HAND TRACKING, FINGER IDENTIFICATION, AND CHORDIC MANIPULATION ON A MULTI-TOUCH SURFACE

by

Wayne Westerman

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering

Spring 1999

HAND TRACKING, FINGER IDENTIFICATION, AND CHORDIC MANIPULATION ON A MULTI-TOUCH SURFACE

by

Wayne Westerman

Approved: _____

Neal Gallagher, Ph.D.
Chair of the Department of Electrical Engineering

Approved: _____

Andras Z. Szeri, Ph.D.
Interim Dean of the College of Engineering

Approved: _____

John C. Cavanaugh, Ph.D.
Vice Provost for Academic Programs and Planning

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

John Elias, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Charles Boncelet, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Phillip Christie, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Kenneth Barner, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

John Scholz, Ph.D.
Member of dissertation committee

ACKNOWLEDGMENTS

Abundant thanks go to my adviser, John Elias, whose fond support, daily teamwork, and unfathomable hardware know-how gave me a unique foundation upon which to compose a dissertation.

Dr. Neal Gallagher, for inviting me to Delaware, ensuring my research in the Electrical Engineering Department and other parts of campus was always fully supported, offering weekly spiritual advice, and challenging me with proclamations of what could and could not be done. May he continue to carry his wisdom all across the country.

Dr. Charles Boncelet, Dr. Kenneth Barner, and Dr. John Scholz, for suggesting many helpful revisions to this manuscript.

Dr. Phillip Christie, for many fascinating lectures, and for challenging me to find the principles behind my inventions.

Dr. Rakesh, for making me write and understand mathematical proofs until they all look trivial.

Chris Thomas, for a most enduring friendship.

My piano teachers Beverly Stephenson and Ruth Anne Rich, for inspiring me with what the hands can do on a properly responsive instrument.

My typing assistants, Sarah Ruth Budd, Sara Levin, Mark Parsia, Denise Lemon, Barbara Westog, my sister Ellen, and my mother Bessie, for helping me to continue to program and publish for the past four years as I perfected less fatiguing methods for data entry.

Samuel Audet, for generously writing HotScroll, OS/2's only continuous scrolling software.

Brian Hall at Microedge, Inc., for adding a keystroke-saving variable-name-completion feature to Visual Slickedit at my bequest.

My fellow residents of Lovett Graduate House, for sharing the television when my mind was too weary for anything else.

I am so lucky to have such a loving, patient, and stable family, who always welcome me home twice a year even though I moved so far away. I thank my father for drawing me back to the farm for refreshing manual labor yet giving me time to develop one crazy project after another on my vacations, and also for enforcing the pragmatism and ethics of the frontier. I thank my mother for spicing my vacation diet with wholesome home-grown foods and swimming. I thank my grandmother Edna and her family for bringing poetry, history, and gentleness to my summers; may she always whisper from above how her family almost got transplanted to California. I thank my grandfather Walt for introducing electricity to our home town with only a fifth grade education; that was only the beginning.

This work was partially funded by a National Science Foundation Graduate Fellowship for Wayne Westerman.

This manuscript is dedicated to:

My mother, Bessie,

who taught herself to fight chronic pain in numerous and clever ways,

and taught me to do the same.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

assignment problem Finding the best one-to-one matching between sets of equal size.

attractor ring A set of points fixed in a ring around the hand, one per finger and palm heel, which are used to associate nearby unidentified contacts with particular fingers.

bimanual manipulation Using two hands simultaneously to navigate, move, or stretch onscreen objects.

carpal tunnel syndrome Compression of the median nerve caused by inflammation of the tendons which pass through the carpal tunnel at the underside of the wrist. Causes numbness, tingling, sharp wrist pains at night, and eventual degradation of hand motor control if untreated. Though one of the most widely feared repetitive strain injuries, it appears in only about 20 percent of RSI cases [117].

chord A combination of fingers on one hand which contact a surface simultaneously. Some combinations are easier for the user to perform or the system to recognize than others.

chordic manipulation 4-DOF control of onscreen graphical objects with slides of two or more fingers across a surface.

channel selection Choosing and touching a particular combination of fingers to select between pointing, dragging, scrolling, *etc.*, in analogy to pressing a subset of mouse buttons.

contact (noun) A general term for signals produced when a grounded conductive object such as a finger approaches a capacitance-sensing surface. The groups, paths, and fingers of Chapters 3 and 4 are each contacts at different stages of processing.

cumulative trauma disorders (CTD) Slightly more general term than repetitive strain injury which includes occupational back injuries.

cubital tunnel syndrome Same as carpal tunnel syndrome except compresses ulnar nerve at the elbow (near the funny bone).

degrees of freedom (DOF) The number of independent directions a solid object or joint can move. Three-dimensional free space has six-DOF, three for translation along the x, y, and z axes, and three for rotation in the xy, yz, and xz planes. The joints of the wrist and fingers have over 20-DOF total.

DeQuervain's Syndrome Entrapment of the tendons which extend and raise the thumb where they pass through tendon sheaths at the wrist. Can be caused by holding thumbs too high when typing or pulling thumb backwards on thumb-operated trackballs.

Dvorak key layout An alternative key layout designed by August Dvorak in the 1930's. Its primary advantage is that the most frequently typed characters are placed on home row, so finger excursions to the front and back rows are greatly reduced.

electrode A thin conductive plate, thousands of which form the sensor array of the MTS. The sensed parameter is the change in electrode capacitance caused by approach of another conductive object such as fingertip flesh. Precise contact locations are obtained by grouping and interpolating neighboring electrode measurements.

finger Any of the thumb, index, middle, ring or pinky.

finger identification Determining which fingertip, palm heel or thumb on a given hand is causing a particular surface contact.

fingertip The tip of any of the index, middle, ring or pinky fingers, but not the tip of the thumb.

frame rate The frequency with which the electrode scanning hardware scans the whole proximity sensing array. Also known as the array scanning rate.

hand identification Determining which of the left or right hands is causing a particular surface contact or cluster of contacts.

floating finger A finger which is in detection range (less than 3 mm from the surface) but is not actually touching the surface.

forearm pronation Rotation of the forearm so that the palm faces down.

forearm supination Rotation of the forearm so that the palm faces up.

forepalms The mounds of often callused flesh protecting the underside of the joint between the metacarpals and proximal phalanges, where the distal palm branches into the fingers.

graphical manipulation Direct control of continuous software parameters whose state changes are usually indicated by movement of something on the screen.

graphical user interface (GUI) A modern software interface like the Windows 95 desktop which has icons, menus, windows, buttons, and dialogue boxes operated principally by the mouse, as opposed to older command line interfaces which only required a keyboard. GUIs rely heavily on the mouse pointer location to determine context and mode.

group In the context of Chapter 3, a set of electrodes which all appear to be affected by the same distinguishable part of the hand.

hunt and peck typing Novice typists typically strike keys with the index fingers only, visually searching the keyboard for each key. Trained typists use this method sporadically for unfamiliar or hard-to-reach key sequences.

integral Control devices or tasks in which it is possible to move along all axes or in all degrees of freedom simultaneously, *e.g.* moving diagonally in a plane.

inner Towards the thumb of a given hand, known more formally as medial.

mouse cursor Usually denoted by an arrow pointer moving across the screen, this cursor has traditionally been controlled by a mouse.

multi-touch surface (MTS) A surface with a proximity sensor array underneath capable of unambiguously measuring the positions of multiple finger contacts.

one-shot A command or key sequence which cannot easily be undone or reversed and which is normally not repeated. One-shot commands are therefore only issued once per hand slide across the surface.

outer Towards the pinky of the given hand.

path In context of Chapters 3–5, the trajectory of a surface contact which is persistently tracked across successive proximity images. If it cannot be associated with an electrode group from the most recent array scan, it is deactivated, representing finger liftoff.

palm heels The pair of fleshy mounds at the base of the palm near the wrist.

pen grip A hand posture or configuration in which the middle, ring, and pinky fingers are curled under the palms and the thumb and index finger are pinched together as if holding a pen.

puck A mouse-like device often used with electromagnetic drawing tablets [149]. The primary differences from the conventional mouse are that the puck can report its absolute position rather than just relative changes in position, and pucks often have 4–16 buttons rather than just 1–3. Pucks and drawing tablets are used most often by professional draftsmen for computer-aided-design (CAD). These same tablets usually support styli as well.

QWERTY key layout The alphabet key layout which has long been standard on most English typewriters and computer keyboards. Character placement seems random, but speed is fairly good because typing of consecutive characters often alternates between hands.

repetitive strain injury (RSI) Long-term damage to tendons, muscles, and nerves caused by highly repetitive and forceful body motions. Tends to affect smaller muscle groups such as those in the arm and hand.

separable Control devices or tasks in which movement is only possible along one axis or degree of freedom at a time, *e.g.* driving in Manhattan geometry, or using orthogonal cursor mode in CAD programs.

slide Coupled lateral motion of all fingers in a chord across the surface.

sliding tap A brief chord contact with the surface including fast lateral finger motion.

stylus A special pen whose motion, pressure, and tilt can be sensed electromagnetically by drawing tablets (*e.g.* [149]). Most recent models are light and cordless, though older versions were encumbered with cords or heavy batteries. Styli and drawing tablets

are used most often by artists and graphics designers but can also be used with handwriting recognition software. Many of these tablets support pucks as well.

tap A quick press and release of the finger to the surface with minimal lateral motion.

tendonitis Inflammation of the tendons, the collagenous tissues which connect muscle to bone, due to overuse.

tenosynivitis Swelling of the sheath which surrounds a tendon where the tendon passes over bones or curves.

text cursor The cursor, usually denoted by a flashing bar or highlight block, at which typed characters are inserted. Can be moved incrementally by arrow and page keys.

touch typing Skilled typing in which all ten fingers are used, finger motions are quick and ballistic, and the typist does not look for the keys.

touchpad A credit-card-sized finger-sensing surface popular in notebook computers. Because touchpads contains long row and column electrodes rather than electrode arrays, they may detect two or three fingers but can only report a global position averaged over all finger contacts.

ulnar deviation Rotated posture of the wrist in which the pinky points outward away from the sides of the body.

