tracked hand is processed by the recognition component and compared with all three postures. If the shape-based statistical model and the skin color-based histogram both report positive for a predefined posture, then the recognition component assert that the posture is performed by the user and transmits this decision forward. As I can see, except that the ROIs worked on are different, the tasks performed by the recognition component are almost identical with the tasks performed by the detection component.

Determining how to predict the recognition scan area and scan scales from the tracked hand position is the key in determining the recognition rate and performance. Because the recognition component only scans the predicted recognition area, inevitably some postures will be missed when they are out of the predicted recognition area. However, if I set the recognition area to be too large, then the recognition performance will plummet. In [33], the author proposes the prediction area as described below:

*Given the center position reported by the tracking component to be [C_x, C_y] and the last match scale to be S. Predict the scan area to be a boundary box centered at [C_x, C_y] and of width 2 * S * TemplateWidth and of height 2 *S * TemplateHeight, and predict the scan scales to be [S/1.75, S*1.75].*

The advantage of the prediction algorithm proposed in [30] is that it takes a constant time to compute, regardless of the scale level used for prediction. However, this advantage is built on two other drawbacks: firstly, the normalized scan scale range of this algorithm is (S', S'*3.06). However, to effectively capture a 25 by 25 pixel template in a VGA size video frame, the theoretical scales range needs to be within (1.0, 18). In practice, the range of (1.0, 8.0) is the minimum to achieve satisfying capture results. This is still much larger than the (S', S'*3.06) range. If the hand has large depth movement since last match, this algorithm is prone to not being able to recognize the corresponding hand posture in video frame. Secondly, when the scale level of last matched image is small, to predict X-Y movement by setting the scan area to be 2 * [scaled level times template size] is not valid under many occasions, because the hand motion could easily exceed the boundary of the predicted area. Say, for example, the hand image maintains 25 by 25 pixels in the video frame but moves at 60pixel/frame speed, then the predicted area will never be able to capture the hand image in the video frame. Because in the system presented by [30], all hand tracker task computations are performed by a single laptop and workload optimization is critical, it is understandable why the specific prediction algorithm is devised. In the hand tracker implementation, because I have the scalable computing techniques to speed up processing (more details to be given in Chapter 6), a reasonable amount of computation should be put on the recognition process to increase recognition rate. Based on these observations, I design the prediction algorithm as following:

Use two predicted recognition areas R1 and R2 for each recognition scan. Both R1 and R2 are centered at [Cx, Cy], which is the tracked hand position. R1 is of size 320 by 240, and R2 is the smaller of the two areas: the whole video frame and a scan area four times of R1. When scanning R1, use the scan range of [1.0, 8.0]; When scanning R2, use the scan range [9.0, 12.0].

Similar to the algorithm in [30], the algorithm also computes in constant-time, because the two scan areas of it are invariant across frames. The reason of only applying large scale levels on the latter scan area is based on the observation that hand movement simultaneously on all three directions are rare occurrences. Although it takes more time than the method in [33], there do exist scalable computing techniques than can significantly improve processing performance (scalable computing techniques are presented in Chapter 6 in details). Also, the method has no preference over a certain scale range and gives the hand posture presence in between scales [1.0, 12.0] evenly and thus is able to recognize some hand posture presence that the method in [33] would have missed. Over a video sequence which consists of 5597 frames, this prediction algorithm is able to recognize 2962 hand postures while the algorithm proposed by [30] captures 2560.

5.4. <u>Evaluation Results</u>

In this section four important metrics of the hand tracker implemented are evaluated. The first is the effectiveness of the sample collection method in improving the shape model quality. The remaining three which are latency related are performances of 1) the detection component, 2) the tracking component and 3) the recognition component.

The evaluation experiments are undertaken on the head node of a computer cluster. The node is 64bit architecture with 2 AMD Opteron processors (2 GHz, Model 246) and 4GB of DDR 400 MHz RAM. The operating system is SuSE Linux 10.

Time is measured with the Linux clock() system function call. Experiments show that the clock() call is capable of reporting time at a precision of 10 milliseconds. The low resolution is due to the fact that the clock() call is subject to context switch latencies. Although I have implemented a time measuring tool elsewhere [74] which is able to measure at microsecond precision but it is not incorporated in the hand tracker due to current software library compatibility issues.

The camera used here is Flea® (PointGrey, Vancouver, BC, Canana). The Flea® is a compact IEEE1394 camera with a 1/3" Sony CCD and 12 bit analog to digital converter. The resolution used is 640 by 480 and due to OpenCV driver limitation, live video is processed in grayscale mode.

5.4.1. <u>Effectiveness of Sample Collection Method</u>

500 hand images are collected for each hand posture and a mixture of 4000 non-hand images are collected as well to train the shape model. Within the 500 hand images, 50 are collected from the sample collection application on the A1200 device, the other 450 are

derived from the first 50 images by applied a random rotation angle from -5 degrees to 5 degrees. Out of the 4000 non-hand images, 3000 are from an online negative sample database, 1000 are collected at EVL-UIC. Again, the first 100 EVL-UIC images are collected using the sample collection application on A1200 device, and the remaining 900 are derived by applying -5 degrees to 5 degrees rotation angles.

To evaluate the effectiveness of the exposure and depth threshold control methods, classifier performances are tested under three conditions. The first condition, illumination and motion controlled, enables the segmentation intensity threshold in the sample collection application for hand images, and restricts the total non-gravity acceleration to be 0.1g during the shutter time for both hand and background images; the second condition keeps the illumination constraint on hand images, but not the motion restriction on either hand or background images. Instead, it does post-processing of deblurring on the collected images using Lucy-Richardson algorithm [64]; the third condition just uses timed capture to get the hand images and background images, and does not apply any constraints during the image collecting process. I then use 1500 hand posture images and 1500 background images to train the haar-feature shape model under these three conditions, and use the remaining 1500 background images as testing sample to obtain detection rate and false alarm rate data to be plotted into a receiver operating characteristics (ROC) graph. Each ROC curve consists of 16 data points with varying parameter configuration of the hand shape model to output different detection-rate and false positive rate value pairs.

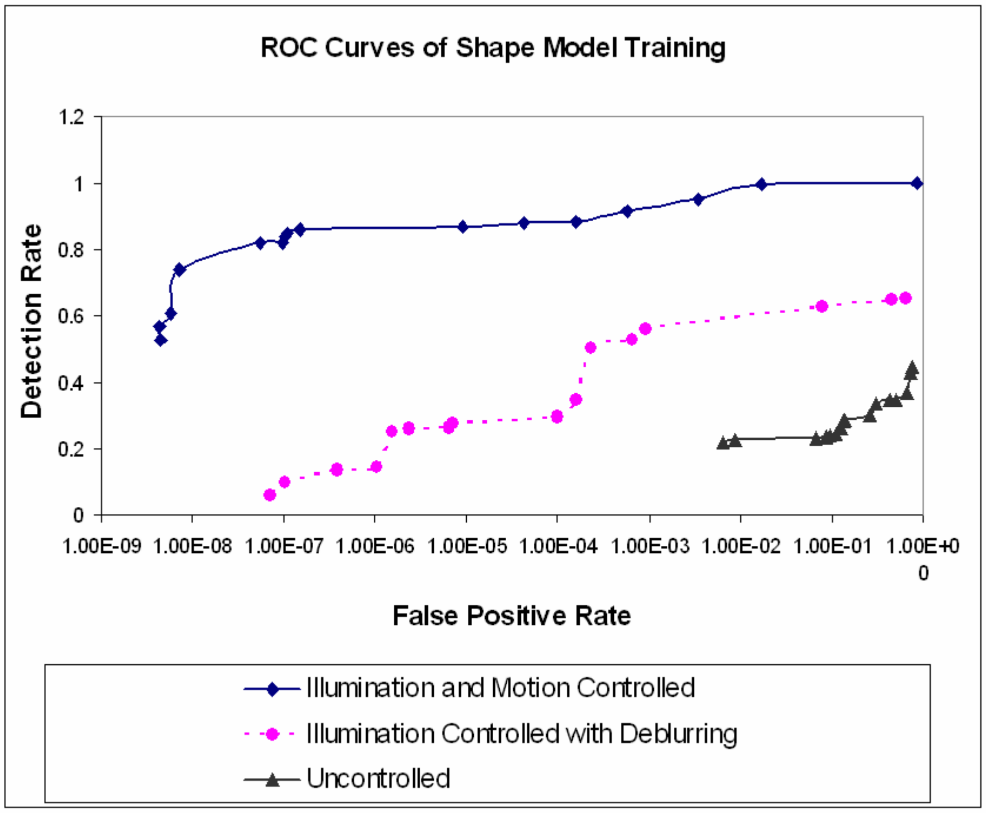


Figure 37 ROC Curves Showing the Comparison among Illumination-and-Motion-Controlled, Illumination-Controlled with Deblurring and Uncontrolled Sample Collection Methods

Figure 37 shows the ROC curves for the hand shape models trained under the three conditions. It can be observed that without any constraints enforced, the noisy samples make the shape model's performance low: the best detection rate is 0.43, while the best false positive rate is only $6.44 * 10^{-3}$. Considering when the template size is 25 by 25 and the frame image size is 640 by 480, each frame contains about $4.7*10^{4}$ template size sub-areas. The false positive rate is equivalent to 8-9 false alarms in each video frame. This accuracy performance is not acceptable for the needs.

By applying illumination constraints significantly improves the performance of the hand shape model trained, the best false positive rate I can get increases to $7.02 * 10^{-8}$, this is about one false positive every 300 frame images when performing 25 by 25 template scan over 640 by 480 frame image. However, when the best false positive rate is achieved the detection rate is only 0.06. Actually, the best detection rate of the trained hand shape model is only 0.65. Obviously, this sample collection method does not suit for the need, albeit it already far outperforms the uncontrolled method.

When both illumination and motion constraints are enforced, I am able to obtain both satisfying detection and false positive rates. When the detection rate is 0.82, the false positive rate is $5.6 * 10^{-8}$, which is about one false positive every 380 frames. Given the frame capture rate at 50 frames per second, this is equivalent to one false positive every 8 seconds. In the implemented system with the second stage skin tone verification, I observe about 3 false positives in a 5597-frame video sequence.

5.4.2. <u>Computation Performance of Detection Component</u>

The processing time of the detection component is almost linearly correlated to the hotspot size. Figure 38 illustrates the processing time of the detection component from frame to frame, when the hotspot area varies from 10% to 100% of 640 by 480 frame image. The detection time is about 15 milliseconds when the hotspot accounts for 10% of the frame size, i.e. 200 by 150 pixels. When the whole frame image is monitored, the processing time is about 205 milliseconds.

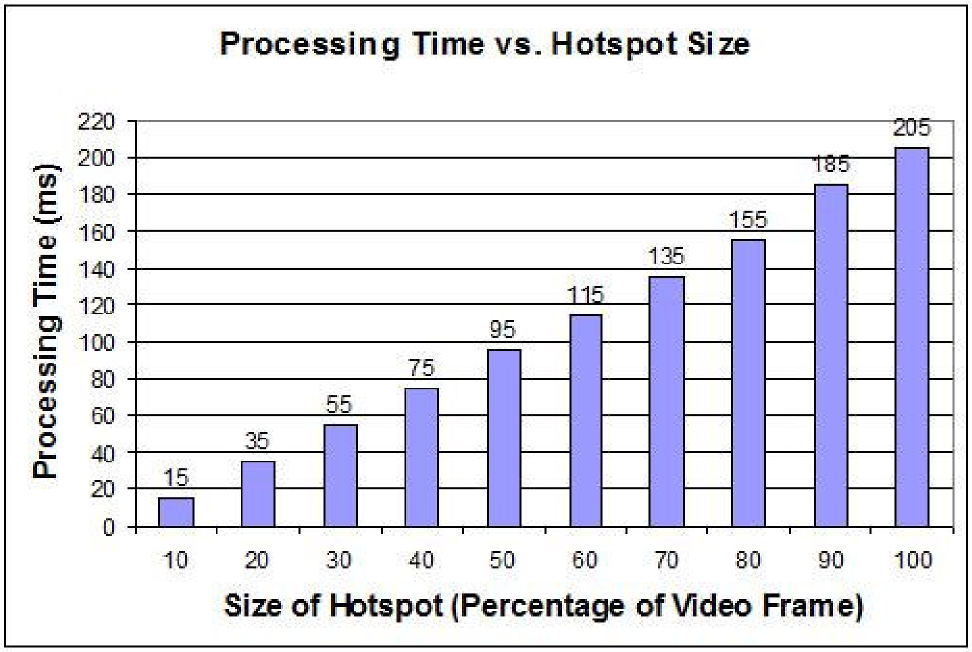


Figure 38 Processing Time of the Detection Component, with Difference Sizes of the Detection Hotspot

In the implementation of the hand tracker, the detection hotspot is set to be 320 by 240, i.e. the hotspot accounts for 25% of the frame image size. Figure 39 illustrates the processing time I recorded in 682 consecutive frames. In 99% percent of the frames, detection component takes 50 milliseconds, while in the rest 1% it takes 60 milliseconds. The average detection processing time is 50.1 milliseconds and the standard deviation is 1 millisecond.

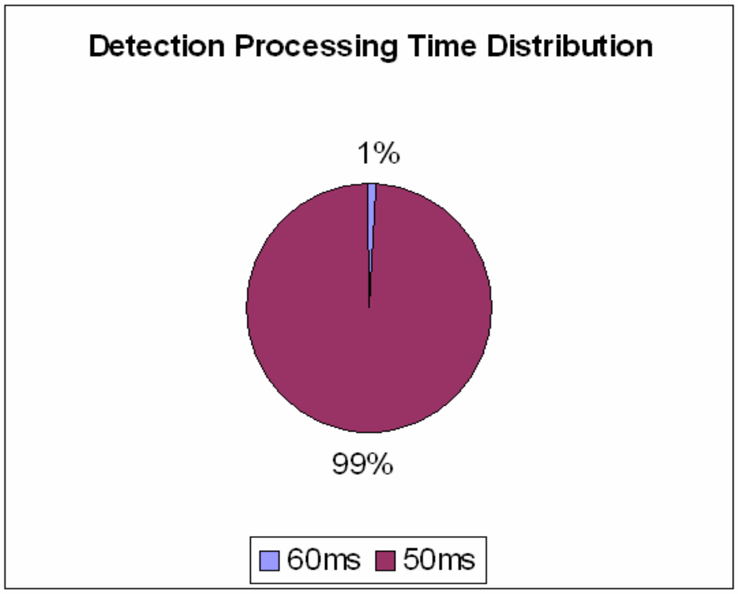


Figure 39 Processing Time Distribution of Detecting Hand in a 320 by 240 Hotspot from Frame to Frame

5.4.3. <u>Performance of Tracking Component</u>

I empirically choose 30 to be the number of KLT features to be tracked between two consecutive frames. A value less than this makes the tracker prone to tracking loss, while a value larger than this makes the number of falsely tracked features to increase. When 30 KLT features are tracked from frame to frame in 682 consecutive frames, the processing time varies from 10 milliseconds to 30 milliseconds and no more than 30 milliseconds. The average tracking processing time is 18.48 milliseconds and the standard deviation is 4.24 milliseconds. 80% of the frames are processed between the 10 milliseconds- 20 milliseconds range and 98% of the frames are processed within 20 milliseconds.

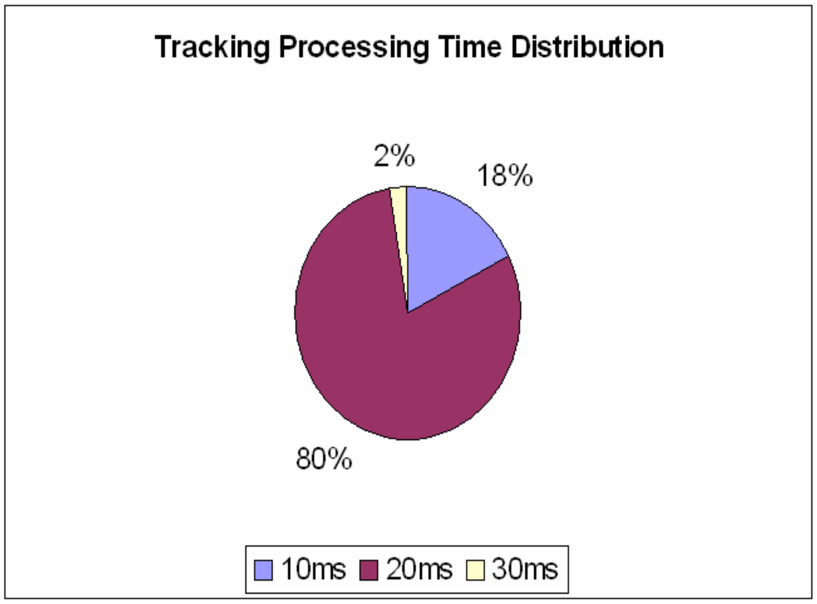


Figure 40 Processing Time Distribution when Tracking 30 KLT Features from Frame to Frame

5.4.4. <u>Computation Performance of Recognition Component</u>

According to the recognition area prediction algorithm, the recognition component scans two areas R1 and R2. Size of scan area and scale ranges when scanning R1 is the same as scanning of the hotspot area by the detection component. Besides R1, the recognition component also needs to scan R2, which is the lesser of the whole frame image and an area at four times the size of R1, at the large scan scale range. The workload of scanning R1 is the same as the workload for scanning the hotspot by the detection component. While scanning R2 introduces extra processing workload. Figure 41 illustrates the recognition processing time recorded over 682 frames. 94% of the frames are processed at 60 milliseconds, 5% of the frames are processed at 70 milliseconds and 1% of the frames are processed at 80 milliseconds. In the experiments, the average recognition processing time is 60.67 milliseconds and the standard deviation is 2.79 milliseconds.

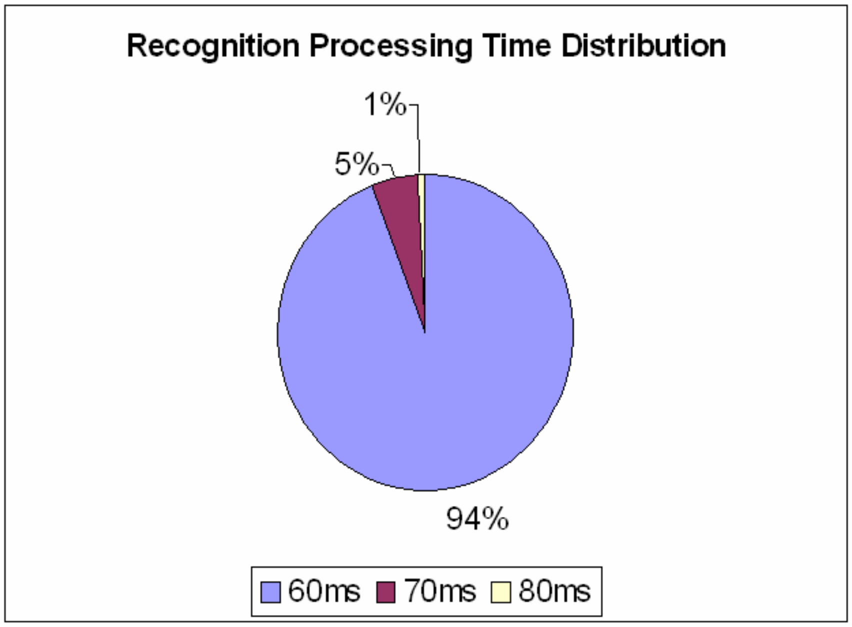


Figure 41 Processing Time Distribution of Hand Recognition from Frame to Frame

5.4.5. <u>Overall Frame-to-Frame Latency of the Hand Tracker</u>

The frame-to-frame latency of the hand tracker is defined as the time for detection in each frame (when the hand tracker is in the detection state), or the sum of tracking and recognition in each frame (when the hand tracker is in the tracking state). From the results above it can be seen that the frame-to-frame latency is about 50 milliseconds when the tracker is in detection state and about 80 milliseconds when the tracker is in the tracking state. If I do not perform posture recognition task under the tracking state, then the frame-to-frame latency is about 20 milliseconds. So, under the three conditions: detection, tracking with posture recognition and tracking without recognition, the frame rate should be 20, 12 and 50 respectively. In the actual system, due to two other latencies: hardware capturing latency of the camera imager and display latency of the rendering, the frame rates measured are averaged at 17, 9 and 38 frames per second under these three circumstances.

5.5. Conclusions and Discussions

In this chapter the design and implementation of a hand tracker are described in detail. The implemented hand tracker is able to perform at interactive rate (about 10 fps when both tracking and posture recognition are undertaken) and satisfy recognition accuracy for VE applications. Compare with other state-of-the-art hand tracking systems, this work stands out with the following unique features:

Efficient use of both the cameras on mobile device and in interactive environment

As best as I know, this work is the first that fully uses the camera on a mobile device to collect hand and non-hand image samples for statistical model training, and uses the camera in an interactive environment for hand tracking and gesture recognition once the statistical models are trained. This task separation makes efficient use of both computing platforms: the mobile device and the VE, with no sacrifice for posture recognition performance and accuracy. Due to the fact that in present days imagers are almost a commodity component on mobile devices, this work finds new potentials to make better use of them.

Combines both tracking and controlling functionality

The hand tracker acts not only as a 2-D coordinates tracking device, but also a three-button control device. This fits the requirements of the majority of VE applications, which call for inputs from both trackers and/or controllers. 2-D tracking is using optical flow methods, which guarantees an interactive rate for the computation, while controlling is handled by template matching, thus recognition accuracy is achieved.

Constructs and makes use of the personalized profile effectively

Besides the shape model training in the form of classification trees, this work also constructs and makes use of the skin tone profile of the user. Skin tone, as an important biometric feature, is captured and modeled completely on the mobile device. The use of personalized skin tone information incorporated in color models improves the recognition accuracy when classifying hand and non-hand objects, as well as hands from different users. Distributive acquisition and storage of the skin tone profile with personal mobile devices solves the problem of maintaining a monolithic database for all users, and makes sure that the skin color profile is always available when the user needs to perform human-computer interaction in a new VE. This solution is thus highly scalable.

Proactive use of motion data to facilitate high-quality sample collection

Acceleration data, when captured synchronously with pixel data, is shown to be effective in addressing an important challenge for user-conducted image sample collection process: image blur caused by camera motion. This work is the first that studies quantitatively the comparison between effectiveness of proactive motion-sensing and post-processing deblurring methods, and shows that the proactive motion-sensing method help to build higher quality hand shape models. Because of the introduction of the motion-sensing element, the sample collection application gains both friendliness and robustness to a degree so that it can be used on a mobile device.

With all the above statement said, there are also opportunities to improve this work. First, SIFT features could be used instead of haar-like features because of the advantage of scale and rotation invariance. Due to the exactness match of SIFT key points, the method has an overtraining problem but this could be compensated by boosting methods ([56] [70]). As aforementioned, the main barrier to implement the SIFT-based hand detector is availability of supporting software packages. To use a publicly available SIFT implementation named libsift ([69]) is a follow-up of this research work.

The second opportunity is to further investigate the use of multi-sensor data for the components of the hand tracker. Besides the use of an accelerometer, other sensors can also improve the performance from multiple aspects for the hand tracking tasks. For example, a magnetometer can indicate the geographical direction the user is facing when collecting hand samples, and thus get better natural lighting calibrations. Similarly, if a gyroscope is attached to the user wrist while he is using the hand tracking component in interactive computing mode, the rotational accelerations from the gyroscope can help the tracker to adjust the rotation angles of the gesture templates as well. The idea of proactive use of sensor data on mobile device in this chapter is just the beginning effort to explore the usefulness of such techniques, and could certainly be extended to cover a broader scope of application fields.

The third opportunity is computation scalability. Grouping of computational components of the hand tracker into off-line preprocessing and real-time computing defines the corresponding roles in a synergy of the mobile device and the VE. The multimedia peripherals of the mobile device are maximally utilized. In the mean time, the mobile device's processing capacities are also made proper use of by the image preprocessing task. On the VE side, its more powerful computation capacity makes it a good candidate for model training. However, as have been seen in the evaluation results, a single desktop system still cannot fulfill the combined hand tracker tasks at very satisfying frame rate. To make use the processing capacity in the infrastructure with higher efficacy, high performance computing techniques should be considered and corresponding solutions need to be developed. In chapter 6 I present the scalable computing techniques devised to address the high performance hand tracking problem.

6. SCALABLE COMPUTING TECHNIQUES FOR HAND TRACKING

In Chapter 5 the design and implementation of a hand tracker are presented. By using shape-based model matching with skin color verification, the tracker achieved satisfying detection and false positive rates. However as described in the discussion above, lack of scalability is the hand tracker's characteristic that can be further improved. For example, due to the requirement for real-time response time for hand detection, the tracker only monitors a sub-region of the camera field of view. This is counter-intuitive as to how a vision-based system is expected to behave. In fact, one of the main desired features of a vision-based tracking system is that camera existence is nearly unnoticeable. Asking the user to follow an explicit protocol for hand detection undermines this goal. If I look further into the hand tracking process, I can also find that hand detection and posture recognition belong to a special category of pattern recognition problems, where the test data itself is dynamic and time-sensitive. Latency in processing of one test sample might cause the system to miss one or more subsequent test samples, and thus give an adverse result not as expected by the user. Based on these observations, this chapter investigates the opportunities to apply scalable computing techniques for the computation-intensive tasks of the hand tracker. The goal is to achieve throughput improvement and processing time speedup, at a reasonable cost of the infrastructure resources. I introduce a novel data structure, called the scanning node tree, which is able to organize multiple processing nodes into effective synergy for hand tracker tasks. Scanning node tree not only fits pre-connected computer clusters, also is suitable for ad-hoc systems because of its multi-hop nature. Then I present a load balancing algorithm which is to automate the scanning node tree construction process, as well as task partitioning optimization. At last, detailed evaluation results are given with respect to the several key metrics of the scalable computing performance.

6.1. Problem Statement

Scalable computing is broadly defined as being able to handle a growing amount of computing workload in a graceful manner. In the hand tracker case, the problem is how to orchestrate a computer cluster to speed up the processing of hand detection and posture recognition. More specifically, I aim to achieve the following goals:

- Increase the throughput of the processed video data.

Because the hand tracker process a portion of the whole frame image for each video frame, under the same frame rate, the higher amount of video data the hand tracker is able to process, the larger percentage of the frame image is able to be covered. In practice, at present day a video frame's resolution ranges from VGA (640 by 480) to 4K (4096 by 2160), using more processors in a computer pool should be able to handle video frame increases gracefully.

- Decrease the frame processing latency and increase frame-per-second metrics

Frame latency introduced by processing of the hand tracking tasks is a major factor that determines the user's experience. The time gap between the performance of a certain posture by the user to the application's response to the user with a perceptual modality (visual, audio or haptic) is the key to the experiences of using a VE. With regard to the frame-per-second metric, it is worth noting that the overall latency of a vision-based tracking system is not solely determined by the frame image processing task but also by other factors, such as video capture and tracking data output methods.

- Get polynomial, preferably linear curve of speed-up values against number of computation nodes.

In a perfect world the processor pool that can be used has unlimited number of nodes to be chosen for any cluster computing algorithm. In the real life the number of nodes in a computer cluster is usually limited. This is especially true in Beowulf[8] clusters because they normally use messaging passing software for parallel processing. I would like to use only reasonable number of cluster nodes to fulfill certain size of hand tracking computing task. The bottom line is that the number of computation nodes should increase polynomially and not exponentially with the task size increase. Preferably, number of computation nodes needed should increase linearly with the task size increase.

- Balance the computation load as evenly as possible among participated cluster nodes.

I would like the computational load assigned to each participating node in the computer cluster to be as even as possible. One advantage of load balancing is that it could help to reduce frame processing time, because frame processing time is determined by the longest time a node takes to process a sub-task of the hand tracker. Another reason of the load balancing is that the computer cluster usually handles multiple jobs at the same time period and the hand tracker is just one of them. To be as friendly as possible to other processing jobs maybe the best way is to develop an even load of CPU usage on each cluster node.

[8] A *Beowulf cluster* is a group of usually identical PC computers running a Free and Open Source Software (FOSS) Unix-like operating system, such as BSD, Linux or Solaris. They are networked into a small TCP/IP LAN, and have libraries and programs installed which allow processing to be shared among them.

- Get low overall computation and communication overheads

With the advantage of parallel processing, a parallel processing solution has the cost of processing and communication overheads. The processing overhead is introduced by the overlaps among sub-tasks when the original task is divided, while the communication overhead is usually defined by the passing of data and control messages when a message passing implementation is used. I would like to get low computation and communication overheads when the parallelization solution is used.

The goals are to be achieved with a reasonable assumption of the characteristics of the computation resources in the infrastructure. First, I assume that these computation resources consist of multiple computer nodes which are interconnected through network links, the interconnect topology can be either central-switched or ad hoc. Next, I assume that the computation resources can be described by LogP model ([78]). The LogP model characterizes a parallel machine by the number of processors (P), the communication bandwidth (g), the communication delay (L), and the communication overhead (o). Besides its basic form, LogP also has an extension model called LogGP [79] (G captures the bandwidth obtained for long messages) to make it more realistic under the circumstances that long messages are passed among computer nodes. Because the longest message to be sent in the hand tracker case is a VGA sized grayscale frame, the size is about 640x480 bytes and counts for only about 3% bandwidth of a gigabit network link. Use LogP to characterize the computation resource in the infrastructure is enough and appropriate.

6.2. <u>Scalability Analysis for Hand Tracker</u>

In this section, I give an in-depth anatomy of the computation tasks performed by the hand tracker. Recall that the interactive computing components group consists of the following three components: hand detection component, hand tracking component and posture recognition component.

The hand detection component monitors the pre-defined detection hotspot and report hand presence when there is a pattern matching. The monitoring process is by scanning the hotspot area of each frame with different size-scaled posture templates, each template matching is added to a matching list, when the whole hotspot has been scanned the best one in the matching list is reported as the hand detected. The hand detection component takes about 50-60 milliseconds processing time on a 320 by 240 hotspot area.

If in the interactive environment a VGA resolution (640 by 480) camera is to be used, then a hand detection component that is able to run at 50-60 milliseconds per frame is able to monitor only a quarter of the frame image size. The processing time bottleneck at a single node prohibits the monitoring hotspot from being enlarged. This as described, is the throughput problem I am trying to address. Then again, if I define multiple hotspots to be monitored, each of them is of the same hotspot size that a single cluster node could handle within 50-60 milliseconds time, but the hotspots are assigned to multiple cluster nodes to process. Then it is possible that the whole tracker's detection throughput can be improved. From the overhead perspective, the preferred data partition will be making the hotspots disjoint with each other. i.e. they do not have overlapped areas. This is acceptable by the interactive environment user. Figure 42 shows the proposed parallelization solution.

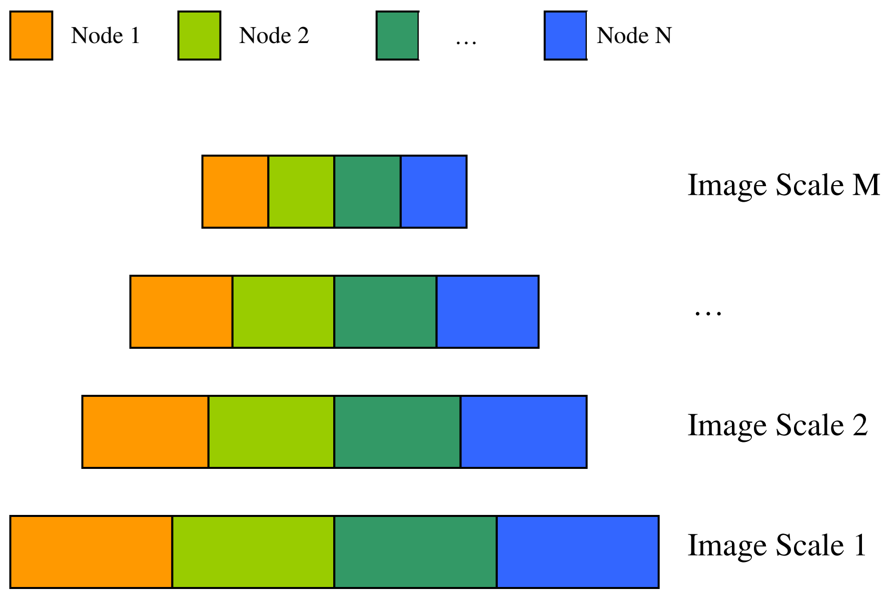


Figure 42 Illustration of the Hand Detection Component Parallelization

The hand tracking component keeps a set of the KLT features identified after the first detection, and then tracks them from one frame to the next using iterative pyramid matching methods. When some KLT features are lost between two consecutive frames, which is common in hand tracking, the hand tracking component is also responsible to identify substitute feature points to maintain the size of the KLT feature pool. The hand tracking component takes 10-20 milliseconds to calculate 30-50 KLT features across two VGA (640 by 480) resolution frame images.

Unlike the haar-feature calculation process performed by the hand detection and recognition components, KLT feature tracking does not calculate as many feature candidates. The pyramid iterative method usually has a limited number of levels (the default levels value used by OpenCV is 3), and just track the features without feature loss is very fast. At the same time, either KLT feature tracking or replenishing need to be performed over the whole frame image, rather than a sub-region because each level of the pyramids needs to be obtained from the whole frame image. Considering these factors, cost introduced by communication overhead of parallel KLT feature tracking will be at least comparable to the speedup gained, if the cost is not outrunning the gains. These considerations make cluster computing for KLT feature tracking less attractive and I decide not to incorporate this in the scalable computing solution.

The posture recognition component scans a hotspot surrounding the tracked hand position. If the hand hand performs a predefined posture, the posture recognition component reports a match. The template matching method in the posture recognition component is the same as the one used by the hand detection component. The main difference between the recognition scanning and the detection scanning is that the hotspot used by the recognition component is static in size but dynamic in location, i.e. in recognition task the hand tracker always scans a fixed-size area around the tracked hand position. While the hotspots used by the detection component are static in both size and location.