ABSTRACT

This research introduces methods for tracking and identifying multiple finger and palm contacts as hands approach, touch, and slide across a proximity-sensing multi-touch surface (MTS). Though MTS proximity images exhibit special topological characteristics such as absence of background clutter, techniques such as bootstrapping from hand-position estimates are necessary to overcome the invisibility of structures linking fingertips to palms. Context-dependent segmentation of each proximity image constructs and parameterizes pixel groups corresponding to each distinguishable surface contact. Path-tracking links across successive images those groups which correspond to the same hand part, reliably detecting touchdown and liftoff of individual fingers. Combinatorial optimization algorithms use biomechanical constraints and anatomical features to associate each contact's path with a particular fingertip, thumb, or palm of either hand. Assignment of contacts to a ring of hand part attractor points using a squared-distance cost metric effectively sorts the contact identities with respect to the ring structure.

Despite the ascension of the mouse into everyday computing, more advanced devices for bimanual and high degree-of-freedom (DOF) manipulation have failed to enter the mainstream due to awkward integration with text entry devices. This work introduces a novel input integration technique which reserves synchronized motions of multiple fingers on the MTS for multi-DOF gestures and hand resting, leaving asynchronous single finger taps on the MTS to be recognized as typing on a QWERTY key layout. The operator can then switch instantaneously between typing and several 4-DOF graphical manipulation channels with a simple change in

hand configuration. This integration technique depends upon reliable detection of synchronized finger touches, extraction of independent hand translation, scaling, and rotational velocities, and the aforementioned finger and hand identifications. The MTS optimizes ergonomics by eliminating redundant pointing and homing motions, minimizing device activation force without removing support for resting hands, and distributing tasks evenly over muscles in both hands. Based upon my daily use of a prototype to prepare this document, I have found that the MTS system as a whole is nearly as reliable, much more efficient, and much less fatiguing than the typical mouse-keyboard combination.

Chapter 1

INTRODUCTION

1.1 The State of Hand-Computer Interaction in 1998

In the first paper to formally demonstrate the advantages of two-handed graphical manipulation, *e.g.* scrolling with one hand while pointing with the other, Buxton and Myers [23] lament:

> To date, very few computer systems easily lend themselves to experimentation with the types of interaction described in this paper.

Twelve years later, further research [62, 86, 171] has verified the efficiency and intuitiveness of simultaneous two-handed manipulation and high degree-of-freedom (DOF) controls. Manually demanding tasks such as web browsing and computer-aided- design (CAD) have also become ubiquitous, but the requisite input devices have yet to appear on the personal computer market. As Leganchuk *et al.* [90] point out,

> One reason for this may be the difficulty in equipping systems with inexpensive and available input devices capable of capturing bimanual input.

Enthusiastic investigation of gesture recognition via alternative input devices has also lulled since pen computing, data gloves, and ergonomic keyboards failed to blossom in the early 1990s. Hope for improvement in human-computer interaction has shifted to speech recognition, but speech is clearly inappropriate for precise manipulation of graphics. Computer manufacturers seem to have concluded that

the combination of a keyboard and a two-dimensional pointing device, *e.g.* mouse or touchpad, cannot be outdone in terms of overall practicality. This dissertation will challenge the status quo with an ergonomic, economical, manual input device which achieves the anticipated performance gains of two-handed gestural interaction, yet is practical enough to replace mice and keyboards in general computer use.

In currently popular graphical user interfaces, most computer users rely heavily upon the mouse to avoid memorizing keyboard commands or non-sensical hotkey sequences. These interfaces are very easy to learn because all possible actions are clearly displayed as buttons or other visual controls and accessible with simple mouse clicks, but this likeable reduction in cognitive load amplifies demands on the hand and breaks the train of thought in other ways. Even the simple task of web browsing may involve a cumbersome sequence of clicking on page links, moving the mouse pointer to distant scrollbar controls, clumsily manipulating the scrollbar, then moving the pointer back to newly uncovered links. The homing distance between keyboard and mouse may also discourage users from moving hands back to the keys for highly efficient keyboard methods. Once at the keyboard, however, the hands face further danger. Unless the keyboard is a truly ergonomic model, stiff keys, ulnar deviation and forearm pronation exacerbate the risk of carpal tunnel syndrome and other painful repetitive strain injuries (RSI).

In the past few years, the growth of the Internet has accelerated the penetration of computers into our daily work and lifestyles. The shear amount of time students spend browsing the web, writing papers, sending e-mail, and playing computer games turns the annoying inefficiencies and poor ergonomic habits cited above into a rash of crippling illnesses [132]. In 1997, over 100 students at Harvard University requested assistance because of RSI, compared to 1 in 1991. Over 200 cases were diagnosed at the MIT Student Health Center, up 44% from 1995. This author, too, has struggled throughout graduate school with tendonitis brought on by the

volume of computer programming for this and other projects. RSI disproportionately affects high achievers because of the intensity with which they work. Nature is imposing a strange limitation on the best minds of the Internet Generation, which says, "the harder you work on glorious new technology, the longer it will take your bodies to recover from the pain."

The inadequacies of the mouse-keyboard interface may also hinder the quality of artistic projects in which computers are the primary tool. Granting that computers enable amazing new audio and visual effects, clumsy interfaces also ensure the artist remains engrossed in the workings of the computer rather than in the artistic vision. Pianos and paintbrushes do not contain artificial intelligences to anticipate the intentions of the artist, yet they provide a subtlety and richness of control which allow nuance to flow from the artist effortlessly. Once a pianist has technically mastered a piece, he or she concentrates during performance on perfecting the musical phrasing, which the hands can modulate subconsciously. I easily experiences such oneness with the piano, but never with the computer, because I cannot sustain subconscious mastery of the interface.

My academic advisor, Prof. John Elias, and I began the present work when we realized that the conventional mechanical keyboard, for all of its strengths, is physically incompatible with the rich graphical manipulation demands of modern software. Single or dual-finger devices such as pointing sticks and touchpads embedded in the keyboard overuse one finger and cannot match the versatility of whole hand manipulation. Most operators will not adopt a drawing tablet with stylus or puck [90] or other bimanual manipulation methods as long as frequent movements back to the keyboard are necessary. Speech recognition reduces dependence on the keyboard in some situations, but total reliance on speech for text and command input can strain the voice and annoy co-workers. Progress appears to be stymied by a Catch 22 in which typing cannot be eliminated, yet gesture capable devices

cannot thrive in the same physical space as the keyboard.

This dissertation attempts a compromise by developing touch typing and whole hand manipulations for a keyless, multi-touch-sensitive, smooth surface. The compromise hinges upon the hypothesis that since typing movements are essentially ballistic and do not carry the subtlety of musical keyboarding, the lack of tactile reference from mechanical keys can be compensated by other means. Though perfection of these means will be left to future work, they can include depressions of the surface around home row, forming a raised dot at the center of each key, tracking hand drift over the key layout using the redundancy of English, and issuing sounds to indicate when a surface tap has been recognized as a keypress. Giving up mechanical keys provides clear ergonomic and economic benefits, as well as allowing detection of all fingers as they slide smoothly across the surface.

By replacing the keyboard with a multi-touch-sensitive surface (MTS) and recognizing hand motions as described in this dissertation, hand-computer interaction can be dramatically transfigured. Scrolling and panning need no longer interrupt the primary task, but can be accomplished with a slide of the fingers on the non-dominant hand akin to flipping the corner of a page. Browser back and forward no longer require a trip to the buttonbar, but become a speedier version of the scrolling gesture. Cut, copy, and paste become quick pinch gestures. Object sizing and rotation in drawing programs no longer requires menu access, but becomes integral with dragging by a simple contraction of the fingers or rotation of the wrist. Handwriting mode can be indicated by forming a pen grip with or without stylus, so the operator does not have to constantly pick up and put down the stylus to type. Because nearly all activity can be distinguished by relative position or velocity, a skilled operator seldom need look at the surface.

Though operators may need a couple days to get used to the different "feel" of interacting with a smooth surface, just as drivers must get used to the different

responsiveness of the controls on a new car, basic typing and pointing skills transfer from conventional keyboards and touchpads to the surface. Novices can hunt and peck on the warped QWERTY key layout printed on the sensing surface. For touch typing, users must learn to hold the hands fairly steady over the key layout and rest them on the surface during lulls in typing. They also must try not to grossly overshoot key rows. To attain the performance gains of instantaneous mode switching, a few sensible finger chord gestures must be memorized, but this is not nearly as difficult as learning a chord keyboard typing scheme. And while the elimination of key activation force makes the ergonomics of the surface exceptional, users should still take rest breaks and vary their posture to prevent minute inflammations from accumulating into long-term injuries.

1.2 Summary of Final Device Operation

Obtaining basic keyboard and mouse functionality from the MTS should be easier than operating a keyboard and mouse, but is necessarily somewhat different. Only the more advanced functionality such as text cursor manipulation and command gestures, which conventional pointing devices cannot support, requires substantial learning or adaption on the user's part. Therefore it is assumed in the following that only the functionality necessary for an application and on par with the user's skill level will be enabled at a given time.

1.2.1 Typing

Though development of typing recognition software is not yet complete, the MTS strives to support both touch and hunt and peck styles of typing.

1.2.1.1 Default Key Layout

A default QWERTY key layout (Figure 1.1) is printed on the MTS with key columns morphed to fit an average hand. The layout is pre-morphed because

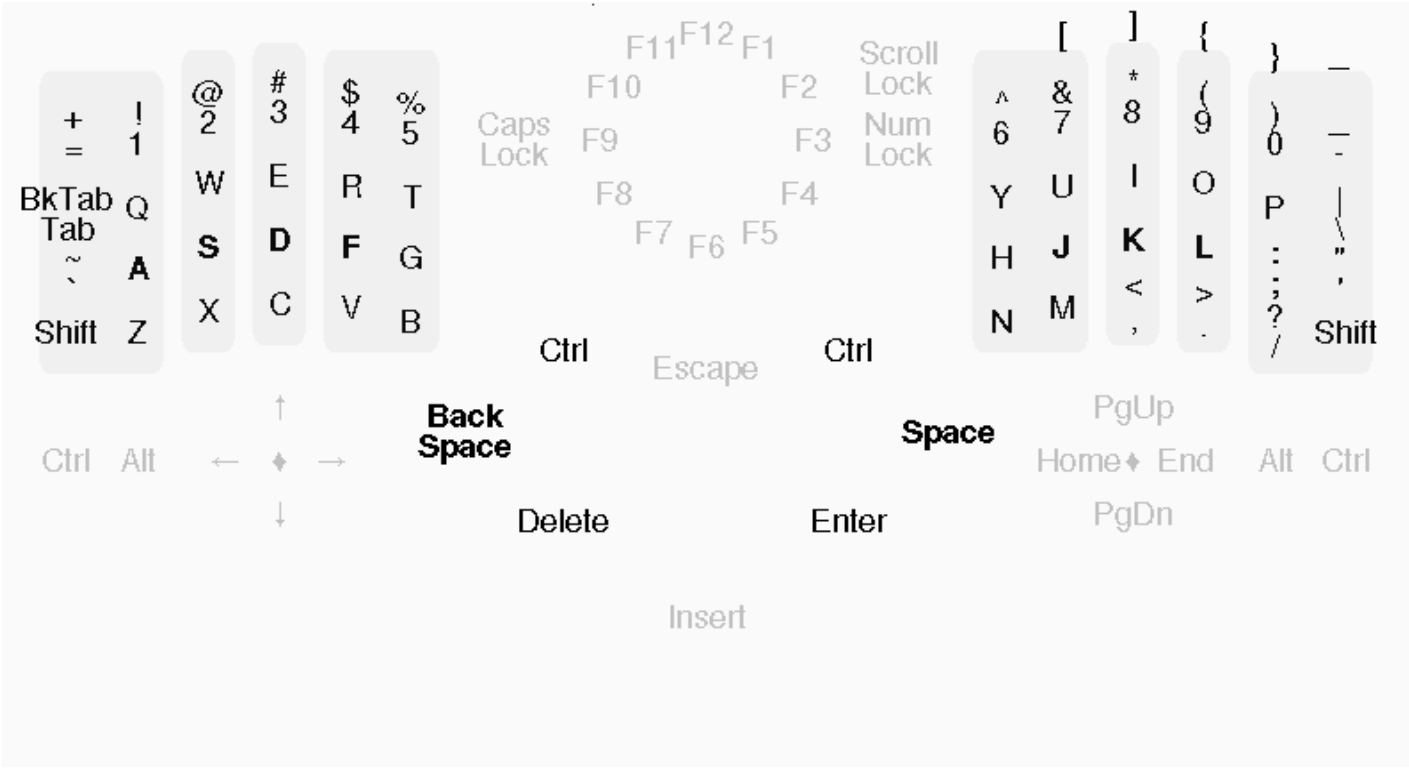


Figure 1.1: Warped QWERTY key layout for the MTS.

without indentations to force the fingers onto a straight home row, the fingertips will naturally fall along an arc. Future versions of the MTS may include indentations or depressions for the home row keys and raised dots at the centers of other keys. The alphabetic number and shift keys are all in their standard relative locations. However, function and editing keys are rearranged (as is often done in laptops) to reduce long-distance hand excursions. Space is placed under the right thumb and backspace under the left, so people who are used to spacing with either thumb will need to adjust. Enter and delete are accessed by extending the right and left thumbs, respectively, as on the Kinesis [29] key layout. The function keys are arranged in a pie [67, 85, 135, 145] at the center of the board where they can be reached easily and invoked with properly angled finger flicks. Arrow and page keys can also be arranged in pies if they are not assigned to chord gestures.

1.2.1.2 Key Activation

The keys are not mechanical in the sense that they do not give when pressed. They are simply areas of the hard surface which are sensitive to quick, light taps by the finger. Thus a key is activated only when a finger touches the surface near a symbol and lifts back off the surface within half a second. If the finger arrives synchronously with other fingers from the same hand or slides around on the surface too much, then the keypress signal will not be generated. This allows the whole hand to rest on the surface and lift off without invoking any action. An auto-repeat or typematic mode for sending identical keypresses is invoked by holding a single finger on a symbol for at least one second while all other fingers are lifted. The repeat rate can be controlled by finger pressure to prevent overshoot at high rates.

1.2.2 Chordic Manipulations

For editing commands and manipulation of graphics the MTS recognizes a variety of *chordic manipulations.* Chordic manipulations are performed by placing a

combination of fingers on the surface at the same time and then sliding these fingers across the surface. The thumb-fingertip combination selects one of the manipulation or command *channels* shown in Table 1.1. Within each channel, the operator can perform:

- a chord tap by quickly lifting all the fingers off the surface after they touch.
- a hand translation, sliding all the touching fingers in the same direction across the surface at the same speed.
- a hand rotation as if turning a jar lid or screw between the thumb and fingertips.
- a hand scaling which pinches the thumb and fingertips together or flicks them apart.

Table 1.2 describes icons for all these chord motions that the MTS recognizes. Tables 6.1 and 1.4 show the most basic mappings between motion channels and command events. Additional mappings for the left hand are shown in Tables 6.2 and 6.4 of Chapter 6. An operator would start learning the mappings from a quick-reference card fashioned after these tables. These simple chord motions should quickly become automatic with use.

1.2.2.1 Pointing

Moving the mouse pointer on the MTS is just like moving it on a touchpad except two adjacent figures excluding the thumb must initially contact the surface, rather than a single finger. After the two-finger chord is initialized, *i.e.*, after half a second, all but one finger can be lifted or the rest can drop to the surface while cursor positioning continues. Once all five fingers are on the surface the hand can be contracted to move in a third axis. Since the finger movements are averaged, stopping all but one finger can cut the sensitivity to one fifth for very fine positioning.

Table 1.1: Legend for finger combination/channel icons.

Channel Icon	*Finger Combination*
	Any 2 fingertips (excluding thumb).
	Any 3 fingertips (excluding thumb).
	All 4 fingertips (excluding thumb).
	Thumb and any fingertip.
	Thumb and any 2 fingertips.
	Thumb and any 3 fingertips.
	Thumb and all 4 fingertips.

Table 1.2: Legend for chord motion icons.

Motion Icon	*Type of Chord Motion*
	Brief tap on surface (one-shot).
	Translation (slide) in any direction.
	Reversible translation up or down.
	Reversible translation left or right.
	Reversible up or down translation, irreversible right translation.
	Translation in a particular direction (one-shot).
	Contractive hand scaling (one-shot).
	Expansive hand scaling (one-shot).
	Clockwise hand rotation (one-shot).
	Counter-clockwise hand rotation (one-shot).

Tapping these two fingers simultaneously on the surface produces a primary mouse click.

1.2.2.2 Dragging

On touchpads, tap-drag and double-tapping are rather clumsy operations. On the MTS, primary double-clicks can be sent by tapping a three fingertip chord just once. Primary dragging is invoked by sliding a three finger chord without the awkward preceding tap of touchpads. Objects can also be resized or rotated during drags by dropping the remaining thumb and finger to the surface and contracting or rotating the hand. On computers which utilize a secondary mouse button, secondary button clicks and drags can be generated from the thumb+two-fingertip channel.

1.2.2.3 Scrolling

Scrolling is initiated by a 4-finger chord, preferably on the hand opposite the pointing hand. Again, by dropping the thumb to the surface, scrolling can be expanded into zooming or rotating the window background. Autoscroll (scrolling momentum) is easily invoked by sliding the four fingers and lifting off the surface in a continuous motion, without decelerating. Browser back and forward is a further variation of the four finger chord consisting of very quick, sliding taps to the left or right. In a graphical user interface (GUI) with a three-dimensional desktop, the thumb+three-fingertip channel could be used to pan and zoom the entire desktop or screen area, rather than a single window background.

1.2.2.4 Text Editing

In text editing or word processing contexts, the chord assignments are split among hands such that the right hand chord controls a mouse cursor operation while the left hand chord controls the corresponding text cursor operation. For example, the right hand two finger chord would move the mouse cursor while the left hand

Table 1.3: Mappings for right hand manipulation channels.