The data partitioning method used by the hand detection component, which divide the whole frame image into multiple hotspots, does not apply to the recognition component because the only hotspot needed is the one surrounding the tracked hand position. In this case, I have to look at other parameters that could be used to divide the computation task. Look deeper into the scanning algorithm, the scanning scales could be identified as such a parameter. Figure 43 shows the proposed parallelization solution for posture recognition.

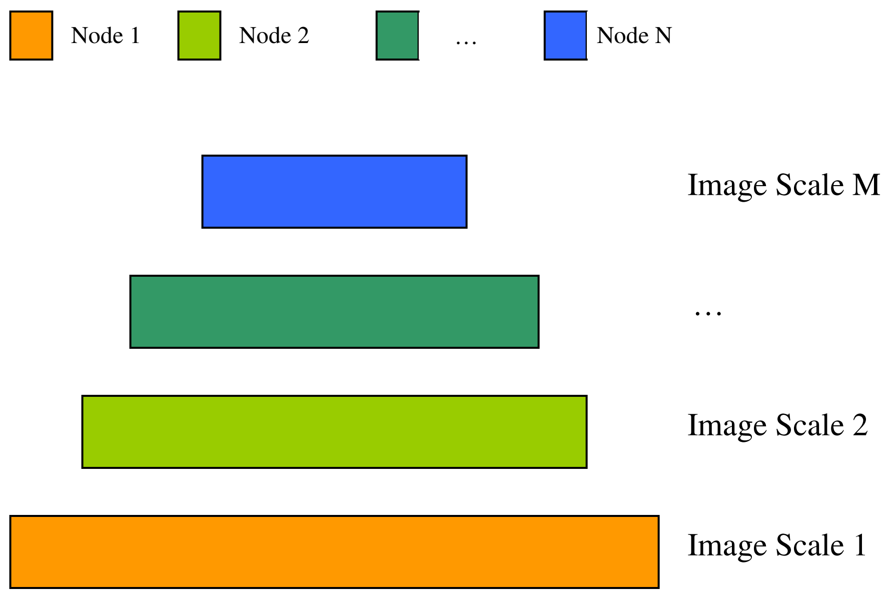


Figure 43 Illustration of the Posture Recognition Component Parallelization

One important point to be addressed is that simple divide-by-scales task partition will not be able to achieve good load balancing results, nor even close to this goal. In Section 6.3.2 I give the details of a load balancing algorithm that solves this problem efficiently.

6.3. Cluster Computing for Hand Tracker Components

I use a computer cluster to run the hand tracker detection and recognition components in parallel. These two components solve same type of scanning problems, and the partitioning parameters are scanning area and scales. Instead of using standard cluster computing middleware, such as Message Passing Interface (MPI), this work develops its own message-based parallelization solution, build on the QUANTA transport library [72].

Two unique technical aspects of this parallelization solution are proposed and implemented: first, a novel data structure, called scanning node tree. The scanning node tree is devised specifically for scanning problems. Second, a dedicated load balancing algorithm, based on the incorporation of both scanning area and scales.

6.3.1. Tree-based Node Organization

I organized the computation nodes in the infrastructure into a fanning-out tree. I named this tree the scanning node tree and defined it as a data structure that has the following characteristics:

- Can have one or more levels.
- Each tree node represents a node in the computation cluster and keyed by the physical node's IP address.
- Each tree node can have one or more children, or do not have children at all.
- Each tree node belongs to one and only one of node types: head node type and worker node type. There can be only one head node in the tree, which is the root. There can be multiple worker nodes.
- The head node has children which are worker nodes. Each worker node can also have children that are other worker nodes.
- The link between two nodes is defined as a two-way network link that allows transmission of data and command messages.

- Each node has two attributes: scanning area and scanning scales. A child's scanning area and scanning scales have to be a subset of it parent node.

Figure 44 illustrates the logical presentation of the scanning tree. In such as tree, when a node is assigned a scanning task, it partitions the task and then assigns the sub-tasks to all its children, if there are any. The node does not need to know whether its children will propagate the sub-task again or not, nor does it need to know or instruct the children about task partition strategy at the children nodes. At the same time, each child only fulfills the sub-task assigned to itself and do not need to know the way its siblings work. The benefit of the scanning node tree is that hand tracker tasks can be pre-partitioned at the global level, while at each local node it is sufficient to have just a local list of the children, and maintains communication link only with its parent and children. If the configuration of a certain node is later changed, it does not affect any other nodes except its direct parent and children.

In implementation, the scanning node tree is stored in the format of a vector, where each vector element is a quadruple {IP, AREA, SCALES, CHILDREN}. IP is the IP address of the node if it is a worker. When the node itself is the head node, I define the IP field to be DETECTION_HEAD for the root of the detection scanning node tree, and the IP field to be RECOGNITION_HEAD for the root of the recognition scanning node tree. The area is a quadruple {left, top, right, bottom} in percentage presentation of the whole frame image, for example, area {0, 0, 0.5, 0.5}, when applied on a VGA (640 by 480) resolution frame image, designates the area {0, 0, 320, 240). The scan scales specifies the start scan and stop scan scales over the image area designated by AREA. For example, {1.0, 8.0} means that the matching starts at the template size, and ends at 8.0 times of the template size. CHILDREN is a vector that stores the children's IP address of the node.

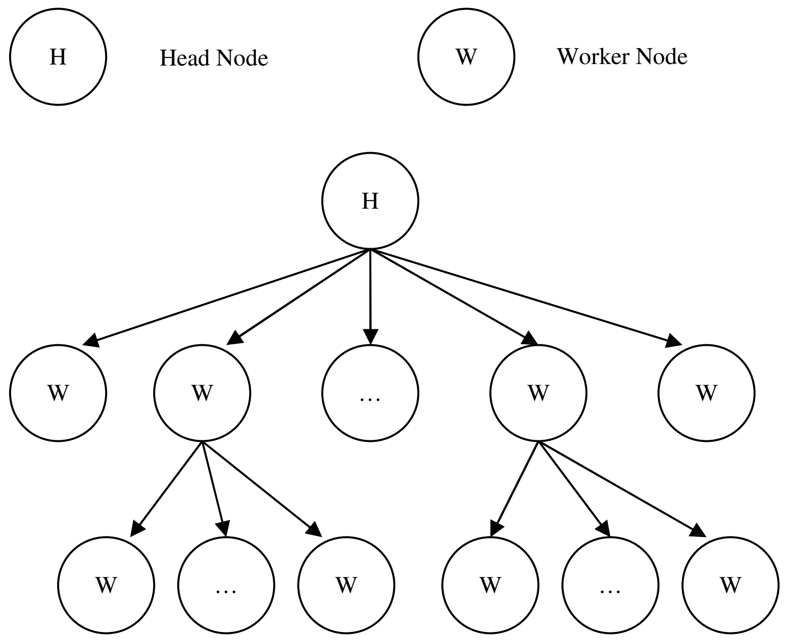


Figure 44 Logical Presentation of the Scanning Node Tree

Table 6 shows the example of the implementation of a scanning node tree. This is a detection tree, where the head node is DETECTION_HEAD. The tree has two levels. The head node has two children, one has the IP address 10.0.8.121 and the other has the IP address 10.0.8.122. Node 10.0.8.122 furthermore has two children, which are node 10.0.8.123 and node 10.0.8.124. It can be observed that the head node partitions its workload based on the scan area. For node 10.0.8.122, it applies a different partitioning strategy and partitions the workload based on scales. Node 10.0.8.122 and its two children all scan the same image area, but at scales {1.0, 2.0}, {2.0, 4.0} and {4.0, 8.0} respectively.

Table 6 Physical Implementation of the Scanning Node Tree

Node IP	Scan Area	Scan Scales	Children Nodes
DETECTION_HEAD	(0, 0, 0.5, 0.5)	(1.0, 8.0)	{10.0.8.121, 10.0.8.122}
10.0.8.121	(0, 0.5, 0.5, 1.0)	(1.0, 8.0)	NULL
10.0.8.122	(0.5, 0, 1.0, 0.5)	(1.0, 2.0)	{10.0.8.123, 10.0.8.124}
10.0.8.123	(0.5, 0, 1.0, 0.5)	(2.0, 4.0)	NULL
10.0.8.124	(0.5, 0, 1.0, 0.5)	(4.0, 8.0)	NULL

By using the vector-based implementation, the tree can be traversed from vector item 0. For each vector item Vi, it knows its children by looking at CHILDREN (Vi). However, items in CHILDREN (Vi) are IP addresses instead of vector index. For this purpose, I construct a IP-to-ID map together with the tree. When each node is created, I push it back into the vector, record its IP address and vector index, and put a tuple {IP, ID} into the IP-to-ID map. Table 7 shows the IP-to-ID map constructed from the tree shown in Table 6. With this table I can iterate the sub-tree of the tree shown in Table 6 from any node. For example, from the DETECTION_HEAD:

Start Tree Iteration:

CHILDREN (0) = {10.0.8.121, 10.0.8.122}

IP_to_ID (10.0.8.121) = 1   IP_to_ID (10.0.8.122) = 2

CHILDREN (1) = {NULL}   CHILDREN (2) = {10.0.8.123, 10.0.8.124}

IP_to_ID (10.0.8.123) = 3   IP_to_ID (10.0.8.124) = 4

CHILDREN (3) = NULL,   CHILDREN (4) = NULL.

End Tree Iteration.

Table 7 IP-to-ID Map Maintained by the Scanning Node Tree

IP	ID
DETECTION_HEAD	0
10.0.8.121	1
10.0.8.122	2
10.0.8.123	3
10.0.8.124	4

In this hand tracker, the scanning node tree is defined in the configuration file of the tracker. Two trees are defined, i.e. the detection tree and the recognition tree, in two different vectors. The HW_DETECTION_CLASSIFIERS <number of detection nodes> defines the size of the detection vector, while the HW_RECOGNITION_CLASSIFIERS <number of recognition nodes> defines the size of the recognition vector. For each node, it has one HW_CLASSIFIER_TYPE definition, specifies whether it is a head node or a worker node, and how many children it has. The node also has a HW_CLASSIFIER_NAME definition specifies the classifier file that node uses. Then follows is the HW_CLASSIFIER_HOTSPOT definition describing the scan area in percentage of the whole frame image. Next is the HW_CLASSIFIER_SCALES definition of the start and stop scales. At the end of the node definition clauses are zero or more HW_CLASSIFIER_CHILD definitions that specify the IP address of each node. Table 8 gives the full text of such a configuration file.

Table 8 the Scanning Tree Defined in Tracker Configuration File

```
HandWand Configuration 1.0

#Number of detection classifiers
HW_DETECTION_CLASSIFIERS 4
HW_CLASSIFIER_TYPE HEAD 3 children
HW_CLASSIFIER_NAME buttons.classifier
HW_CLASSIFIER_HOTSPOT x=0.0->0.5, y=0.0->0.5
HW_CLASSIFIER_SCALES 1.0->8.0
HW_CLASSIFIER_CHILD 10.0.8.121
HW_CLASSIFIER_CHILD 10.0.8.122
HW_CLASSIFIER_CHILD 10.0.8.123
# yorda1-10
HW_CLASSIFIER_TYPE WORKER 0 children
HW_CLASSIFIER_NAME buttons.classifier 10.0.8.121
HW_CLASSIFIER_HOTSPOT x=0.5->1.0, y=0.0->0.5
HW_CLASSIFIER_SCALES 1.0->8.0
# yorda2-10
HW_CLASSIFIER_TYPE WORKER 0 children
HW_CLASSIFIER_NAME buttons.classifier 10.0.8.122
HW_CLASSIFIER_HOTSPOT x=0.0->0.5, y=0.5->1.0
HW_CLASSIFIER_SCALES 1.0->8.0
# yorda3-10
HW_CLASSIFIER_TYPE WORKER 0 children
HW_CLASSIFIER_NAME buttons.classifier 10.0.8.123
HW_CLASSIFIER_HOTSPOT x=0.5->1.0, y=0.5->1.0
HW_CLASSIFIER_SCALES 1.0->8.0

HW_RECOGNITION_CLASSIFIERS 2
HW_CLASSIFIER_TYPE HEAD 1 children
HW_CLASSIFIER_NAME buttons.classifier LOCAL
HW_CLASSIFIER_HOTSPOT x=0.0->0.5, y=0.0->0.5
HW_CLASSIFIER_SCALES 1.0->1.0
HW_CLASSIFIER_CHILD 10.0.8.124
# yorda4-10
HW_CLASSIFIER_TYPE WORKER 0 children
HW_CLASSIFIER_NAME buttons.classifier 10.0.8.124
HW_CLASSIFIER_HOTSPOT x=0.0->0.5, y=0.0->0.5
HW_CLASSIFIER_SCALES 1.2->8.0

HW_NUM_BUTTONS 3
```

6.3.2. <u>Load Balancing Algorithm</u>

In the previous sections I describe that the work load partitioning over tree nodes are based on scan areas and scales. In this section I look at the partitioning problem in more detail. Three aspects of the load partitioning and balancing are discussed: computational overhead introduced by scan area overlapping; unbalanced workload at each scale level, and communication cost introduced by image propagation.

6.3.2.1. Scan Area Overlapping

In Section 6.2 the scan area partitioning is briefly introduced. To scan a frame image that used to be scanned by a single node with multiple nodes, the scan areas of these nodes cannot be disjoint but have to be overlapped. Otherwise, image patches on the boundary of the defined scan areas will be missed and the tracker will have scanning loss if the predefined pattern exists in these image patches. To further illustrate this problem, Figure 45 shows two predefined scan areas who shares a common boundary. In this figure, W_{d1} and W_{d2} are the defined width of scan area 1 and scan area 2. However, if I just scan areas covered by defined scan area 1 and defined area 2, any template occurrences within the middle pinked area will not be matched because none of scan node 1 or scan node 2 will be seeing the whole template instance but just part of it. To make sure template occurrences in the middle pink area is also processed, both scan node 1 and scan node 2 need to expand their scan area into the neighbor's defined scan area, thus overlaps are introduced.

In Figure 45 it can be observed that scan node 1 actually needs to scan an area that has the width of W_{a1} instead of W_{d1}. For same reason, scan node 2 actually needs to scan an area that has the width of W_{a2} instead of W_{d2}. The overlap width ($W_{di} – W_{ai}$) is related to the template width and the scanning scales. At its minimum, the overlap width has to be equal to the product of the template width and the current scanning scale.

Because each individual node in the scanning tree has overlapped scan area larger than its defined scan area, its workload cannot be simply calculated by the percentage of its defined scan area size over the whole frame image size, but has to be calculated by the percentage of its actual scanned area size over the whole frame image size. As the template size itself is an invariant, the larger scan scale is, the larger is this overhead. Quantitatively, the actual scanning workload/defined workload ratio R_{ad} at a specific node can be calculated using the following equation:

$$R_{ad} = \frac{(W_d + W_t * S)(H_d + H_t * S)}{W_d * H_d}$$

Where: W_d is the defined scan area width, W_t is the template width, S is the scanning scale, H_d is the defined scan area height and H_t is the template height.

Equation 12 Calculation of the Actual/Defined Workload Ratio

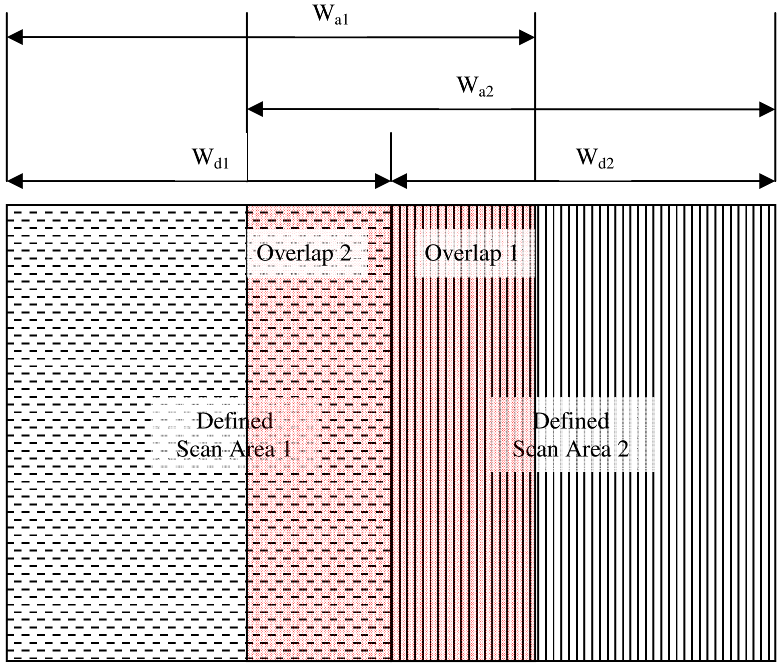


Figure 45 Scan Areas Overlapping to Avoid Scanning Loss and Overhead Introduced

To get more intuitive understanding of the overhead, Figure 46 plots the overhead ratio introduced by overlapping. Here W_d is 320, H_d is 320, W_t is 25 and H_t is 25. S varies from 1.0 to 8.0. It is very clear that the overhead ratio increases with the scale level. When scale level is 6 and beyond, the overhead ratio exceeds 2.0, which means the processing workload of the overlapped area has exceeded the processing workload of the defined area.