Right Hand Channel	*Chord Motion*	*GUI Action*
••		Primary mouse button click.
••		Mouse cursor manipulation.
•••		Primary mouse button *double*-click.
•••		Dragging/Selection via primary mouse button.
••••		No mapping to avoid accidents.
••••		Continuous scrolling/panning of current window.
••••		Key layout homing.
••••		No mapping to tolerate shifts in resting hand posture.

two finger chord moves the text cursor with the arrow keys. Similarly, the right three fingers would select with the mouse, and the left three with the text cursor via <shift> arrow keys. The left four finger chord would control the scrollbar while the right four finger chord emulated the page keys.

1.2.2.5 Menu Commands such as Cut, Copy and Paste

Even after all this, some room remains in the chord space for common menu commands. Setting the thumb and forefinger down apart and then pinching them together intuitively invokes cut. Copy becomes a simple, simultaneous tap of the thumb and a fingertip. Setting thumb and forefinger down together and flicking them apart invokes paste. A clockwise rotation as if turning a screw saves the current file, and a counter-clockwise rotation pops up the open file dialog. Additional menu commands could be invoked on future systems with handwriting gestures.

1.3 Hardware Summary

All experiments in this dissertation are conducted on a MTS prototype consisting of separate processor and sensor circuit boards.

1.3.1 Sensing Hardware

The MTS prototype has approximately the same footprint as an enhanced IBM PC AT keyboard. Thus it is the first multi-touch device with a sensing area wide enough (20 cm $\times$ 40 cm) for simultaneous use by both hands. An ergonomic arch across the middle of the surface tilts the hands sideways about 15°, reducing forearm pronation and ensuring whole hand resting is comfortable. The active sensing area is divided into 1600 electrode plates (see Figure 2.4 on Page 41). For electrical insulation and low friction, the electrodes are typically covered by a .1 mm thick polymer sheet.

Table 1.4: Mappings for right hand command gesture channels.

Right Hand Channel	*Chord Motion*	*GUI Action*
		Cut (to clipboard).
		Copy (to clipboard).
		Paste (from clipboard).
		Secondary mouse button click (popup menu).
		Dragging/Selection via secondary mouse button.
		Popup application window list.
		Browser Back.
		New file.
		Open file dialog.
		Save the current file.
		Close the current file or subwindow.

The proximity sensors measure the electrode self-capacitance, or capacitance from electrode to ground. This self-capacitance changes when a grounded conductive object approaches the electrode and concentrates electric field lines. Note that some touchpads sense mutual-capacitance between electrodes, rather than self-capacitance, by measuring how a synchronous frequency couples from a drive electrode to an overlapping sense electrode. The processor scans all electrodes every 20 ms, producing a 50 frames per second (fps) stream of *proximity images*.

With suitable array segmentation and interpolation as described in Chapter 3, the centroid resolution for finger-sized objects contacting the surface is about .2 mm in the x direction (width), and .5 mm in the y direction (height). Objects separated by as little as 6 mm in the x direction and 12 mm in the y direction can be distinguished. As objects rise off the surface, position accuracy and distinguishability degrade until 2 millimeters above the surface, whence small objects become undetectable.

Note that the MTS's novel sensor technology is uniquely immune to parasitic capacitances and can therefore be scaled to very large dimensions without degrading the signal-to-noise ratio. The sensor technology is also compatible with very low cost thin-film manufacturing techniques. This means that in high production volumes, MTSs could become as cheap as conventional keyboards. If applied to a flexible substrate, the sensor technology would also be suitable for handheld, portable, and wearable computers.

1.3.2 Signal Processing Hardware

The processor boards contains a digital signal processor (DSP), static RAM and FLASH memory, scanning state machine, and communication ports. A 60 MHz Texas Instruments TMS320C32 floating point DSP is responsible for all scanning, filtering, recognition, and communication algorithms. With 60 MFLOPS peak floating point performance, this DSP is well-matched to the computational demands of

the algorithms developed in this dissertation. By 1999 standards, 60 MFLOPS is moderate performance available in quantities for less than $10 per processor. The DSP is responsible for:

- controlling the sensor array scan.
- forming and filtering the scanned proximity images.
- segmenting the proximity images into groups of pixels distinguishable as flesh contacts.
- tracking motion of each flesh contact across the stream of images.
- identifying which part of which hand, *i.e.*, fingertip, thumb, or palm, causes each flesh contact.
- extracting hand motions from the contacts identified as fingers.
- generating keyboard and mouse events for the host computer in response to motions of particular finger combinations.

These steps are also summarized by the MTS block diagram in Figure 1.2.

One megabyte of flash EEPROM stores the program and multiple user configurations on board. Two 10 kbps PS/2 ports are available for emulating IBM PC keyboards and mice. With PS/2 converter boxes available from Kinesis Corp. [29], mouse and keyboard emulation for Sun workstations and Macintosh computers is also supported. A 1.2kbps RS-232C serial mouse port is included for interfacing to older PCs. A 115kbps RS-232C serial port can exchange configuration and finger tracking information with any host computer capable of running a Java 1.1 MTS monitor application.

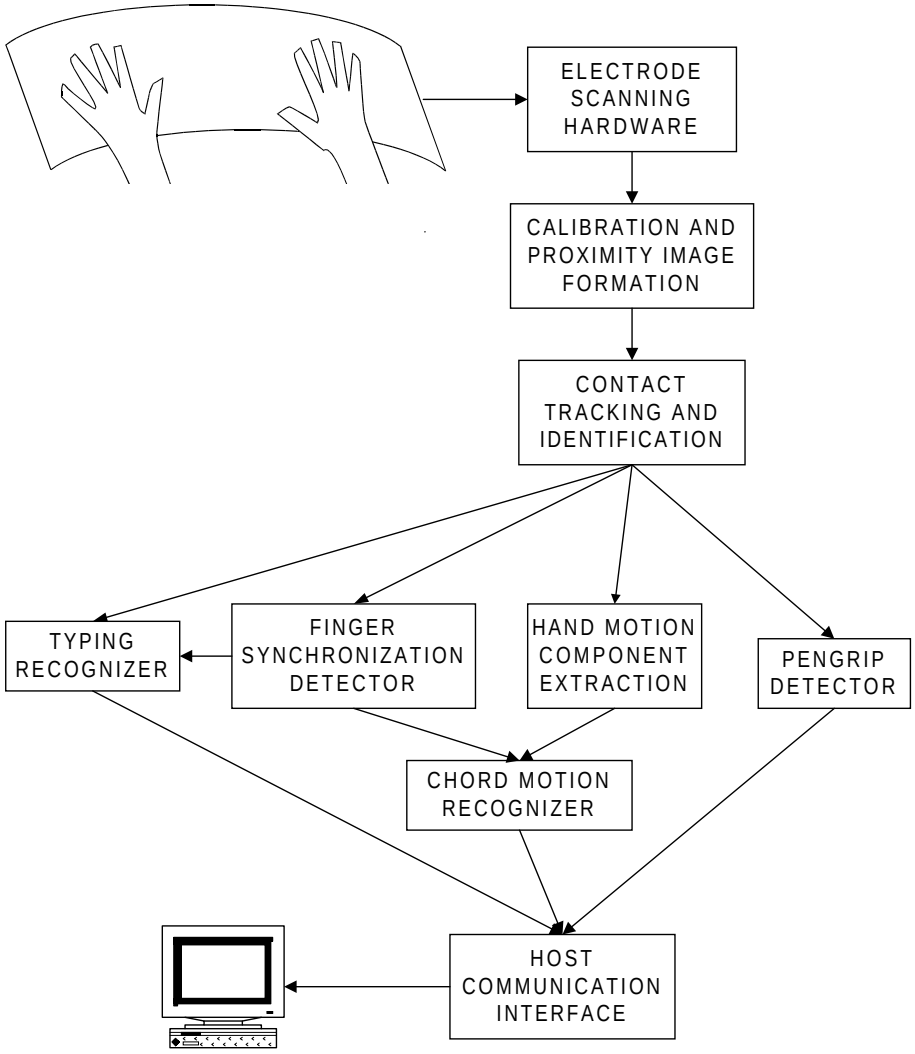


Figure 1.2: Overall block diagram of the MTS hardware and software modules.

1.4 Summary of Contributions

The work described in this dissertation breaks ground in the following areas:

- segmentation of surface contacts in proximity images.
- identification of thumb, fingertip, and palm contacts tracked across successive images.
- translation-invariant sorting of contact points with respect to the inter-attractor angles of an attractor point template.
- partitioning of contacts into left and right hand clusters.
- extraction of independent, 4-DOF velocity parameters from multiple finger paths.
- integration of typing and pointing on the same surface via the distinction between simultaneous and asynchronous finger touchdown.

While this is not the first implementation of a multi-touch device [15, 17, 88, 89, 107], it is the first to fully develop the unique integration potential of such a system. I have encountered and overcome many problems unique to proximity sensing along the way.

Though proximity image segmentation is simplified by the fact that proximity images lack the background clutter and lighting variations which plague optical images, the topology of proximity images also presents special challenges which have not been addressed in previous image processing research. The most difficult of these is the invisibility of hand parts which float above the surface. Also, the low resolution of the proximity sensing array compared to video cameras obscures boundaries between adjacent contacts. The conflicting segmentation needs of fingertips, thumbs, and palms are effectively resolved via feedback of bootstrapped

hand position estimates. Though the segmentation rules developed for the MTS are somewhat ad hoc, any segmentation system will need to utilize the anatomical constraints identified here to overcome proximity image ambiguities.

Correct identification of hand parts from proximity sensing information alone has not been attempted before, apparently because previous researchers of multi-touch devices [130] did not consider it possible. Nevertheless, distinguishing palm contacts from finger contacts on a large MTS is imperative for the motion recognition algorithms to ignore palm motions and allow palms to rest on the surface. While distinguishing fingers from one another is not always necessary or entirely feasible, reliably distinguishing the thumb contact from the other fingers on a hand doubles the number of finger chords which can be recognized compared to just counting those fingers which touch the surface. Identifying the thumb and maintaining a consistent order for other finger contacts also aids extraction of hand motion parameters.

Finding the minimum cost one-to-one assignment of surface contacts to a ring of attractor points is shown to be an elegant solution to the finger identification problem. Each attractor point represents the identity of one hand part such as fingertip, thumb or palm. The attractor points are placed in a ring at default finger locations to capture the shape of a relaxed hand. Translating each hand's attractor ring by a conservative hand position estimate helps stabilize identifications of hand parts which temporarily lift off the surface. An illuminating property of such attractor rings is also proved. If the assignment cost is composed of the sum of squared distances between contacts and their assigned attractor points, the identities of any pair of contacts will not be erroneously swapped unless the vector angle between the two contacts differs from the angle between the pair of attractor points by more than 90°, regardless of whether the attractor ring is properly centered on the hand. This property specifies for a given set of inter-attractor angles exactly how much finger pair rotation will be tolerated before identities are erroneously swapped, and

it also captures in a manner independent of hand translation and size the biomechanical constraint that fingers tend not to cross over one another. When extended to more than two contacts, this property causes the assignment algorithm to sort the contacts with respect to the attractor ring angles and orderings, again regardless of whether the attractor ring is aligned with the hand. A fuzzy thumb classification routine verifies thumb presence with tests of inter-contact angles, separations, and velocities which are not easily incorporated into the attractor framework.

Determining which hand causes each surface contact is also a challenge on surfaces which do not prevent hands from sliding across the middle to the opposite side of the surface. Again, a combinatorial optimization approach is employed which relies on independent attractor rings for each hand. Since the number of hand partition hypotheses to be considered is carefully limited to about a dozen, the evaluation of each hypothesis can be quite extensive, incorporating the consistency of finger identifications for each hand as well as several hand separation and velocity constraints.

Even when position measurements are independent and unbiased, simultaneous placement of an object using more than two degrees of freedom can be difficult for users [152]. As the user approaches the final position along the x and y axes, spurious arm motions can disrupt the final z position, and vice versa. This problem is exacerbated on the MTS by finger motions during hand scaling, translation, and rotation which do not cancel properly. When all five fingers flex on a surface in a hand scaling manipulation, the sum of their velocities gives a net translation toward the elbow. If hand scaling is to modulate a z-axis parameter at the same time as hand translation controls x and y axes, the extracted scaling and translation parameters must somehow be made independent. Biomechanical observations suggests that the thumb and pinky translations during finger flexion cancel. Finger velocity weightings carefully based on this and other considerations attempt to

extract independent translation, scaling, and rotation components from sums and differences of finger velocities. Dead zone filtering with dead zone widths dependent on the distribution of motion component speeds removes the remaining interference between motion components.

This research converges upon what is arguably the best possible way for mouse functionality to coexist with typing on a surface. Asynchronous single finger activity and the 5-finger chord are reserved for typing-related activities, requiring graphical manipulation modes to begin with 2- to 4-finger chords. This is necessary to prevent glancing keystrokes or resting hands from spuriously nudging the mouse cursor. But as an enormous fringe benefit, switching between keyboard and mouse only need involve lifting the fingers and putting them back down in a new chord configuration, rather than hitting an explicit mode-switch button or moving between specialized areas of the surface. To encourage operators to rest the weight of their hands on the surface as much as possible, the chord selection state machine allows all five fingers to drop onto the surface at any time during manipulation after a finger subset has initially chosen the desired chord. New chord channels can also be selected by synchronously raising and touching a subset of the fingers while the others remain resting on the surface.

1.5 What is Not Covered

The proximity sensing and scanning hardware of the MTS was invented by Prof. John Elias and will not be analyzed in this dissertation. However, some specifications for the MTS prototype used in this work (designed by Elias with suggestions from myself), have been given in Section 1.3. Section 2.1.8 includes additional explanation of the interleaved parallelogram electrodes composing the sensing array. Further details of the proximity sensing and array scanning technologies are disclosed in pending patents [159, 162].

The chordic manipulation gestures to be recognized in this work consist of simple linear slides, rotations, and scalings involving multiple fingers. Much previous research has investigated recognition of complex paths or handwriting. With suitable preprocessing, one of the finger chord or pen grip channels could easily be output to an existing handwriting recognition engine such as IBM's Pen for OS/2 [142]. Thus no attempt will be made to recognize complex paths in this work. Though the pen grip hand configuration is briefly introduced, vertical smearing by the prototype electrode array prevents the pinched fingers of this configuration from being segmented reliably. Therefore the preprocessing of the pen grip images necessary to feed them to a handwriting recognizer is not investigated here either.

This research bucks the trend toward machine learning in pattern recognition systems. This is justified by the fact that hand tracking and identification from proximity images involve previously unexplored patterns, yet these problems easily lend themselves to experimentation through the author's own hands. Throughout the development of the MTS, incorporation of additional biomechanical or anatomical observations into the algorithms has always been much more effective at reducing recognition errors than parameter tweaking. Now that this research has identified the constraints crucial to robust performance, future work can attempt to fit them into a machine learning architecture for statistical optimization over a wide population of MTS users. Since the attractor-based identification algorithm is by design very tolerant of variations in hand size, only marginal performance gains are expected from optimizing or adapting it to various hand sizes. A more fruitful line of inquiry would be extension of Rubine's automated gesture recognition work [130] to tolerate idiosyncrasies in hand motion patterns during chordic manipulation.

Though the MTS utilizes sophisticated typing recognition algorithms to find the most likely character sequences in response to typing motions [158], these algorithms have not yet been perfected and are not included in this dissertation. Only

the finger synchronization detector which distinguishes typing motions from chordic manipulations is covered.

Since the MTS prototype was only completed recently, there has not been time to conduct formal user studies. However, I do include my testimonial and detailed observations from using the MTS as my primary graphics, text and command entry device during preparation of this document. While this may not stand as a formal scientific evaluation, anyone who has written a dissertation will appreciate that it is remarkable for such a radical new input device to meet all the diverse and intense interaction demands of dissertation editing by one who already suffers from RSI.

1.6 On the Design of Ergonomic Input Devices

Repetitive strain injury (RSI) causes loss of tendon and muscle strength in severe cases, but more universally it causes painful loss of endurance [117]. Strength can usually be recovered in a few weeks with proper stretching and mild exercise, but endurance can take months or years to recover [117]. Doctors tend to downplay the impact of device design on these injuries, placing most of the blame on improper use of devices, poor posture, and working continuously to meet deadlines. As in any engineering field, radical new device designs can have unforeseen consequences. One major source of unforeseen consequence in device design is shifting too much workload to previously dormant muscles [117].

1.6.1 What is the Role of Ergonomic Device Design?

While I agree with Pascarelli [117] that poor user habits play a large role in most RSI cases, I have found that after my bad habits were corrected, I could still tolerate minimal activation force devices several times longer than medium or high-force devices. This relationship has held for rollerball pens, optical mouse buttons, capacitive touchpads, free-wheeling finger rollers, and low-force keyswitches. What

these minimal force devices have in common is that they allow my hand to remain essentially relaxed during use. Note that unlike isometric pointing sticks [131], which actually seem to cause me tension buildup, these minimal force devices do not try to eliminate motion. Note also that endurance problems are often so severe that not even zero-force devices allow work to continue indefinitely without painful relapse. While my anecdotal evidence does not prove that minimal force devices similarly benefit all users, it is certainly consistent with the RSI causation theories cited in Appendix A.

In defense of those users with poor habits, the poor ergonomic design of standard devices encourages people to compensate via awkward postures or excess effort. For instance, to hold their hands on home row of a standard keyboard, people often rotate their hands outward and hold their elbows away from the body, straining their wrists and shoulders. Or when faced with a mouse button which is hard to press, people will clench the mouse and apply several times the necessary force to ensure that the button clicks. Conscientious users can learn proper techniques which limit the risk of injury on plain keyboards and mice, but this takes a lot of self-discipline [117].

Devices *can* be designed so that even the most recalcitrant users gravitate toward proper postures and usage habits. For example, most people correct their wrist posture instantaneously when they adopt a split keyboard [65, 114]. As a hypothetical example, a compliant touchpad surface might soften fingertip impact but allow combative people to keep banging on the surface. If presented with a hard touchpad surface, the painful jarring of hard taps would force the same people to adopt a lighter, and ultimately healthier, touch. The problem shifts from getting people to consciously learn and apply proper technique to simply getting them to adopt and use a more ergonomic device.

Unfortunately, moderating work intensity and taking frequent rest breaks are

very important preventive measures which the device designer can do little to encourage. Users may remain crouched all day at their terminals because of their own compulsion, their bosses' whips, Internet addiction, or other psychosocial factors. Once injured, the RSI patients who cope best are those who have the most control over their work schedules, so they can slow down whenever pain reappears [117]. The tendency for computers to suck up all allotted time also limits the actual benefits of efficient devices. Clever devices may boost productivity or allow prolonged activity, but if the effective amount of hand usage remains the same, RSI patients will remain at the edge of their bodies' tolerances.