According to the goals I set for the scalable computing solution, this is an unfavorable outcome and will be address by the scanning node tree calculation algorithm in Section 6.3.2.3.

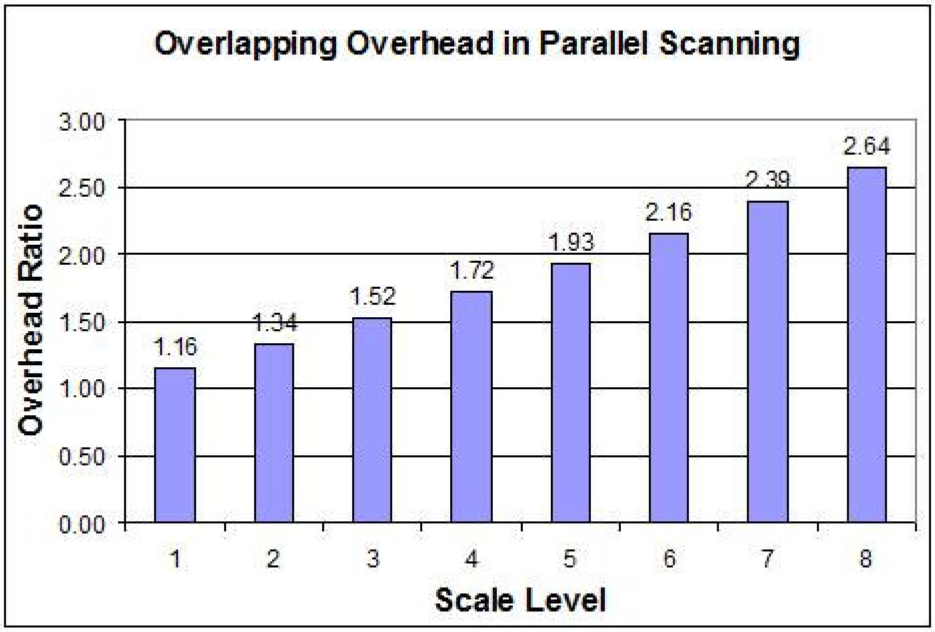


Figure 46 Overlapping Overhead in Parallel Scanning for 320*240 Defined Scan Area and 25*25 Template

6.3.2.2. <u>Unbalanced Workload at Scale Levels</u>

The previous section discussed the challenge from scan area partitioning for load balancing. In this section challenges from scale levels partitioning are studied. Because in a scanning process multiple scale levels are used, a simple parallelization algorithm will assign each (or a set of) scale level to a different cluster node. For example, scale 1.0 scanning task can be assigned to node 1. Scale 2.0 scanning task can be assigned to node 2, etc. However, this partition method will not be able to get satisfying speedup results. Since the workloads at different scale levels are unbalanced.

As I know that the atomic operation in a scanning process is to match a template (no matter which scale it is at) to an image patch of the frame image of the same size. The complexity of this operation across different scale levels is a quasi constant, because each matching evaluation compares nearly the same amount of haar features. What really determines the scanning complexity at a specific scale level is the number of scans needs to be completed. The number of scans can be calculated using the equation below:

$$NumberOfScans = [(W_i - W_t * S)/(StepX * S)] * [(H_i - H_t * S)/(StepY * S)]$$

Where: Wi is the scan area width, Hi is the scan area height, Wt is the template width, Ht is the template height, S is the scanning scale, StepX is the X translation between two consecutive scans, StepY is the Y translation between two consecutive scans.

Equation 13 Number of Scans Needed Calculation

To further illustrate the relationship between number of scans needed and the scale level. Let’s set the scan area size to be 320 by 240, the template size to be 25 by 25, X translation is 2 and Y translation is 3. In the mean time I vary scale level from 1.0 to 8.0. Figure 47 plots the number of scans needed at these scale levels. Please note that the Y axis is plotted in logarithm scales. It can be clearly seen that the number of scans needed decreases drastically when the scan scale increases. If I simply assign each one (or a set of) scale value to a node, the work load will be very unbalanced.

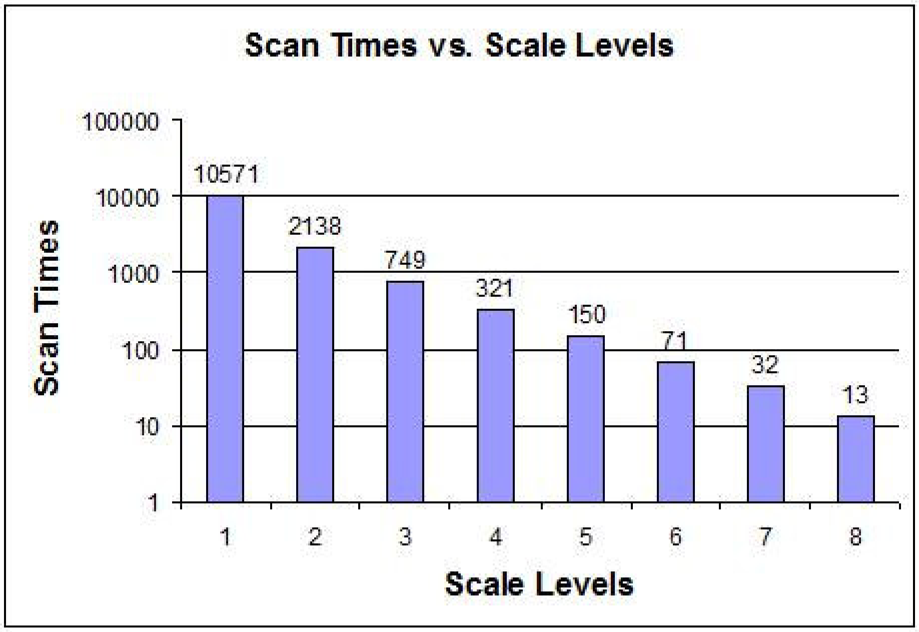


Figure 47 the Number of Needed Scans versus the Scale Level

6.3.2.3. <u>Communication Cost Between Cluster Nodes</u>

Two reasons make communication cost another important factor to consider in load balancing. The first is message latency introduced by network transport. Because in the hand tracker application, each frame image is transmitted to all processing nodes for parallel processing and the total frame latency is a scarce resource, it is important to control communication costs within a certain boundary. In the tree structure node organization, message passing is parallel among different levels of the tree, so the total latency can be measured by the largest one among the tree levels.

Another reason is that hand tracker is usually not the main application running on the computer cluster but as an auxiliary application to facilitate users' interaction with other applications. Under many occasions these other application themselves are scalable applications as well and consumes a large portion of the processing and communication resources of the computer cluster. If the hand tracker takes too much communication cost, the other applications' performance will be hindered.

The number of frame transmissions can be mapped to the number of edges in the tree. That is, the total number of communication cost across the network is the total number of edges in the tree multiply the frame size; the total number of communication cost from a certain node is the total number of its children multiply the frame size; and the total number of communication cost at a certain tree level is the total number of nodes in the next level multiply the frame size.

6.3.2.4. <u>Scanning Node Tree Calculation</u>

To sum up all the discussions in the previous three chapters, the following observations can be made:

1. Given defined scan area size and template size, the larger the scan scale is, the larger the overlapping overhead is.

2. Given defined scan area size and template size, the larger the scan scale is, the less the needed number of scan is.

3. The communication cost at a specific node is proportional to the number of children of the node. If the head is the node that has most children in the tree, the total latency introduced by communication cost is proportional to the number of the workers that are direct children of the head.

These observations form the basis of the design consideration of the load balancing algorithm:

- Based on observation 2, I group multiple large scan scale levels into a group to aggregate the workload. And partition the workload at small scan scale levels into sub-tasks further processed at next tree level.

- The partition strategy for the workload at a certain small scan scale level is to partition the scan area into multiple sub-scan areas that have overlaps with each other. Because the overlapping overheads are small at small scan scale levels, this partition makes sure that only reasonable overlapping overhead is introduced.

- When partitioning the workload of a certain node into multiple sub-tasks to be processed by its children, I set the arbitrary limit of the number of children to be three, the purpose is to control the communication cost from this specific node.

- Because computation workload is only determined by the task itself but communication workload is determined by both the task itself and the computer cluster configuration, I do not incorporate the communication cost into quantified load balancing calculation but calculate it separately.

To further illustrate this strategy, assume I am going to scan a 320 by 240 resolution image area with a recognition tree. The template size is 25 by 25, X translation is 2, Y translation is 3, and scale levels are from 1.0 to 8.0 and increase at a step of 1.2.

First, I calculate the number of scans at different scale levels and put them into Table 9.

Table 9 Workload Partition Based on Scale Levels[9]

Node ID	Scan Area	Scan Scales	Number of Scans Needed
1	(0, 0, 320, 240)	(1.0, 1.0)	12800
2	(0, 0, 320, 240)	(1.2, 1.2)	8889
3	(0, 0, 320, 240)	(1.44, 1.44)	6173
4	(0, 0, 320, 240)	(1.73, 1.73)	4287
5	(0, 0, 320, 240)	(2.07, 2.07)	2977
6	(0, 0, 320, 240)	(2.49, 2.49)	2067
7	(0, 0, 320, 240)	(2.99, 2.99)	1436
8	(0, 0, 320, 240)	(3.58, 3.58)	997
9	(0, 0, 320, 240)	(4.30, 4.30)	692
10	(0, 0, 320, 240)	(5.16, 5.16)	481
11	(0, 0, 320, 240)	(6.19, 6.19)	334
12	(0, 0, 320, 240)	(7.43, 7.43)	232

Then, I group the scanning tasks at large scale levels. In this example, scan scales from 2.49 to 7.43 are grouped and processed by node id 6. The number of scans needed at node id 6 is now 6239, as shown by Table 10.

Table 10 Workload Partition after Scan Scale Grouping at Large Scale Levels

Node ID	Scan Area	Scan Scales	Number of Scans Needed
1	(0, 0, 320, 240)	(1.0, 1.0)	12800
2	(0, 0, 320, 240)	(1.2, 1.2)	8889
3	(0, 0, 320, 240)	(1.44, 1.44)	6173
4	(0, 0, 320, 240)	(1.73, 1.73)	4287
5	(0, 0, 320, 240)	(2.07, 2.07)	2977
6	(0, 0, 320, 240)	(2.49, 7.43)	6239

It can be seen that node id 1 and node id 2 are still taking too much workload compare to node id 6. Because node id 1 and node id 2 fulfills scan tasks at small scan scales (1.0 and 1.2 respectively), the overlapping overhead introduced by scan area partitioning will be moderate. Based on this observation, I further divide the workload at node id 1 and node id 2 by scan area, each into four sub-tasks. Node id 1 offsets its workload to node id 7, node id 8 and node id 9. In the mean time, node id 2 offsets its workload to node id 10, node id 11 and node id 12. Table 11 shows the new workload partitioning results.

[9] For simplifying purpose, number of scan times are directly calculated with $(W_i * H_i) / (W_t * H_t)$. Without special instruction, the number of scan times will be calculated with this in the text now and after.

Table 11 Workload Partition after Scan Area Partition at Small Scale Levels

Node ID	Scan Area	Scan Scales	Number of Scans Needed
1	(0, 0, 160, 120)	(1.0, 1.0)	4416
2	(0, 0, 160, 120)	(1.2, 1.2)	3252
3	(0, 0, 320, 240)	(1.44, 1.44)	6173
4	(0, 0, 320, 240)	(1.73, 1.73)	4287
5	(0, 0, 320, 240)	(2.07, 2.07)	2977
6	(0, 0, 320, 240)	(2.49, 7.43)	6239
7	(160, 0, 320, 120)	(1.0, 1.0)	4416
8	(0, 120, 160, 240)	(1.0, 1.0)	4416
9	(160, 120, 320, 240)	(1.0, 1.0)	4416
10	(160, 0, 320, 120)	(1.2, 1.2)	3252
11	(0, 120, 160, 240)	(1.2, 1.2)	3252
12	(160, 120, 320, 240)	(1.2, 1.2)	3252

After the discussion of a specific example, I now describe the load balancing method in a formal algorithm. As the scanning node tree is created with the progress of workload partition, the final outputs of this algorithm include the scanning node tree in vector format. Also because the overlapping overhead and communication cost can be calculated during load balancing calculation, they are also included in the algorithm's outputs.

Algorithm Calculate_Tree

Input:

1. Size of the whole scan area (frame image for detection tree, and predicted hotspot for recognition tree): W_i and H_i.
2. Size of the template: W_t and H_t.
3. X translation and Y translation at each scanning step: StepX and StepY.
4. Start, stop and the stepping value of the scanning scales: S_{start}, S_{stop} and S_{step}.
5. The desired speedup: P.

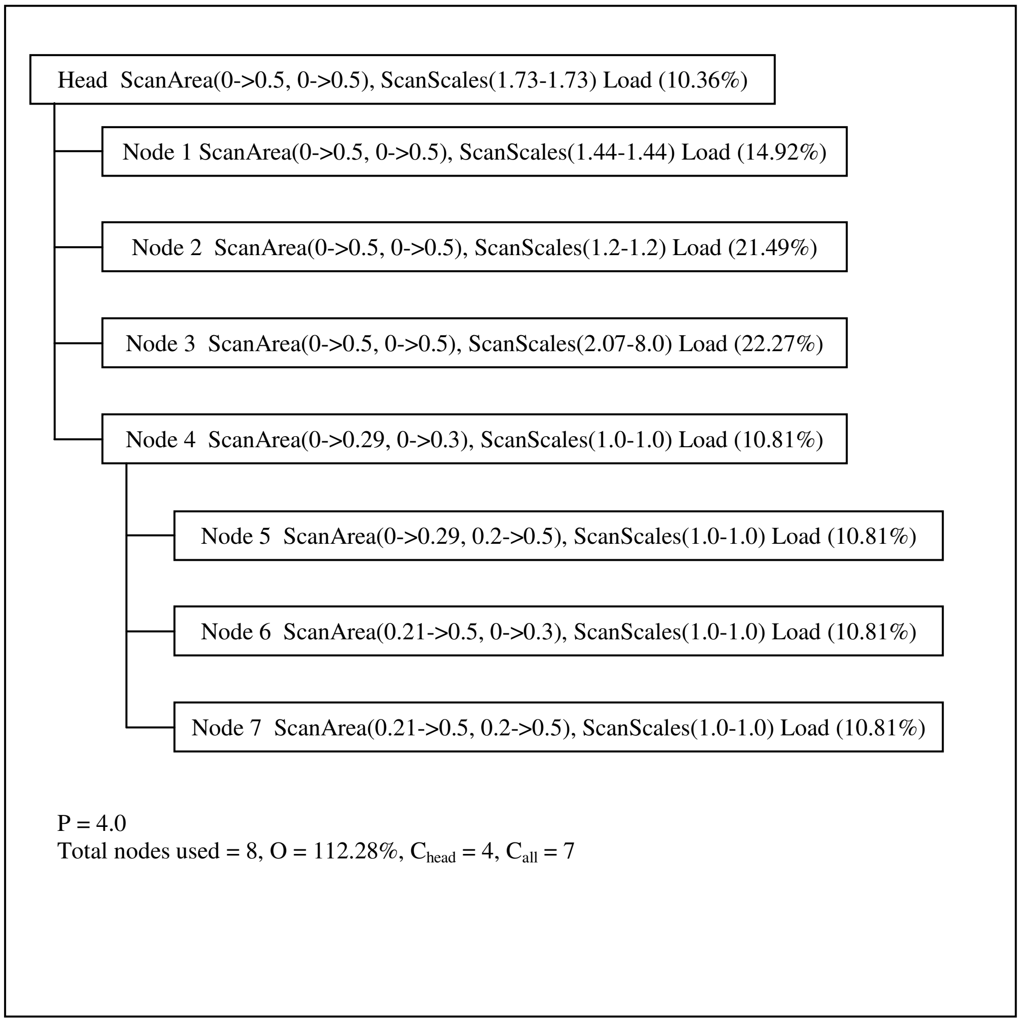
Output:

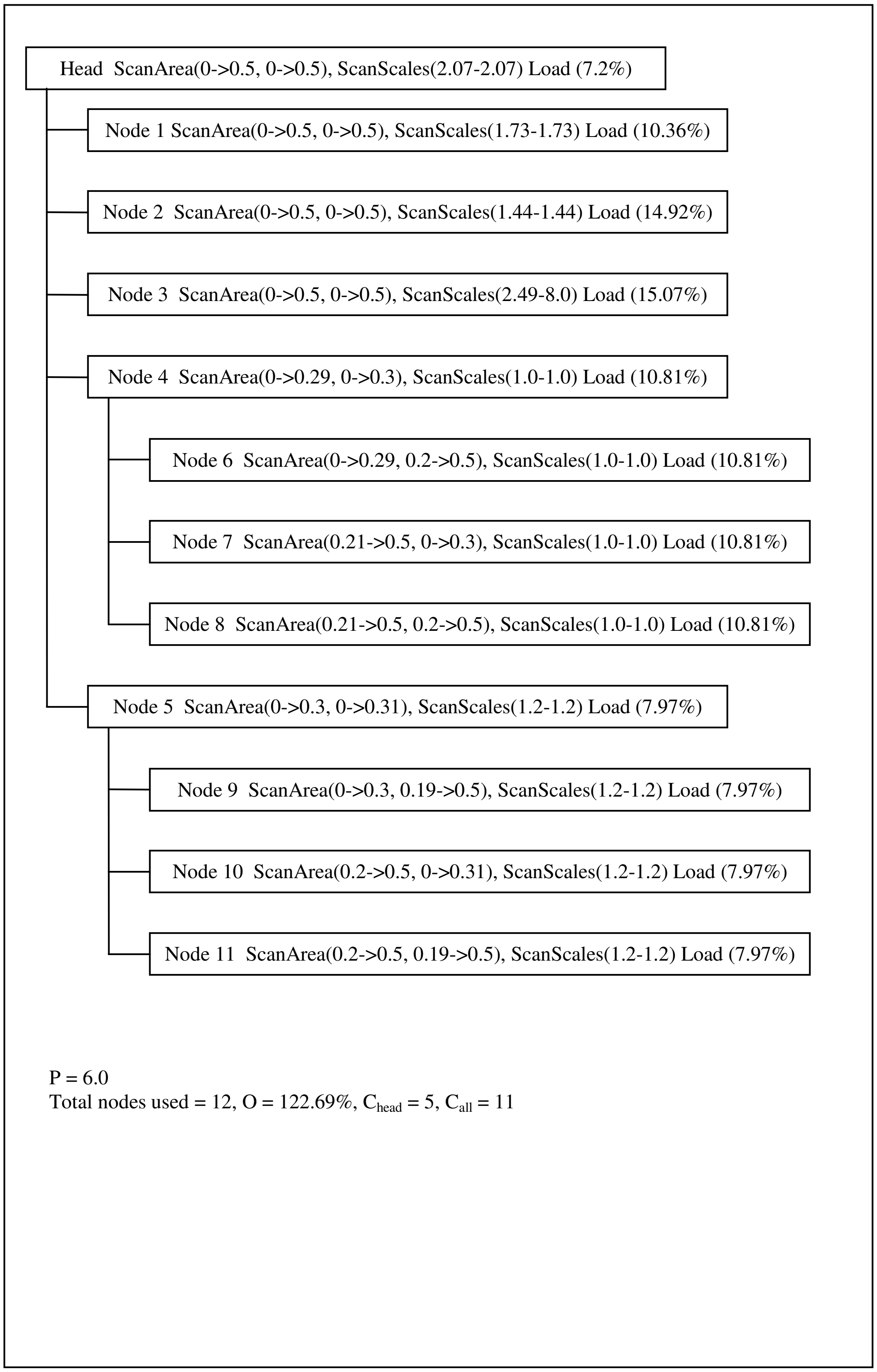
1. The scanning node tree with scan area and scales partitioned, in vector format: T_{scan}.
2. The efficiency estimation of overall overlapping overhead, communication cost of the head and communication cost across all the tree nodes: O, C_{head} and C_{all}.

Pseudo Code:

```
$T_{scan}$ = NULL;
 O = $C_{head}$ = $C_{all}$ = 0;
For (scale = $S_{start}$:$S_{stop}$)
    Calculate the number of scan times needed at scale and put in an array scale_level[];
End;
Calculate the sum of all elements in scale_level[], and then create two other arrays scale_level_percentage and scale_level_integrated_percentage;
For (i = 1: sizeof(scale_level)
    Scale_level_percentage[i] = scale_level[i]/sum;
    Scale_level_integrated_percentage = sum(scale_level_percentage[i:$S_{stop}$]);
End;
Iterate the arrary scale_level_integrated_percentage from the **END**, until find the first element k that has the value less or equal to 1/P;
Iterate the array scale_level_percentage_from the **START**, until find the first element j that has the value less or equal to 1/P.
Node.scanArea = {$W_i$, $H_i$};
Node.scales = {Scales[k-1].startScale, Scales[k-1].startScale};
Add Node to $T_{scan}$ as head node;
For i = j:k-2;
    Node.scanArea = {$W_i$, $H_i$};
    Node.scales = {Scales[i].startScale, Scales[i].startScale};
    Add Node to $T_{scan}$ as children of the head node;
End;
Group all scales after Scale[k] into a set, assign it to a node and add this node to $T_{scan}$;
Split all scales before Scale[j] based on scan area, and append the partitioned nodes to $T_{scan}$;
Sum up workload value calculated at all nodes and put this as O.
Sum up communication cost at the head and put this as $C_{head}$;
Sum up communication cost at all the nodes and put this as $C_{all}$;
Output $T_{scan}$, $C_{head}$, and $C_{all}$;
```

I create a utility application tree_calculator as the implementation of the algorithm described above. Below are two sample trees calculated using this utility application. In the 8-node parallelization, the mean percentage workload across all nodes is 14.03% and the standard deviation is 5.06%. In the 12-node parallelization, the mean percentage workload across all nodes is 10.22 and the standard deviation is 2.64%. It can be seen that the tree calculation algorithm is effective in fulfilling the load balancing task.


Head ScanArea(0->0.5, 0->0.5), ScanScales(1.73-1.73) Load (10.36%)
Node 1 ScanArea(0->0.5, 0->0.5), ScanScales(1.44-1.44) Load (14.92%)
Node 2 ScanArea(0->0.5, 0->0.5), ScanScales(1.2-1.2) Load (21.49%)
Node 3 ScanArea(0->0.5, 0->0.5), ScanScales(2.07-8.0) Load (22.27%)
Node 4 ScanArea(0->0.29, 0->0.3), ScanScales(1.0-1.0) Load (10.81%)
Node 5 ScanArea(0->0.29, 0.2->0.5), ScanScales(1.0-1.0) Load (10.81%)
Node 6 ScanArea(0.21->0.5, 0->0.3), ScanScales(1.0-1.0) Load (10.81%)
Node 7 ScanArea(0.21->0.5, 0.2->0.5), ScanScales(1.0-1.0) Load (10.81%)
P = 4.0
Total nodes used = 8, O = 112.28%, C_head = 4, C_all = 7



```
Head  ScanArea(0->0.5, 0->0.5), ScanScales(2.07-2.07) Load (7.2%)
    Node 1 ScanArea(0->0.5, 0->0.5), ScanScales(1.73-1.73) Load (10.36%)
    Node 2  ScanArea(0->0.5, 0->0.5), ScanScales(1.44-1.44) Load (14.92%)
    Node 3  ScanArea(0->0.5, 0->0.5), ScanScales(2.49-8.0) Load (15.07%)
    Node 4  ScanArea(0->0.29, 0->0.3), ScanScales(1.0-1.0) Load (10.81%)
        Node 6  ScanArea(0->0.29, 0.2->0.5), ScanScales(1.0-1.0) Load (10.81%)
        Node 7  ScanArea(0.21->0.5, 0->0.3), ScanScales(1.0-1.0) Load (10.81%)
        Node 8  ScanArea(0.21->0.5, 0.2->0.5), ScanScales(1.0-1.0) Load (10.81%)
    Node 5  ScanArea(0->0.3, 0->0.31), ScanScales(1.2-1.2) Load (7.97%)
        Node 9  ScanArea(0->0.3, 0.19->0.5), ScanScales(1.2-1.2) Load (7.97%)
        Node 10  ScanArea(0.2->0.5, 0->0.31), ScanScales(1.2-1.2) Load (7.97%)
        Node 11  ScanArea(0.2->0.5, 0.19->0.5), ScanScales(1.2-1.2) Load (7.97%)
```

P = 6.0
Total nodes used = 12, O = 122.69%, C_{head} = 5, C_{all} = 11

Figure 48 Two Sample Outputs of the Tree Calculation Algorithm

6.3.3. Hand Detection Process

In Section 6.3.1 and Section 6.3.2 I present the static part of the cluster computing solution for hand tracker tasks, i.e. the node organization of the computation cluster and the load balancing algorithm. In this and the following sections I introduce in detail the dynamic part of the cluster computing solution, both for the hand detection and for the posture recognition processes. Two different hand tracker applications are launched on the processing node depends on whether the node is of type "head" or "worker".

6.3.3.1. Head Node Side

On a head node, the application reads in the detection tree configuration at launch time, and then set the scan area and scales of itself based on the configuration settings. The head node does not keep the scan area and scales information of any other nodes, including its children. After the initialization with configuration settings, the head sends out an "INIT_REQUEST" message to all its children, and blocks to wait for the children to respond "INIT_RESULT" messages. Once all "INIT_RESULT" messages have been received from the children, the head node is ready to start the detection task.

During the detection process, frame images are pumped in from the camera one after another. Each time when the head node acquires a frame image, it forms a "DETECTION_REQUEST" message and sends it to all the children. The body of the message has the image data of the frame image. Then the head node process the frame image with its own scan settings, and put any match result into a vector. The head node then receives all the scanning result from its children in "DETECTION_RESULT" messages, and put these results into the match vector. If the match vector contains one or more match results, a "best match" candidate is used as the sole result of this scan.

When the hand tracker application on the head node is going to exit, it sends out a "CLOSE_REQUEST" to all of its children. For this message, the head node does not expect to receive result messages back.

6.3.3.2. <u>Worker Node Side</u>

On the worker node side, when the application launches, it reads in the configuration file and sets its own scan area and scales according to definitions in the configuration. The worker does maintain such information for its children, if there are any.

The worker node then goes into a block waiting state for any message from its parent. It is worth noting here that a worker node's parent could be either the head or another worker note. However, a worker node's child is always a worker node. When the worker node receives a "INIT_REQUEST" message, it propagates this message to all its children, if there are any, wait for "INIT_RESULT" response message from the children, and then send "INIT_RESULT" back to its parent.

When the worker node receives a "DETECTION_REQUEST" message, it first relays this message, as well as the image data come with it, to all its children. The worker node then performs its own scanning task based on the configuration definition, and saves any matching result into a match vector. After performing its own scanning, it receives all match results from its children, puts them into the match vector, and chooses the best match to send back to its parent through the "DETECTION_RESULT" message.

There are couple empirical rules that should be followed when a node is offsetting workload to its children. First, sending the request message together with image data should always be undertaken before the local scanning happens. Second, the node should always pick the smallest sub-task to be processed locally and offset the larger sub-tasks out. Third, when sending request messages out to children, always send the largest sub-task first. The reasoning behind these empirical rules are: 1) the children nodes can start processing at the earliest time; 2) the parent node's communication cost is compensated by lighter processing loads; and 3) the child who is assigned the largest subtask is the first to start processing.

6.3.4. <u>Posture Recognition Process</u>

Recognition process, just like the detection process, is also processed by a scanning node tree. The two processes share a large part in common but also have differences from each other. The main difference is that for the recognition process, the overall scan area itself is a sub-area of the frame image and might change among frames. Thus each node has the responsibility to notify its children about the updated scan areas.

6.3.4.1. <u>Head Node Side</u>

On the head node side, when it reads in the configuration file to construct the recognition tree, both the scan area and scales information of itself and all its children are maintained. This is different from the head node behavior in the detection process, where the head node just maintains the scan area and scales information of its own.

During each recognition iteration, the head node acquires the frame image and calculates the recognition scan area. It then compares this scan area with the pre-defined scan area from the configuration, and calculates the offset (dx, dy) between the predefined scan area and the current-frame recognition scan area. After the offset is calculated, the head node applies it over all the pre-defined scan area of its children and then sends the "RECOGNITION_REQUEST" messages to all its children. The difference between the

"DETECTION_REQUEST" message and the "RECOGNITION_MESSAGE" is that the latter has scan area and scales information in its header. The head node then expects "RECOGNITION_RESULT" message back from its children.

6.3.4.2. <u>Worker Node Side</u>

On the worker node side, when the application launches it reads in the configuration settings of itself and any children from the configuration file and set them accordingly. The worker node then expects the "RECOGNITION_REQUEST" message.

Once the worker node receives the "RECOGNITION_REQUEST" message, it retrieves the scan area and scales information out of the message header and calculate offset (dx, dy) from its pre-defined scan area. The worker node then sets all its children's scan area and scales according to (dx, dy) and their pre-defined settings, and sends corresponding "RECOGNITION_REQUEST" messages out to the children. The worker then performs its local scanning task and put any match results into a match vector. After the local scanning, the worker receives all match results from its children, put them into the match vector and then send back to the head.

6.4. <u>Evaluation Results</u>

I evaluated the scalable computing techniques on a 32-node computer cluster at EVL-UIC. Each node is a 64bit architecture with 2 AMD Opteron processors (2 Ghz, Model 246), 4GB of DDR 400 Mhz RAM, 1.5TB of shared local storage over parallel filesystem (PVFS) [8x250GB on a RAID5 configuration] and 250GB of local storage. Interconnections at each node consist of 3 gigabit network interfaces: on-board gigabit, single-mode fiber gigabit and multi-mode fiber gigabit.

The metrics I am interested in are in line with the design goals listed in Section 7.1. These metrics are: throughput improvement for frame image processing; processing time reduction and processing speed up at reasonable node costs; balanced workload across cluster nodes; and lastly, low communication cost. In the following sections, I present the evaluation results over each of these metrics.

6.4.1. Improvement of Detection Component Throughput

Because of processing time limitations, the detection component implemented in Section 5.3.1 only monitors a hotspot of the size 320 by 240, the user has to put his/her hand into a designated area to be detected. In this experiment, I use four nodes to monitor a 640 by 480 frame image area. There are no scan area overlaps between any two nodes. The scan area partition is as follows:

Node 1: (0, 0, 320, 240); Node 2: (320, 0, 640, 240);

Node 3: (0, 240, 320, 480); Node 4: (320, 240, 640, 480);

According to the overhead calculation, the processing overhead is zero because the four scan areas have no overlaps. However, the latency caused by communication cost will create some impact on frame processing time, and in turn affect frame rate at detection state. I define a unit throughput to be the processing of one 320 by 240 scan area, thus, one frame processed equals to 4 throughput units. Two control conditions are defined, the first condition uses a single node to process the 320 by 240 area; while the second condition uses four nodes to process the whole frame image. Each condition is repeated 10 times, each time for 200 frames, under different lighting condition and camera field-of-view. Then the mean and stand deviations of throughput per second are recorded and shown in Figure 49.

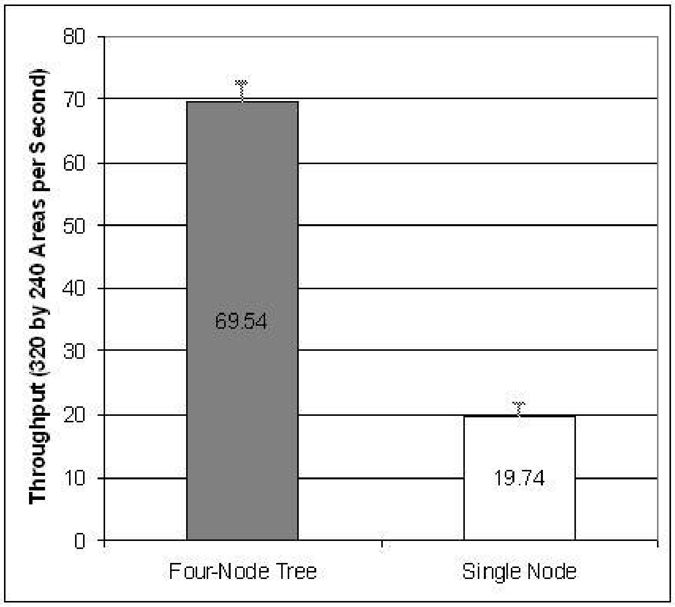


Figure 49 Throughput Improvements with Four Nodes in Hand Detection

Data plotted in Figure 49 shows that the hand detection throughput of original hand tracker is 19.74±2.02 units. When a four-node scanning node tree is used the throughput improved to 69.54±2.97 units. The throughput improvement is about 3.52 when a four-node scanning node tree is used. The gap between 3.52 and 4.0 is caused by image streaming latency from the head to the other three nodes.

6.4.2. Speedup of Recognition Component

In this section I study the speedup performance of the scanning node tree, for the hand recognition task. The reason to study the hand recognition task instead of the hand detection task is because that the former is susceptible for overlap overheads and normally more communication cost due to depth of the scanning node tree. Thus, the speedup value predicted by the tree calculation utility will not be exactly the same as the speedup observed. It is of interest to see the scanning node tree's performance in the field.

This study can be done by predicting different speedup values, and then use the tree calculated to measure actually speedup values, and then compare these two. The numerator to compute speedup is the time used when hand recognition is performed with a single node scanning node tree. I use frame 157 through 5524 of the pre-recorded 5597-frame video sequence, where the recognition task starts and ends. Tracking component's processing time is subtracted from overall frame latency to get the recognition component's processing time. To remove the noise of image display latency, each frame image is only processed but not displayed.