1.6.2 Ergonomic Design Objectives

The following ergonomic design objectives have been formulated based upon the ergonomics research reviewed in Appendix A, recommendations gathered from doctors, and my own experience using many competing input devices and methods. Though typical methods for meeting these objectives are given here, their justification is left for Appendix A.

1.6.2.1 Minimize device activation force

Muscular effort is believed to be a major factor in RSI, but most input device designs do not try to minimize it. Though low-force mechanical mouse buttons and keyswitches reduce tension and fatigue somewhat, the most dramatic benefits come from zero-force capacitive or optical sensors.

1.6.2.2 Minimize repetitive action of the same muscles

Otherwise known as the art of distributing workload across body and time. I have encountered four general ways to avoid overuse of any particular muscle:

customize software to automate redundant tasks Effective methods include aliases for long text commands, command-line histories, word completion, clever GUI design, writing scripts for frequent

batch tasks, and macro recording and playback of repetitive mouse sequences.

reduce homing motions If excessive, motions between devices or surface regions become annoying and discourage the user from switching to the most appropriate device for a task. Modes can be more efficiently indicated with hand or finger configuration.

support high-level gestures A complex sequence of low-level device actions can often be expressed in one continuous motion if a device has alternative manipulation channels available.

alternate muscle or limb usage At the hand level this entails employing various fingers or finger combinations alternately, rather than the same finger all the time. At the body level, the feet are appropriate for mode-switching, the vocal cords for text input, and the eyes for indicating the focus of attention. A worthy device must be either highly compatible with fellow devices which employ complementary muscle groups, or it must be a universal device which draws upon most limb muscles evenly.

1.6.2.3 Encourage neutral postures

The design space for comfortable keyboard, mouse, and trackball shapes has been explored aggressively. At the truly beneficial end of the spectrum, split keyboards offer quick relief for mild wrist pain, while thumb-operated trackballs can quickly cause DeQuervain's syndrome [117] if used for heavy mouse work such as drafting. Devices should not discriminate against left-handed users. Flat surfaces are literally the most neutral because they make no assumptions about hand size or shape.

1.6.2.4 Allow variation of posture

Not even neutral postures should be held forever, as fancy device shapes encourage. Changes in posture rejuvenate blood flow and shift loads to different muscles. Plain surfaces have the advantage here because they can be flexible, portable, and operated from a variety of hand positions.

1.6.2.5 Minimize user anticipation

Rather than taking healthy micro-breaks, users may remain tense while expecting a response from the computer. Lags in visual feedback as small as 75 ms during direct manipulation can reduce pointing performance [64, 99]. Open-loop, high-level control gestures make the user less dependent on intermediate computer responses.

1.6.2.6 Do not discourage rest breaks

Software is available to remind people to take a break after a certain amount of time or a certain number of keystrokes. All the device designer can do is avoid wires or gloves attached to the body which might discourage people from regularly leaving their workstations.

1.6.3 Can so many ergonomic objectives be met at once?

The healthiest existing devices address these criteria only piecemeal. For example, the concave-keywell keyboard from Kinesis Corp. [44] encourages neutral postures and distributes workload according to finger strength, but it does not eliminate key activation force or allow posture variability. Touchpads from Cirque [51] and Synaptics [143] eliminate device activation force but overuse one or two fingers with inefficient, low-level mouse operations. Tablets with styli support high-level gestures and handwriting, but the stylus can also be an encumbrance which is incompatible with simultaneous keyboard use.

With a zero-activation-force surface at the foundation and abundant finger chords to distribute effort and support gestures, this dissertation aims to demonstrate that all of these ergonomic objectives can be addressed simultaneously.

1.7 Outline

Organization of the dissertation roughly follows the MTS block diagram of Figure 1.2 on Page 17, with the exception that the typing recognizer and pen grip detector will not be covered in full, nor will implementation details of the electrode scanning hardware or host computer interface.

Chapter 2, Proximity Image Formation and Topology, begins with a review of related hand motion sensing technologies such as data gloves, video cameras, and touchpads. It then describes calibration of the proximity sensors for the MTS and proximity image formation. Chapter 2 concludes with examples of several important hand configurations captured by the prototype sensor array and points out the topological characteristics unique to such proximity images.

Chapter 3, Hand Contact Segmentation and Path Tracking, presents the anatomical constraints and segmentation rules necessary to separate different parts of the hand in proximity images. The discussion also covers feature extraction for segmented contacts. After explaining segmentation results for a variety of hand configurations, methods for tracking contacts across successive proximity images are discussed, including techniques to reliably detect finger touchdown and liftoff.

Chapter 4, Finger Identification and Hand Position Estimation, begins with a review of hand gesture recognition techniques. The chapter then introduces combinatorial optimization algorithms which successfully identify the hand and finger which cause each surface contact. A conservative hand position estimation algorithm helps to stabilize the identifications.

Chapter 5, Chordic Manipulation, reviews the advantages of bimanual and high-DOF manipulation and explains how they are achieved on the MTS. A finger synchronization detector distinguishes asynchronous taps intended as typing from synchronous finger chord motions intended as pointing or gesture commands. Carefully weighting the velocities of particular fingers ensures that velocity components

extracted from hand translations, rotations, and scalings remain independent.

Chapter 6, Future Directions and Conclusions, presents the author's personal testimonial after using the first fully functioning MTS prototype for three months to prepare this document. This testimonial underscores issues which deserve formal evaluation by user studies and offers suggestions for future MTS enhancements. Chapter 6 concludes with an outline of how software applications and operating systems should be modified to take full advantage of bimanual, 4-DOF input such as the MTS provides.

Appendix A, Ergonomics for Engineers, reviews physiological and epidemiological evidence of how poor or fixed posture, activation force magnitude, repetition, and lack of rest periods conspire to cause RSI. It includes issues which have previously arisen in the design of ergonomic keyboards, and is intended for engineers faced with input device design who wish to understand the reasoning behind the ergonomic principles of Section 1.6.

Appendix B contains non-linear interpolation techniques for diminishing the vertical biases from interleaved parallelogram electrodes.

Appendix C explains the convergence problems encountered when applying bubble sort to the attractor ring assignment problem.

Chapter 2

PROXIMITY IMAGE FORMATION AND TOPOLOGY

Limited hand and finger tracking experiments have previously been conducted with a variety of sensing technologies. This chapter begins with a review of these sensing technologies and explains why proximity sensing arrays are particularly well-suited for everyday applications of hand tracking. Then the chapter discusses proximity image pre-processing such as background object removal, sensor offset adaptation, and electrical noise filtering. The chapter concludes with a sampling of proximity images which illustrate the typical features and arrangements of hand contacts. This hand topology section is particularly important to the understanding of the contact segmentation and identification algorithms in Chapters 3 and 4, which rely heavily on relative contact shape and position constraints.

2.1 Related Methods for Hand Motion Sensing

Hand position and motion can conceivably be detected with mechanical or electromagnetic sensors attached to the hand, with remote optical or acoustical sensors, or with proximity or pressure sensors mounted on an object in the user's environment. At first glance the attached sensor methods seem advantageous because they can capture three-dimensional hand activity in free space, unconstrained by the physical form factor of an interfacing object. Data gloves and computer vision systems have been popular in virtual reality experiments for this reason. Such systems are clearly appropriate for capturing the free-space hand gestures and sign

language as they appear in communication between humans, but several factors make them impractical for everyday human-computer interaction.

2.1.1 Free-Space Gestures

The first problem lies with holding or slowly adjusting hand position in free space. The quick, relative motions of sign language may be easy to perform, but holding the unsupported hands out in front of the body for extended periods is very tiring [152, 153]. In such postures fingertip positions are also somewhat unstable, so considerably less precision is possible than when some part of the hand or arm rests against a firm object. Also, it is very difficult for a computer to distinguish motions intended to be instructions for the computer from postural adjustments or gestures to co-workers. This is known as the gesture saliency problem. To appreciate the difficulty of this problem, consider how often we humans mistakenly think someone is gesturing at us when the gesture is actually intended for someone behind us or no one at all. If the direction of gaze of the sender is not known, determining the intended recipient of gestures is even more troublesome.

2.1.2 Data Gloves

Free-space motion sensing technologies have limitations as well. Though DataGloves [148] can potentially capture the entire range of finger flexion and extension, in practice the flexion sensors are imprecise yet expensive and cumbersome to wear. Furthermore, as a bodily attachment, gloves must often be removed when the user resumes non-computer tasks. This is both a practical disadvantage and an ergonomic disadvantage because it discourages users from taking rest breaks and mixing in non-computer tasks which rely on other muscle groups. FakeSpace, Inc. [36] markets *pinch* or *chord gloves* for virtual reality systems which detect contact between electrically conducting fingertip pads rather than general flexion and extension of the fingers. The lack of flexion sensors reduces cost, and consistent with

the design philosophy of this dissertation, such physical fingertip contact turns out to be more reliable and easier to learn than free-space finger motion gestures [60].

2.1.3 Video Gesture Recognition

Computer vision technologies avoid the encumbrance of wearing gloves but cannot always infer fingertip location. Assuming decent lighting is available, much of the luminosity information that a video camera supplies is unnecessary for finger tracking, and must be filtered out with computationally intensive algorithms [115]. The body of the hand can occlude the fingertips at some camera and hand angles. Occlusion and limited camera resolution also make it very difficult to determine exactly when the fingers touch a surface.