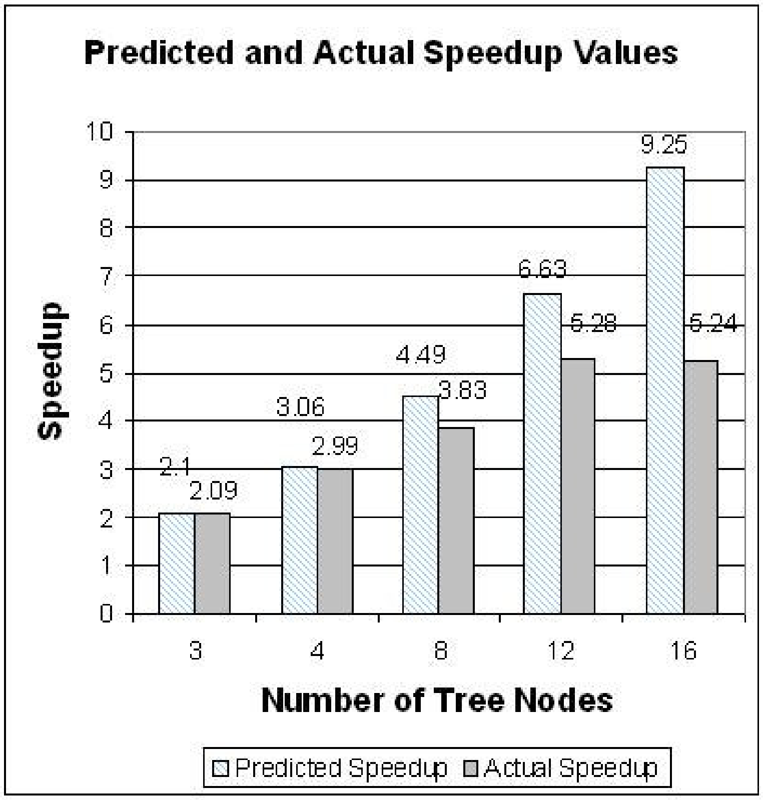


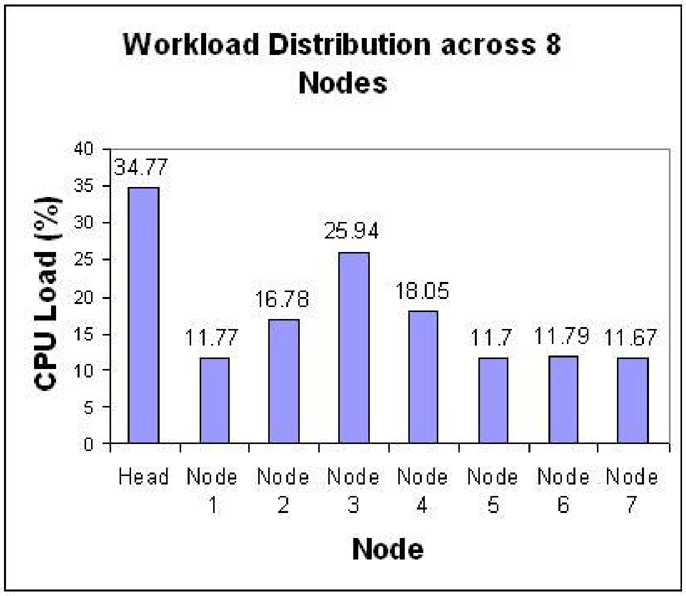
Figure 50 Predicted and Actual Speedup Values vs. Number of Tree Nodes

Figure 50 shows the actual speedup values recorded at predicted speedup values. It can be seen that the actual and predicted values are close to each other when the scanning node tree is composed of 12 or less nodes. For the 16-node scanning node tree, there is large discrepancy between the actual and predicted values. Also, the actual speedup value measured is smaller than the actual value when a 12-node scanning node tree is used. The reason is due to network transmission latency for image data broadcast. If the cost of communications can be ignored (e.g. for 3- and 4-node scanning node trees) or not significant (e.g. for 8- and 12-node scanning node trees), the predicted speedup performances of the tree are in line with the actual measured values.

6.4.3. Efficiency of Load Balancing Algorithm

In this section I evaluate the efficiency of load balancing algorithm in partition workloads across cluster nodes. To study this, I measure the CPU loads at each node in the scanning node tree, and compare them with the predicted workload allocations. The Linux utility top is used to record CPU workloads during hand recognition task. The experiments are undertaken over the pre-recorded 5597-frame video sequence.

Load balancing are tested under two scanning node tree configurations, one tree consists of 8 nodes and the other tree consists of 12 nodes. Figure 51 illustrates the CPU load data recorded under both configurations. It can be seen that the load recorded on the worker nodes confirms to the tree prediction well, that is, the workload exhibits a ramp curve for the head node's direct children and shows even values for the worker nodes who are children of other worker nodes. The head node demonstrates a much higher CPU load than predicted by the tree model. There are two reasons for this observation: first, in the tree construction the head node is usually the node that has the highest out degrees in the tree, thus it has more communication cost than the other tree nodes. Second, besides the recognition task, the head node also needs to fulfill tracking and frame image retrieval tasks. These tasks make the head node takes significant more workload than the other nodes.



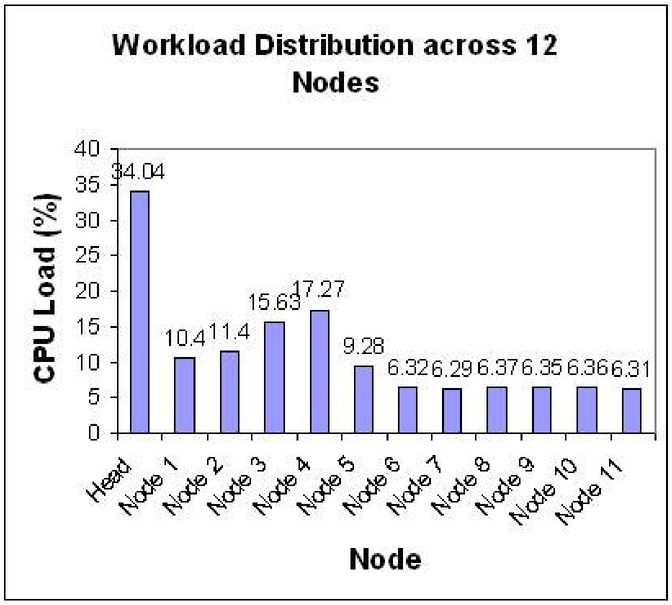


Figure 51 CPU Load Distributions over 8 (top) and 12 (bottom) cluster nodes

Recall that in the load balancing algorithm, when a node partitions the scan area and assign it to all its children, the workload allocation is even. In Figure 51, this means that node 4 should have same CPU load as node 5, 6 and 7 in the 8-node tree configuration. Similarly, node 4 should have same CPU load as node 6, 7, 8 and node 5 should have same CPU load as node 9, 10, 11 in the 12-node tree configuration. However, it could be seen that the parent node always consumes more CPU processing than its children nodes. The reason for this is because a worker node which has children needs to relay the frame image to its children, and the communication cost causes more processing at the node. Except for the unbalance discussed above, which are introduced by other processing tasks and communication costs, the load balancing algorithm achieves satisfying results comparing to its own predictions.

6.4.4. <u>Communication Cost Results</u>

In this section I evaluate the communication cost of the hand tracker at different scanning node tree configurations, which consists of 3, 4, 8, 12 or 16 nodes. The data is measured with two parameters: one is the data flow in and out from the head node during one frame, and the second is the number of frames processed every second. The bandwidth consumption at the head node is calculated from the product of these two parameters under difference tree configurations.

Under the 3-, 4-, 8-, 12- and 16-node tree configurations, the head node needs to send out 2, 3, 4, 5 and 6 frame images respectively during each frame processing. In the mean time, the frame per second is recorded to be 18, 21, 26, 35 and 33 respectively. Use these values to derive head node bandwidth, the results are shown in Figure 52.

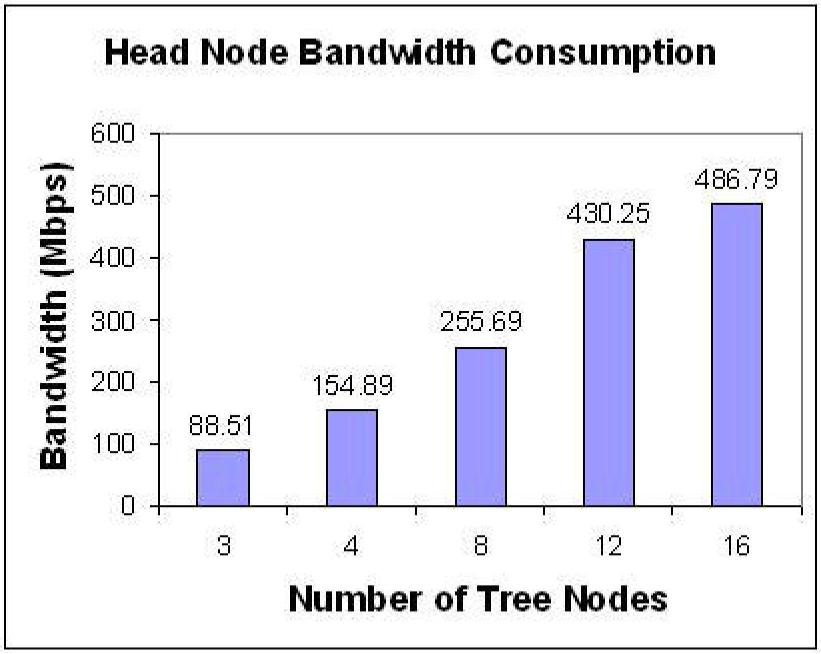


Figure 52 Network Bandwidth Consumption at the Head Node

It can be seen that with the increase of the number tree nodes, the communication cost of the head node also increases. Assuming the nodes are interconnected with Gbps network interfaces, then streaming frame images could take as high as 49% of the total bandwidth if not optimized. To further reduce the bandwidth consumption, either frame image compression should be performed at the application layer or the interconnect hardware and software need to be improved.

6.5. <u>Conclusions and Discussions</u>

In this chapter, the scalable computing techniques to improve the hand tracker's performance are presented. I first give the reasons why scalability is an important challenge to be address for the hand tracker, as well as the technical difficulties in achieving desired outcomes. Then an in-depth analysis of the opportunities for hand tracker performance improvements is conducted. I identify two parameters of the scanning task after the analysis as the keys in data- and computation-decomposition, that is, the scan area and the scan scales range.

After identifying the two task parameters for scalable solution design, I propose several novel techniques to implement the solution. First, a new data structure, called the scanning node tree is proposed and implemented. The scanning node tree organizes multiple cluster nodes effectively to fulfill parallel processing of the hand tracker tasks. Using the scanning node tree instead of sequential data structures to hold node information reduces the communication cost, incorporates task partitioning information in the tree topology itself, and is inherently suitable for node addition, node deletion or processing capability changes in the cluster. Second, a load balancing algorithm, based on optimized grouping and partition of scan areas and scales, is proposed and implemented. The load balancing algorithm requires minimal input parameters from the system administrator, and is capable of planning the work load partition automatically. The introduction of this algorithm, as shown in the evaluation results, reduces the system administrator's burden to a large extent, and is able to achieve optimized usage result of the computation and communication resources of the cluster. Third, I design and implement the corresponding applications to be run on head and worker nodes of the cluster. A in-house message-passing mechanism among the cluster nodes is also designed, implemented and proved to be effective.

Instead of making the communication cost a quantified parameter into the task partitioning and load balancing algorithms, its impact is empirically considered. The reason for this is that the scalable solution does not optimize the communication, while there is a large quantity of off-the-shelf methods for communication optimizations. For example, at the application layer, image data can be usually compressed by a factor of 5-10. At the link layer such as the network interface card drivers, data can also be compressed, or sent and received collectively to improve transmission performance. Hardware improvement trends are also under the consideration, such as the 10Gbps network cards which are already in market albeit still expensive. All these factors can potentially improve the system to perform better than the current proof-of-concept implementation.

Although the scalable computing solution presented in this chapter is directly devised for hand tracker task parallelization, in a broader scope, this solution can be applied to a set of “scan-and-match” problems, where the task is about identifying pattern presence/type in continuous data streams and testing the incoming data is in the form of intensive scans at different scales. Such kinds of problems are not rare in the mobile and pervasive computing environment, such as pattern recognition in ad-hoc sensor networks.

7. HAND WAND: AN EVALUATION APPLICATION

In this chapter, I evaluate the hand tracker from the application-centric approach. I enhance the hand tracker with networking capabilities to integrate it with a high-performance graphics rendering middleware. The enhanced hand tracker, named HandWand, serves as a human-computer interaction device. This chapter serves as a reference design to integrate the hand tracker with a real-life graphical application.

7.1. Problem Statement

Developed at EVL-UIC, the LambdaVision [76] is an ultra-high-resolution visualization and networking instrument designed to support collaboration among co-located and remote experts requiring interactive ultra-high-resolution imagery. LambdaVision investigates means to advance both science and public safety as validated by users in various disciplines of earth science research, and training exercises in disaster response and crisis management. In one application, LambdaVision serves up to 60 trillion bytes (TB) of U.S. urban city map data to distributed 100-megapixel displays, enabling scientists and local, state and Federal agencies to compare real-time imagery of disasters (fire, earthquakes, hurricanes, etc.) with stored city, regional and national maps in high detail [77].

What's driving the visualization rendering on the LambdaVision is a graphics middleware called Scalable Adaptive Graphics Environment (SAGE) [75]. The network-centered architecture of SAGE allows collaborators to simultaneously run various applications (such as 3D rendering, remote desktop, video streams and 2D maps) on local or remote clusters, and share them by streaming the pixels of each application over ultra-high-speed networks to large tiled displays.

There are three reasons to make it attractive to make a vision-based hand posture interface for the LambdaVision: the first reason is that SAGE requires one or more computer cluster, interconnected by high speed networks as its infrastructure. This requirement coincides with the computing facility requirement of the hand tracker. The second reason is that LambdaVision has much larger screen size than traditional 2D displays. In turn, interaction between the users and the LambdaVision could not be satisfyingly fulfilled by traditional interaction techniques, such as a 2D mouse. Vision-based techniques stand themselves out under this circumstance on tetherlessness and larger space of freedom movement of the users. Last but not least, the performance of SAGE is subject to the computation and communication resources availability of the computer cluster, running the hand tracker together with SAGE, more hands-on data of the compact of the hand tracker on the application it serves for can be measured.

Based on these three motivations, I enhanced the hand tracker by adding a networking interface to it and integrating it with SAGE. The enhanced hand tracker is named HandWand. The name comes from a hand held 3D tracker and controller in the original CAVE system [44].

7.2. Design and Implementation

Based on personal communications[10], the mechanism SAGE maps interaction device data to screen events is as following:

[10] with developers of SAGE UI at EVL-UIC.

The devices send data messages to the SAGE device manager through a socket interface. The message starts with the device's name, followed by its type, and then a private data part which is specific to the device.

When a message from the device is received by the device manager, the manager checks if SAGE has a plug-in defined for the specific device, based on its name and type. If SAGE has the corresponding plug-in defined, then the device manager forward the private data part of the message to that specific plug-in to process.

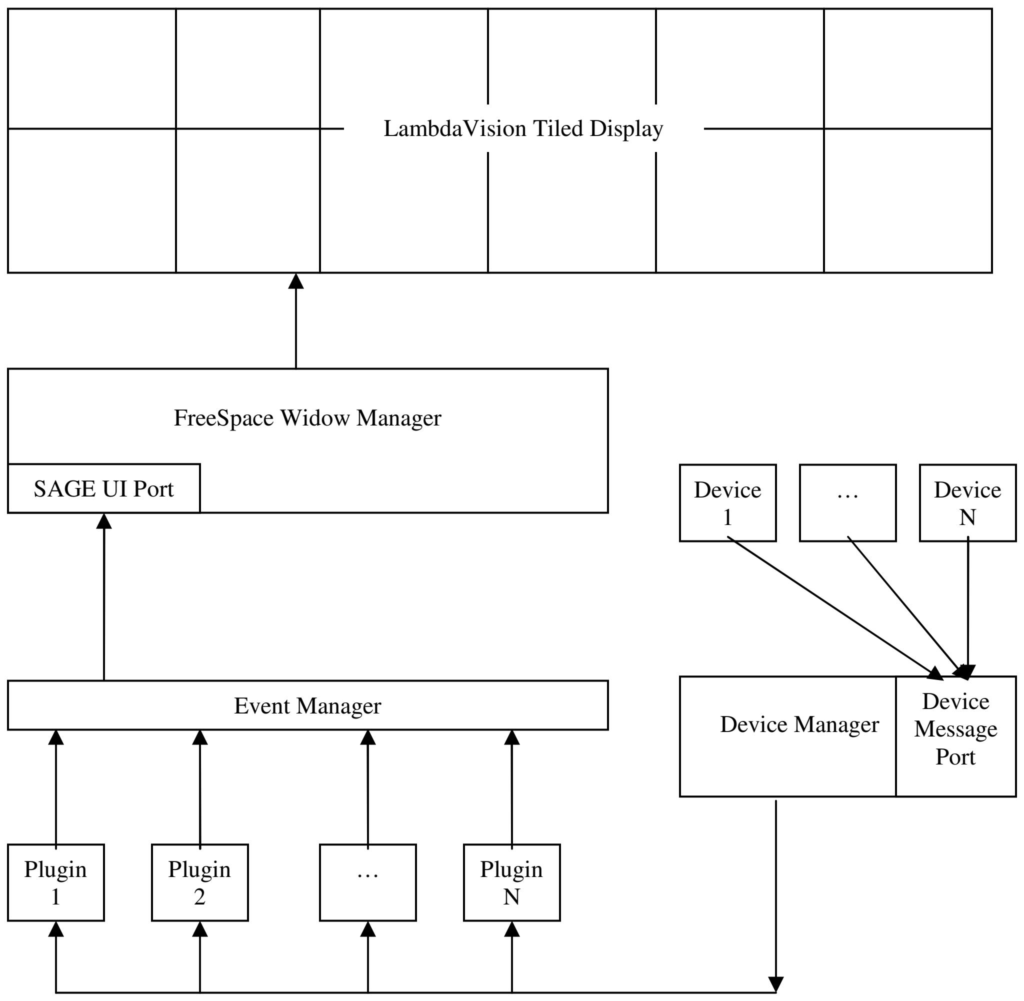


Figure 53 Mechanism used by SAGE to Map Device Data to Screen Events

The device plug-in parses the private data into tracking and controlling information and post UI events through the SAGE event manager.