2.1.4 Benefits of Surface Contact

Most importantly, the emphasis on hand tracking in three-dimensional free space ignores the long history of manipulating hand tools and musical instruments which provide rich haptic feedback as the tool is acquired. While economics may preclude customizing the shapes of general-purpose input devices as much as hand tools are customized, detection of contact with a physical surface provides, at the bare minimum, a clear demarcation between motions on the surface that the computer is intended to recognize and motions away from the surface that the computer should ignore. Though individual finger activity on a surface is constrained to two-and-a-half dimensions, Chapter 5 will demonstrate that extra degrees of freedom can be extracted from rotational and scaling motions of multiple fingers on a surface. For many applications the improved clarity of user intent and tactile feedback that surface contact imparts will more than make up for the slight reduction in movement freedom.

2.1.5 Sensing Finger Presence

Technologies which have been applied to detecting finger or stylus contact include resistive membranes, surface acoustic wave, active optics and finger capacitance sensing (see Lee's 1984 Master's Thesis [88] for an early review). Most implementations are limited to unambiguous location of a single finger because they rely on what Lee calls "projective" sensor matrices. In a projective matrix (Figure 2.1a), one sensor element is allocated to each row and column at the edge of the active

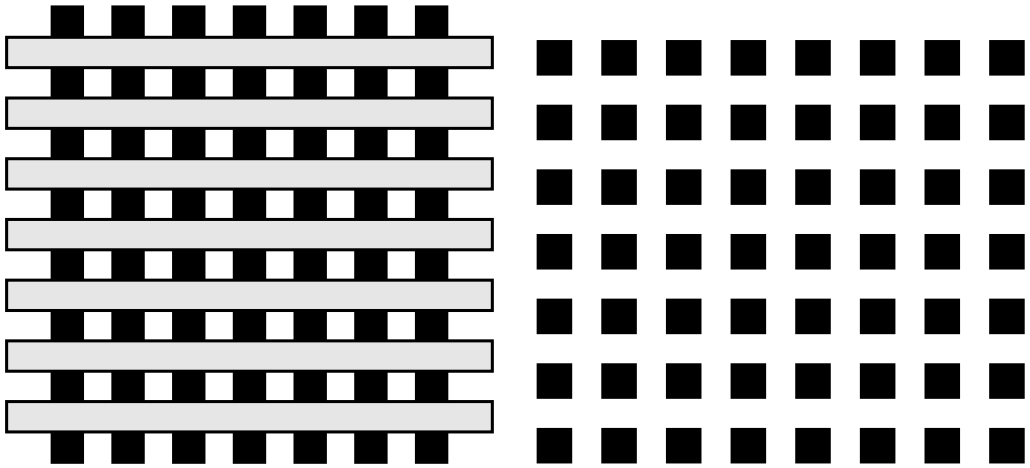


Figure 2.1: The two basic multi-touch proximity sensor arrangements. In a), "projective" row and column spanning sensors integrate across each row and column electrode and only need connections at the edges of the matrix. Touching fingertips can be counted by counting the maxima in the column signals assuming the fingertips lie in a roughly horizontal row unobstructed by thumb or palms. The square sensors in b) only integrate over the local square. The exact locations of any number of fingertip-sized contacts can be interpolated from the 2D array of square sensors, but a connection matrix must be run underneath the sensor array to connect the sensors to signal processing circuitry.

area. Finger presence anywhere along a row will register on that row's sensor, so that a finger affects roughly one row and one column sensor. While the total number

of sensors needed is related to only the square root of the active area, multiple finger contacts can confuse these systems [88]. As was true in 1984, the surface acoustic wave and infrared touchscreens as well as capacitive touchpads on the market still suffer from this limitation.

Some devices on the market partially utilize multiple fingers despite the ambiguities of projective sensing. For example, touchpads manufactured by Logitech, Inc. [15, 78] for laptop computers are able to detect the presence of up to three fingertips. The patent to Bisset and Kasser [15] explains that this is done by assuming the fingers lie in a row and counting the number of maxima in the column projection. However, as will be seen in Figures 2.2 and 2.3 below, this projection maxima counting method becomes ambiguous for larger touch surfaces in which one hand part can intersect the same column as another, such as when both fingers and palms touch the sensing area or the hand rotates so fingers lie diagonally or in a column.

Figures 2.2 and 2.3 demonstrate the limitations of this projection approach compared to the two-dimensional arrays of sensors (Figure 2.1b) to be discussed in Section 2.1.7. Fingertip, thumb, and palm heel surface contacts are simulated with two-dimensional Gaussians of varying widths on the 2D square grid. The grid samples the Gaussians at 2.5 mm intervals such as would occur in a capacitive sensing array with moderate spatial resolution. The darkness of the squares is proportional to the finger capacitance or proximity sampled at the square. The projective signals which would be measured from the row and column spanning electrodes of Bisset and Kasser [15] are simulated by integrating over each row of the 2D array to obtain the horizontal bar plots to the left of each grid and by integrating over each column to obtain the vertical bar plots under each grid.

Figure 2.2 shows the projection sensing ambiguities which can occur when the fingertip row is not horizontal, but lies diagonally instead due to various hand

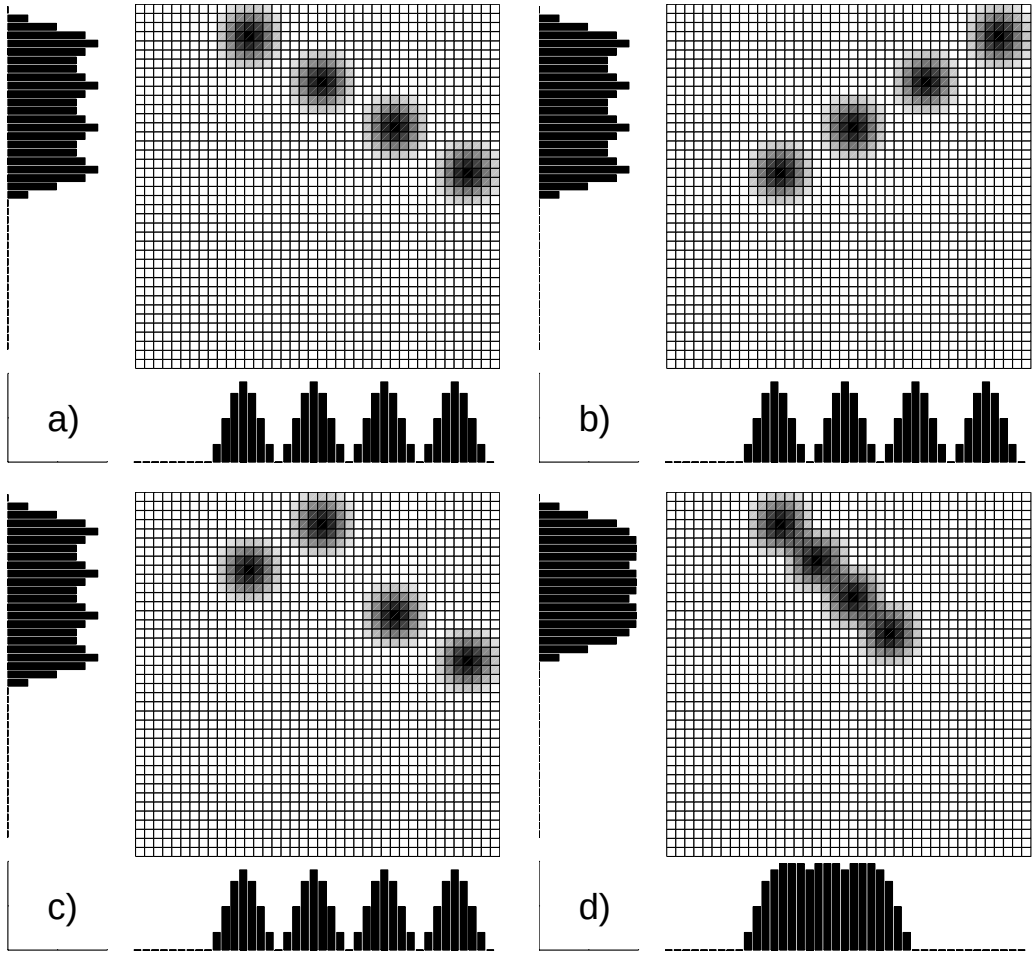


Figure 2.2: Projection sensor ambiguities for various diagonal arrangements of fingertips. The different fingertip contact arrangements shown on the square sensor grid in a)-c) all produce the same row and column projections (horizontal and vertical bar plots), preventing the projection method from determining the hand rotation, though it can still count the fingertip maxima. In d) the fingertips are so close together that the projection minima between fingertips disappear, preventing fingertip counting, though the diagonal minima are still discernable in the square sensor grid.

rotations. In Figure 2.2a-c four maxima appear in both the row and column projections (bar plots), indicating at least four objects are touching the surface, but the projections are the same in each case even though the fingertip arrangements (grid) differ. The same projections could be obtained from a 4×4 array of 16 fingertips also, though most human operators will not have that many fingertips. In Figure 2.2d the fingertips are so close together in their diagonal row that the projection maxima merge, though local maxima are still clearly separated by diagonal partial minima in the sampled 2D array.