SAGE event manager communicates with the freespace window manager through a socket interface called SAGE UI port. Freespace window manager then manipulates screen pixels.

From the process described above, I need to design two parts of the HandWand to integrate it with SAGE, in other words, the networking interface to communicate with the device manager and the SAGE plug-in to parse HandWand data.

7.2.1. Networking Interface

A TCP-based networking module is designed and implemented for the HandWand. The networking module implements a connection manager which listens at port 5853 for incoming connections. The connection manager managers a non-blocking TCP server and a vector data structure for active clients. When HandWand initiates, it starts the TCP server. While HandWand is running, it checks for incoming connections at the beginning of each frame processing, in a non-blocking fashion. If there is an incoming connection request, then the connection manager accepts this request, and put the client into the active clients vector. At the end of each frame processing, the connection manager collects 2D coordinate data and posture data and put them into a string in the following format:

"X, Y, Button"

In this message string X is a float value from 0 to 1, as the percentage of X position with regard to frame image width, and Y is a float value from 0 to 1, as the percentage of Y position with regard to frame image height. Button is a string that equals to the posture name if it is detected, and empty if there is no posture detected. If the hand is not tracked in the camera's field of view, then both X and Y are set to be -1.0. When HandWand exits, the connection manager disconnects all client connections.

7.2.2. HandWand Device Plug-in

The HandWand device plug-in is a post-processing call-back routine run by SAGE when a message from HandWand is received. The most simple plug-in would be a routine that just breaks the private data into three values: X, Y and Button. In practice, the plug-in does more work than simple string token retrievals. There are three tasks that need to be fulfilled by the plug-in: coordinate smoothing, OpenCV-SAGE coordinates conversion and button press debouncing.

The hand is highly deformable and often has drastic shape changes during two consecutive frames. Although the KLT feature tracker is very robust in tracking the blob position of the hand, the exact center position of the hand is difficult to be accurately reported (I assume the center of the hand is the center of mass). Because of this, the 2D position output of the tracker has frequent local jitters. To remove these jitters, I use a moving average window to smooth the coordinate data. The moving average window size is 5, if the hand lost track within the 5 most recent frames, then only valid position (i.e., position readings that are not -1.0) are counted and averaged.

The second task of the handwand device plug-in is to convert the tracker-reported device coordinates into SAGE screen coordinates. In the device coordinates system, the top left corner of the frame image is mapped to (0, 0) and the bottom right of the frame image is mapped to (1.0, 1.0). While in the SAGE coordinate system, the bottom left corner of the tiled display screen is mapped to (0, 0) and the top right corner of the tiled display screen is mapped to (SAGE_WIDTH, SAGE_HEIGHT). The conversion calculation between SAGE and HandWand coordinates can be described by Equation 14.

$$\begin{bmatrix} SageX \\ SageY \end{bmatrix} = \begin{bmatrix} SAGE\_WIDTH & 0 \\ 0 & SAGE\_HEIGHT \end{bmatrix} \begin{bmatrix} HandWandX \\ 1 - HandWandY \end{bmatrix}$$

Equation 14 Conversion Calculation between SAGE and HandWand Coordinates

Lastly, due to hand movement and tracker detection rate, when the user keeps performing specific posture during a span of frames, it is common that the recognition component does not report a series of consecutive posture presences. Instead, the recognition component usually reports several blocks of posture presences, with intermittent absence of the posture. When the posture presence is translated to button states, there will be multiple button press and releases during a semantic “button press”. This is similar to the bouncing effect of an electromechanical push button.

To address the bouncing problem, a debouncing algorithm is used to provide steady posture presence/absence state. I keep a posture state window of the most recent posture presence/absence states and use a threshold to calculate the smoothed posture presence state. This approach is proved to be very effective when the posture state window size is 20 and the threshold is 0.8.

7.3. <u>Running the Integrated System</u>

Figure 54 shows a scene where the HandWand is used as a human-computer interaction device of the Lambda vision. The user can easily move the hand to control a 2D cursor on the screen, and perform postures to be mapped to button presses. In the configuration illustrated by Figure 54, the PointGrey camera is mounted on a tripod. By positioning the tripod at difference places and depths in front of the LambdaVision tiled display, the size of interaction space can be optimized for various application needs.

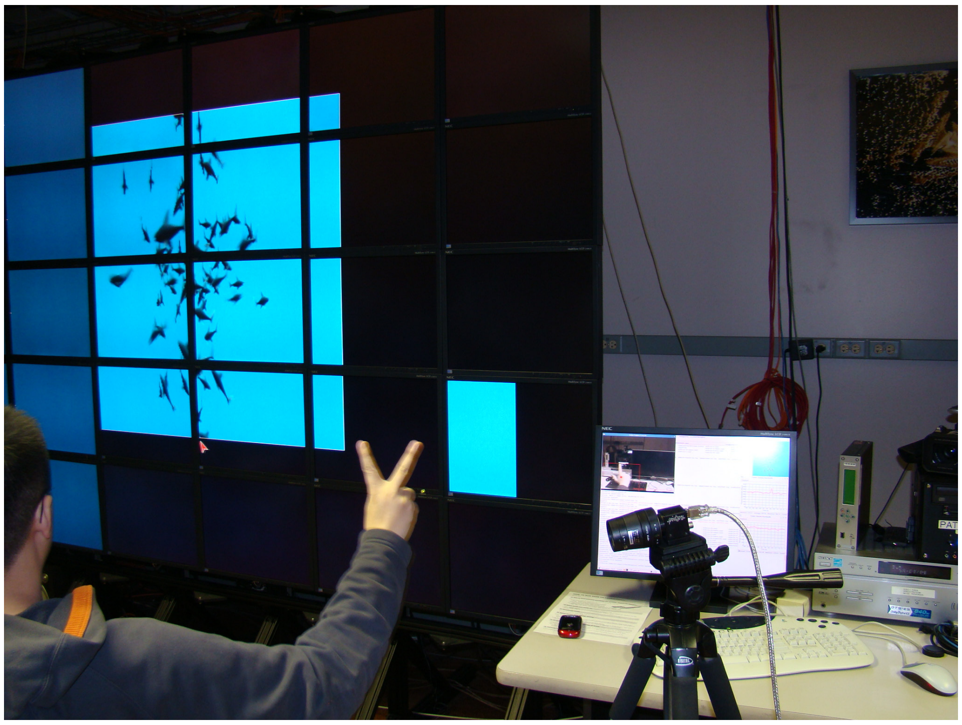


Figure 54 Running HandWand as a Human-Computer Interaction Device for the LambdaVision

Since SAGE also uses pixel streaming across nodes to achieve large scaled area displaying, it is of interest to observe whether the deployment of HandWand will hinder SAGE performance. Figure 55 shows the means and standard deviations of display bandwidth as well as frame rate recorded by SAGE, when running the "Atlantis" benchmark application. The "Atlantis" application uses the cluster head node as its sole rendering node, and SAGE streams the pixels to a 4 by 5 tile, each at the resolution of 1600 by 1200 pixels.

The two conditions are running SAGE with and without running HandWand. When the HandWand is running, it is using a 12-node scanning tree for the hand recognition task.

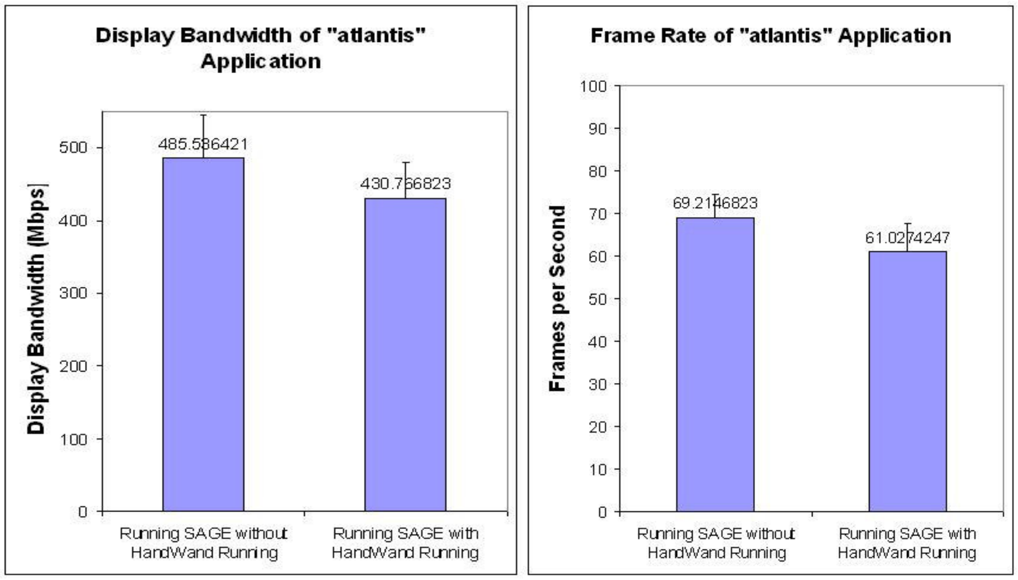


Figure 55 SAGE Display Bandwidth (left) and Frame Rate (right) with and without HandWand Running

It can be seen from Figure 55 that when SAGE is running without HandWand running, the Atlantis application has the display bandwidth of 455.59±59.74 Mbps and the frame rate of 69.21±5.26 fps. When SAGE is running with HandWand running, the atlantis application has the display bandwidth of 430.77±49.84 Mbps and the frame rate of 61.03±6.44 fps. Running HandWand does cause performance degradation of the SAGE application but it is only about 11.3% for the display bandwidth and 11.8% for the frame rate for this specific experiment. This result is within the acceptable range.

7.4. <u>Conclusions and Discussions</u>

In this chapter the design and implementation of a real life application of the hand tracker – HandWand – is presented. The HandWand shares the same computing infrastructure with a high-performance graphics middleware, and is proved to be effective as a human-computer interaction device for large scale tiled displays.

I illustrate that to make the hand tracker core functionality usable for applications as an interaction modality, only proper data sharing wrappers as well as device data post-processing need to be added on. Although HandWand uses the socket interface for communications, there are no obstacles to restrict it from using other common inter-process mechanisms used in VEs, such as shared memory.

In the HandWand application, the computer vision based posture interface exhibits two major advantages compared with traditional interaction techniques. The first is the effective use of the high-performance computing facility, while many large scale visualization middleware and applications require a computer cluster to run, they usually do not use up all the computation and communication resources available. By introducing the HandWand optimizes the usage such resources. The second advantage is more flexible interaction space. Computer vision based interaction can be performed in a wide space near the visualization instrument, as long as this space is in the camera's field of view. With the increasing needs for large scale scientific, public safety and educational visualization instruments and techniques, I expect the use of applications like HandWand will gain more popularity in the near future.

8. CONCLUSIONS AND FUTURE WORK

This thesis systematically presents the background, needs and technical challenges to create a computing environment that fits the population of mobile devices users, is capable of being personalized for its user, and utilizes the processing power of the infrastructure to a maximized extent. The name of *Personal Augmented Computing Environment*, or PACE, reflects my research endeavors towards such an environment framework that integrates the human factors, the mobile/portable device, and the infrastructure computing resources more tightly into an ecosystem.

As stated in the introduction and background chapters, the proliferation of personal mobile devices and lack of an available framework to effectively create synergy between the mobile devices and infrastructures are the main motivation of this work. The three goals: size constancy study, personalized hand tracker and cluster computing for hand tracking reflect the research efforts to make a proof-of-concept framework that might benefit next-generation consumer products, as well as being useful for the present-day scientific and commercial applications. The quantitative and application-centric evaluations conducted as part of this work prove that the proposed techniques are feasible, user-friendly and effective.

Although this work sets the research problems to be specifically on visualization and interaction aspects, the motivations, design philosophies and technology solutions presented in this work can be surely applied to a wider context. The research problems can be generalized into the following four questions:

Which personal profiles of an individual user should a computing environment maintain?

In what ways can user-specific characteristics be used to facilitate user-centric computing?

How to effectively partition computational tasks between personal mobile devices and infrastructure systems?

How to maximize the benefits of the infrastructures' processing power for personal tasks using scalable techniques?

Each chapter of this thesis answers one or more of the above questions to a certain extent and contributes a tile to the mosaics of research activities in the related fields. I believe that in the next one or more decades, personal mobile devices will be equipped with much richer multimedia features, be pervasive like never before, and start to play the central role of personal computing, just like the personal computers nowadays taking dominance position over their mainframe predecessors a couple of decades ago.

The contributions of this work can be summarized as follows:

Firstly, it studies the effects of three major visual factors in a VE on the size constancy performance of the VE users. Among the three visual factors, namely scene complexity, stereoscopy and motion parallax, the former two seem to have significance in determine users' size constancy performance, while the latter seems not. This work presents the set of experiment designed and conducted for studying these three visual factors, as well as discussed in detail both the group and the individual performance results.

Secondly, it proposes and implements an effective personal profile capture and construction process that is able to be carried out fully on a mobile device. This process allows easy-to-use and robust hand image samples collection by the mobile device users, maintains the user specific characteristics – the hand shape and the skin color – to construct personal hand profile. Based on the personal profiles, a hand tracker implemented by this work is able to integrate both shape and color cues for hand detection, tracking and gesture recognition and has obtained satisfying accuracy and frame rate performances.

The third contribution this work makes is identification of parallelizable computation components in the hand tracker tasks and scalable computing techniques to speed up the processing of these components. Scalable computing techniques are applied on detection and recognition tasks and achieved satisfying results. I propose a novel data structure – the scanning node tree – which can effectively organize computation nodes in both centralized and ad-hoc systems. I also propose a load balancing algorithm that automates the workload partitioning across the nodes in the scanning node tree. Evaluation results show that the proposed techniques can improve system throughput and reduce overall frame latency at a reasonable cost of infrastructure networking and processing resources.

Last but not least, a real-life application: the Hand Wand is designed and integrated with a large-scale visualization instrumentation to prove that PACE not only can be a reference framework design of future system, but can also contribute to present-day scientific visualization systems as a useful interaction modality.

This work also showcases quite a few research opportunities for following up. Size constancy performance, as discussed in chapter 4, can be further conducted in VEs different from CAVE-like systems, such as mobile device screens. Hand tracker as discussed in chapter 5, can use more advanced features for model training and hand identification/recognition. It will also be interesting to integrate other user-specific characteristics other than the skin color to construct personal hand profiles. The cluster computing techniques discussed in chapter 6 can be strengthened at least in two ways. One opportunity is using hybrid parallelization schemes, which make use of both the vector processors and message passing libraries. The other opportunity is to apply scalable computing techniques to the cascade training process, which is left in this work.

9. CITED LITERATURE

1. The Bluetooth Special Interest Group, http://www.bluetooth.com.

2. The 3rd Generation Partnership Project, http://www.3gpp.org.

3. The WiMAX Forum, http://www.wimaxforum.org.

4. Stytz, M.R., Frieder, G., and Frieder, O., “Three-dimensional Medical Imaging: Algorithms and Computer Systems”. ACM Computing Surveys, 23(4), 421-496.

5. Defanti, T.A. and Brown, M.D., “Visualization in Scientific Computing”, Advances in Computers, 33, 247-305.

6. Scott, W.B., “Computer Simulations Place Models of Humans in Realistic Scenarios”. Aviation Week & Space Technology, 134(25), 64-65.

7. Dix, A, Finlay, J. Abowd, G., and Beale, R., “Human Computer Interaction”, New York: Prentice Hall, 1993.

8. Kamper D.G., Harvey R.L., Suresh S. and Rymer W.Z., “Relative Contributions of Neural Mechanisms versus Muscle Mechanics in Promoting Finger Extension Deficits Following Stoke”, Muscle & Nerve, pp. 309-318, Sept. 2003.

9. Kamper D.G. and Rymer W.Z., “Impairment of Voluntary Control of Finger Motion Following Stroke: Role of Inappropriate Muscle Coactivation”, Muscle & Nerve, 24 pp. 673-681.

10. Gowland C., Stratford P., Ward M., Moreland J., Torresin W., et al, “Measuring physical impairment and disability with the Chedoke-McMaster Stroke Assessment,” Stroke, vol. 24, pp. 58-63, 1993.

11. Jacobson-Sollerman C, Sperling L. “Grip function of the healthy hand in a standardized hand function test. A study of the Rancho Los Amigos test”, Scand J Rehabil Med. 1977;9(3):123-9..

12. Parker V.M., Wade D.T., and Hewer R.L., “Loss of arm function after stroke: measurement, frequency, and recovery,” Int. Rehabil. Med., vol. 8, pp. 69-73, 1986

13. Bhakta B.B., Cozens J.A., Chamberlain M.A., Bamford J.M., “Impact of botulinum toxin type A on disability and carer burden due to arm spasticity after stroke: a randomized double blind placebo controlled trial”, J Neurol Neurosurg Psychiatry 2000:69, pp. 217-221.

14. Wade D.T., “The epidemiologically based needs assessment reviews. Volume 1,” in. Health Care Needs Assessment, A. Stevens and J. Raftery, editors, pp. 111-255, 1994.

15. Kahn L.E., Averbuch M., Rymer W.Z., Reinkensmeyer D.J., “Comparison of Robot-assisted Reaching to Free Reaching in Promoting Recovery from Chronic Stroke”, Int. of Assisted Tech. in the Information Age, Amsterdam, IOS Press, pp. 39-44, 2001.

16. Trombly C.A., Occupational Therapy for Physical Dysfunction, Baltimore: Williams and Wilkins, pp. 454-471, 1989.

17. Burgar C.G., Lum P.S., Shor P.C., and Van der Loos M., “Development of robots for rehabilitation therapy: the Palo alto VA/Stanford experience,” J. Rehab. Res. Develop, vol. 37: pp. 663-673, 2000.

18. Desrosiers J, Bravo G, Hebert R, Dutil E, Mercier L. “Validation of the Box and Block Test as a measure of dexterity of elderly people: reliability, validity, and norms studies”, Arch Phys Med Rehabil. 1994 Jul;75(7):751-5.

19. Taub E. and Crago J.E., "Constraint induced movement therapy: a new approach to treatment in physical rehabilitation," Rehab. Psychol., vol. 43, pp. 152-170, 1998.

20. Kamper D.G., Hornby T.G. and Rymer W.Z., "Extrinsic Flexor Muscles Generate Concurrent Flexion of All Three Finger Joins". J. Biomechanics, 35 (2002) 1581–1589.

21. Merians A.S., Jack D., Boian R., Tremaine M., Burdea G.C., Adamovich S., Recce M., and Poizner H., "Virtual Reality-Augmented Rehabilitation for Patients Following Stroke", Physical Therapy, Vol. 82(9), pp. 898-915, September 2002

22. http://www.immersion.com/3d/products/cyber_grasp.php.

23. Kamper D.G., Cruz E.G. and Siegel, M.P. "Stereotypical Fingertip Trajectories during Grasp". J Neurophysiol, Vol 90, pp. 3702-3710, 2003.

24. Nokia Computer Vision Library, http://research.nokia.com/research/projects/nokiacv.

25. The Second Life Virtual Universe, http://www.secondlife.com.

26. The There Virtual World, http://www.there.com.

27. Lok, B., Naik, S., et al.. "Incorporating Dynamic Real Objects into Immersive Virtual Environments", Proceedings of the 2003 Symposium on Interactive 3D Graphics, Monterey, California, ACM Press: 31-40, 2003.

28. Quek, F., "Unencumbered Gestural Interaction." Multimedia, IEEE 3(4): 36-47, 1996.

29. Xiong, Y., Quek, F., et al. "Hand Motion Gestural Oscillations and Multimodal Discourse", in Proceedings of the 5th International Conference on Multimodal Interfaces, Vancouver, British Columbia. Canada, ACM Press: 132-139, 2003.

30. Starner, T., Auxier, J., et al. "The Gesture Pendant: a Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical

monitoring". In Proceedings of the Fourth International Symposium on Wearable Computers, 2000.

31. Starner, T., Weaver J., et al. "Real-time American Sign Language Recognition using Desk and Wearable Computer Based Video". IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12): 1371-1375.

32. Kurata, T., Okuma T., et al. "The Hand Mouse: GMM Hand-color Classification and Mean Shift Tracking. Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems". In Proceedings of the 2001 IEEE ICCV Workshop.

33. Kolsch, M., Turk M., et al. "Vision-based Interfaces for Mobility", in Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS), 2004.

34. Girado J., Sandin D., DeFanti T.A., and Wolf L., "Real-time Camera-based Face Detection using a Modified LAMSTAR Neural Network System". in Proceedings of IS&T/SPIE's 15th Annual Symposium Electronic Imaging 2003, Applications of Artificial Neural Networks in Image Processing VIII, Santa Clara, California, USA, 2003.

35. Luo X., Kenyon R.V., Kline T., Waldinger H. and Kamper D.G., "An Augmented Reality Environment for Post-Stroke Finger Extension Rehabilitation", In Proceedings of IEEE 9th International Conference on Rehabilitation Robotics (ICORR'05), Chicago, IL. June 2005.

36. Luo X., Kline T., Waldinger H., Kenyon RV. and Kamper DG., "Integration of Augmented Reality and Assistive Devices for Post-Stroke Hand Opening Rehabilitation", In Proceedings of the 27th International Conference of the IEEE

Engineering in Medicine and Biology Society (EMBC'05), Shanghai, China, September. 2005.

37. Huber J., Stringer N., Davies I., and Field D., "Only Stereo Information Improves Performance in Surgical Tasks", SPIE 5372: 463-470, 2004

38. Beall A., Loomis J., Philbeck J., and Fikes T., "Absolute Motion Parallax Weakly Determines Visual Scale in Real and Virtual Environments", SPIE 2411:288-297, 1995

39. Rondot P., Lessard J., and Robert J., "Study of Motion Parallax in Depth Perception with a Helmet-Mounted Display System used in Teleoperation", SPIE 2590:151-159, 1995

40. Ikehara C., Cole R., and Merritt J., "Effects of Test Structure on Depth Perception Measurement Tasks", SPIE 1669:135-141, 1992

41. Watt S., et al, "Can Observers Exploit Enhanced Motion Parallax to Control Reaching Movements within Telepresence Environments?", SPIE 4299:429-438, 2001

42. Rosen P., Pizlo Z., Hoffmann C., and Popescu V., "Perception of 3D Spatial Relations for 3D Displays", SPIE 5291:9-16, 2004

43. Hirose M., Hirota K., and Kijima R., "Human Behavior in Virtual Environments", SPIE 1666: 548-553, 1992.

44. Cruz C., Sandin D., Defanti T., Kenyon R., and Hart J., "The CAVE Audio-Visual Environment". ACM Trans. on Graphics, 35: 65-72, 1992.

45. Holaday B.,"Die Grössenkonstanz der Sehdinge bei Variation der innerenund äusseren Wahrnehmungsbedingungen". Arch. ges. Psychol., 88: 419-486, 1933.

46. Harvey L., Leibowitz H., “Effects of Exposure Duration, Cue Reduction, and Temporary Monocularity on Size Matching at Short Distances”, J. Opt. Soc. Am., 57: 2, pp. 249-253, 1967.

47. Leibowitz H., Dato R., “Visual Size-Constancy as a Function of Distance for Temporarily and Permanently Monocular Observers”. Am. J. Psychol. 79: 279, 1966.

48. Eggleston R., Janson W. and Aldrich K., “Virtual Reality System Effects on Size-Distance Judgments in a Virtual Environment”. Proc. VRAIS, pp 139-146, 1996

49. Baitch L., Smith R.C., “Physiological Correlates of Spatial Perceptual Discordance in a Virtual Environment”, in Proc. Fourth International Projection Technology Workshop, Ames, Iowa, 2000.

50. Holway A., Boring E., “Determinants of Apparent Visual Size with Distance Variant”. American Journal of Psychology, 54, pp. 121-151, 1941.

51. Kenyon R.V., Sandin D., Smith R., Pawlicki R. and Defanti T., “Size-Constancy in the CAVE”, Presence, 16(2), 2007.

52. Keshner E.A., Kenyon R.V., Langston J., “Postural Responses Exhibit Multisensory Dependencies with Discordant Visual and Support Surface Motion”. J Vest Res. 14:307-319, 2004.

53. Viola P., Jones M., “Rapid Object Detection Using a Boosted Cascade of Simple Features”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), ISSN: 1063-6919, Vol. 1, pp. 511-518, December 2001.

54. Lienhart R, Maydt J., “An Extended Set of Haar-like Features for Rapid Object Detection”, IEEE ICIP 2002. vol. 1; 2002. p. 900-903.

55. Kuranov A., Lienhart R, and Pisarevsky V., "An Empirical Analysis of Boosting Algorithms for Rapid Objects with an Extended Set of Haar-like Features," Intel Technical Report MRL-TR-July02-01, 2002.

56. Freund Y. and Schapire R.E., "Experiments with a New Boosting Algorithm," in Machine Learning: Proceedings of the Thirteenth International Conference, Morgan Kauman, San Francisco, pp. 148-156, 1996.

57. Bradski G., "Computer Vision Face Tracking for Use in a Perceptual User Interface," Intel Technology Journal, http://developer.intel.com/technology/ itj/q21998/articles/art_2.htm, Q2 1998.

58. Shi J. and Tomasi C., "Good features to track". In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94), pages 593--600, IEEE Computer Society, Seattle, Washington, June 1994.

59. Haralick R.M. and Shapiro L.G., "Computer and Robot Vision", vol. I (1992) and vol. II (1993), Addison-Wesley.

60. Ballard D.H. and Brown C.M., Computer Vision, Prentice-Hall Inc., 1982 George H.J. and Donald G., "Colour Spaces for Computer Graphics", Computer Graphics, vol. 12, no. 3, pp. 20-25, 1978.

61. Perez F. and Koch C., "Toward Color Image Segmentation in Analog VLSI: Algorithm and Hardware", International Journal of Computer Vision, vol. 12, no. 1, pp. 17-42, 1994.

62. Nevatia R., "A color Edge Detector and Its Use in Scene Segmentation", IEEE Trans. Syst. Man, Cyb. 7(11), pp. 820-826.

63. Adel M., Wolf D., Vogrig R. and Husson R., "Evaluation of Colour Space in Computer Vision Application of Wood Defects Detection", Proceedings, IEEE International Conference on Systems, Man, and Cybernetics, pp. 499-504, 1993.

64. Lucy L.B. "an Iterative Technique for the Rectification of Observed Distributions", The Astronomical Journal 79.6 (1974): 745-753.

65. Reiter C. "With J Fast Fourier Transforms and Removing Motion Blur", ACM SIGAPL APL Quote Quad 31.1 (2000): 16-17.

66. Lowe D.G., "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

67. Lowe D.G., "Object recognition from local scale-invariant features", International Conference on Computer Vision, Corfu, Greece (September 1999), pp. 1150-1157.

68. Lowe D.G., "Local feature view clustering for 3D object recognition", IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii (December 2001), pp. 682-688.

69. Nowozin S., libsift, http://user.cs.tu-berlin.de/~nowozin/libsift/.

70. Wang C.C. and Wang C.C., "Hand Posture Recognition Using Adaboost with SIFT for Human Robot Interaction", International Conference on Advanced Robotics (ICAR'07), Jeju, Korea (August 2007).

71. Sheridan T. and Ferrel W., "Remote Manipulative Control with Transmission Delay", IEEE Transactions on Human Factors in Electronics, 4:25-29, 1963.

72. DeFanti, T., Leigh, J., Yu, O., Krishnaprasad, N., Eliason, J., Alimohideen, J., He, E. “Quanta: a Toolkit for High Performance Data Delivery”. Future Generation Computer Systems 1005 (2003).

73. Bouguet, J.Y., “Pyramidal Implementation of the Lucas Kanade Feature Tracker”. Intel Technical White Paper, included in OpenCV distribution.

74. Luo, X., “TSC-I2: a Micro-Second Precision Time Measuring Tool”, http://tsc-xluo.sourceforge.net.

75. Jeong B., Renambot L., Jagodic R., Singh R., Aguilera J., Johnson A. and Leigh, J. “High-Performance Dynamic Graphics Streaming for Scalable Adaptive Graphics Environment”, IEEE/ACM International Conference for Supercomputing (SC06), Tampa, FL (December 2006).

76. Renambot L., Jeong B., Jagodic R., Johnson A., Leigh J. and Aguilera, J., “Collaborative Visualization using High-Resolution Tiled Displays”, CHI 06 Workshop on Information Visualization and Interaction Techniques for Collaboration across Multiple Displays, Montreal, Canada (April 2006).

77. http://www.evl.uic.edu/cavern/lambdavision/

78. Culler D., Karp R., Patterson D., Sahay A., Schauser K.E., Santos E., Subramonian R., and von Eicken T, “LogP: Towards a Realistic Model of Parallel Computation”, SIGPLAN Not. 28, 7 (Jul. 1993), 1-12.

79. Alexandrov A., Ionescu M., Schauser K., and Scheiman C., “LogGP: Incorporating Long Messages into the LogP Model for Parallel Computation”, Journal of Parallel and Distributed Computing, 44(1):71--79, 1997.

10. VITA

NAME: Xun Luo

EDUCATIONS: B.S. Computer Science. University of Electronic Science and Technology. Chengdu, China. 1999.

M.S. Computer Science. University of Electronic Science and Technology. Chengdu, China. 2002.

Ph.D. Computer Science. University of Illinois at Chicago. USA. 2008.

EXPERIENCES: Software Engineer, Motorola China. (2001 – 2002)

Software Intern, Qualcomm Inc. (2006)

Senior Software Engineer, Motorola Labs. (2006 –2008)

Research Assistant (2003 – 2006), Electronic Visualization Laboratory, University of Illinois at Chicago.

PUBLICATION: Luo X., Schuler F. and Jonnalagadda K., "Token-Based Dynamic Authorization Management of RFID Systems". US Patent Pending.

Schuler F., Jonnalagadda K. and Luo X., "Method, System and Configuration for Activity Management & Data Exchange for Shared Group Activity Task & Item Tracking". US Patent Pending.

Liu H., Luo X. and Yao Y.W., "Two Manifold Learning Techniques for Sensor Localization", In Proceedings of the 2007 IEEE International Conference on Systems, Man and Cybernetics(SMC'07), Montreal, Canada, Oct. 2007.

Luo X., Piret K. and Guan Y., "A Service Framework for Temporal Link Analysis with Historical Segments Integration", In Proceedings of the 2007 IEEE International Conference on Systems, Man and Cybernetics(SMC'07), Montreal, Canada, Oct. 2007.

Luo X., "PACE: Augmenting Personal Mobile Devices with Scalable Computing", In Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07), Rio de Janeiro, Brazil, May 2007.

Luo X., Kenyon R.V., Kamper D.G., Sandin D. and DeFanti T., "The Effects of Scene Complexity, Stereovision, and Motion Parallax on Size Constancy in a Virtual Environment", In Proceedings of IEEE Virtual Reality Conference 2007(VR'07), Charlotte, NC. March 2007.

Simone L.K., Sundarrajan N., Luo X., Jia Y. and Kamper DG., "A Low Cost Instrumented Glove for Extended Monitoring and Functional Hand Assessment", Journal of NeuralScience Methods, October, 2006.

Luo X., Kenyon R.V., and Guan Y. "Pervasive Web Community Summarization: A Machine Learning Approach". In Proceedings of the 2006 International Conference on Machine Learning; Models, Technologies & Applications (MLMTA'06). Las Vegas, NV, June 2006.

Luo X., Kenyon R.V., Kline T., Waldinger H. and Kamper D.G., "An Augmented Reality Environment for Post-Stroke Finger Extension Rehabilitation", In Proceedings of IEEE 9th International Conference on Rehabilitation Robotics (ICORR'05), Chicago, IL. June 2005.

ACTIVITIES: IEEE Technical Committee on Scalable Computing Young Researcher Travel Grant, 2007.

University of Illinois at Chicago Provost Award for Graduate Research, 2006.

Google "Summer of Code 2006" Award, 2006.

Google "Summer of Code 2005" Award, 2005.

USENIX Student Travel Award, 2004.

External reviewer for IEEE VR and 3DUI.conferences

External reviewer for International Journal of Integrated Computer-Aided Engineering

11. APPENDIX

This appendix is the copy of the approval notice document from the UIC Institutional Review Board (IRB) for the size constancy study protocol used in thesis research.