Figure 2.3 shows how fingertip counting from projection sensors is occluded by the presence of thumb and palms in a neutral hand position. In Figure 2.3a four fingertips lie in a slight arc, producing four maxima in the column projections and one in the row projection. Figure 2.3b includes the thumb in nearly the same column as the index fingertip, causing an additional maximum in the row projection (horizontal bars) only. The index fingertip is removed in Figure 2.3c; because the thumb is still in the same columns, the number of projection maxima does not change, though the amplitudes change somewhat. Because the amplitudes also depend on how lightly each finger touches the surface, the change in projection amplitudes cannot reliably resolve this ambiguity; the amplitude changes could also be a result of a lightening in hand pressure. In Figure 2.3d the palms touch as well, leaving three maxima in the row projection but causing the column projection maxima to merge into just two. Therefore from the row projection one could surmise that some palms, the thumb, and some fingertips are touching, but one can no longer tell how many fingertips are touching because the palm column projections get integrated with and obscure the fingertip column signals.

As Lee points out, measuring projections from additional angles such as diagonals can help disambiguate multiple contacts, as is done in tomography systems,

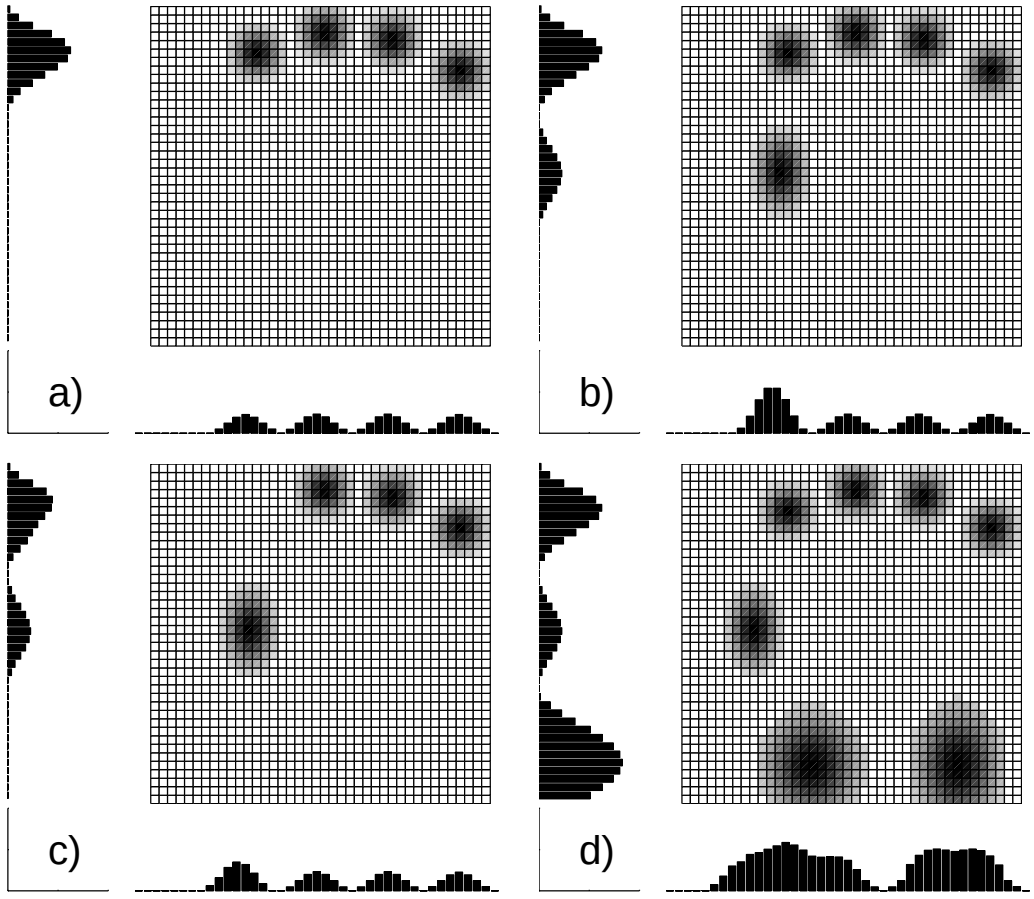


Figure 2.3: Ambiguities in projective sensing caused by presence of the thumb and palms in the same columns as fingertips. a) simply contains a slightly arched row of fingertips producing four column projection maxima (vertical bars at bottom) and one row projection maximum in the horizontal bars. Adding a thumb contact in b) adds a row maximum but not a column maximum because the thumb intersects nearly the same columns as the index fingertip. Removing the index fingertip in c) does not chance the number of projection maxima, meaning fingertips cannot be counted reliably in the presence of the thumb. Adding the palms in d) further obscures the fingertip row projection maxima, which get merged with those of the palms.

but details inside concave contacts will still be undetectable [88]. The number of unambiguously locatable contacts is generally one less than the number of projection angles utilized [88]. McAvinney's "Sensor Frame" [107, 108, 129], an attachment to the screen of a computer monitor which senses intersection of fingers with infrared beams from four directions, utilizes this tomography approach to unambiguously locate up to three fingers.

2.1.6 Tactile Imaging

This complex tomography approach can be avoided with a regular two-dimensional array of individually addressable sensors (Figure 2.1b), in which each sensor corresponds to a pixel in a "tactile image." Layered resistive-membrane pressure sensors can be constructed economically in this configuration, but their substantial activation force is ergonomically inferior to zero-activation-force proximity sensing. Another approach is to place a camera under a translucent tabletop and image the shadow of the hands [81, 110]. Unfortunately the bulky optics under the table will limit portability and leg room, and such systems cannot differentiate finger pressure [88]. Active optical imaging with an array of infrared transmitters and receivers on the surface could easily detect finger proximity, but would be prohibitively expensive and power consumptive.

2.1.7 Capacitance-Sensing Electrode Arrays

The remaining option is to measure the capacitance between the fingers and an insulated array of metal electrodes. The presence of a finger effectively increases the electrode capacitance to ground since the capacitance between the conductive fingertip flesh and an electrode plate is typically a few pF but the capacitance of the human body with respect to earth ground is relatively large (about 100pF) [88]. Since the capacitance between parallel plates drops quickly in inverse proportion to the distance between the plates, this technique can only detect fingers within a

few millimeters of the electrodes. Spatial resolution increases dramatically as the fingers approach the electrodes. Precision of .2 mm can easily be obtained with 4 mm electrode spacings by computing a finger centroid, *i.e.*, interpolating between neighboring electrodes. The capacitive technique also indicates finger force up to a couple Newtons because the effective capacitor area increases as the fingertip pulp flattens against the surface [134]. While the limited proximity sensing range of electrode arrays ensures fingertip proximity information is clear and uncluttered, it also prevents detection of the finger joints and palms unless the whole hand is flattened against the surface.

Lee built the first such array in 1984 with 7mm by 4mm metal electrodes arranged in 32 rows and 64 columns. The "Fast Multiple-Touch-Sensitive Input Device (FMTSID)" total active area measured 12" by 16", with a .075mm Mylar dielectric to insulate fingers from electrodes. Each electrode had one diode connected to a row charging line and a second diode connected to a column discharging line. Electrode capacitance changes were measured singly or in rectangular groups by raising the voltage on one or more row lines, selectively charging the electrodes in those rows, and then timing the discharge of selected columns to ground through a discharge resistor. The principal disadvantage of Lee's design was that the column diode reverse bias capacitances allowed interference between electrodes in the same column. Even with 2048 electrodes and suitable interpolation between electrodes, the electrode spacing was probably too coarse to reproduce the fine mouse positioning achieved with current single-finger touchpads [46–48, 50, 51, 111]. Though its scanning rate depended irregularly on the number of and positions of surface contacts, for ten fingers it would have only been able to achieve 1-5 fps, which is much too slow for either typing or gesture applications.

Rubine [129, 130] reports seeing another multi-touch tablet demonstrated at AT&T in 1988 by Robert Boie which could detect all ten fingers. It boasted a 30

fps frame rate and resolution of 1 mil (.025 mm) in lateral position and 10 bits in pressure. Possibly it measured sensor capacitance with the synchronous detection technology in a 1995 patent by Boie *et al.* [17] that briefly mentions multi-touch tablets as an application.

2.1.8 The MTS's Parallelogram Electrode Array

The MTS contains a 16 × 96 electrode array (Figure 2.4) much like those in the above multi-touch tablets. It employs a special wedge electrode geometry to reduce the number of rows necessary by a factor of three without causing serious non-uniformities in vertical position interpolation. This reduction in electrode count speeds fabrication of research prototype arrays by lowering the discrete part count, but would not necessarily be beneficial for volume manufacturing techniques.

Rectangular electrodes (Figure 2.5) like those used by Lee [88] are more sensitive to vertical position changes near the top and bottom of the electrodes, where it is possible to interpolate between two electrodes, than in the middle of an electrode. If a finger is in the middle, the electrode is so tall that the electrodes above and below do not register enough signal to get a reliable interpolation.

In contrast, the vertically interleaved parallelogram electrodes interpolate via their physical geometry. The ratio of the horizontal cross-sections between electrodes in a column varies continuously with vertical location of an object (Figure 2.6a-d)). Though this improves uniformity of vertical interpolation compared to rectangular electrodes of the similar height, it also has the effect of vertically smearing signals, making it difficult to distinguish objects which appear in the same electrode column less than one row spacing apart. For research prototyping purposes this is tolerable because the fingers tend to lie in a row, no more than one per column. However, once in awhile the thumb or pinky pass behind and intersect columns of the other fingertips, becoming indistinguishable from the fingertip in front of them (see Section 2.3.3). Also, as is discussed in Appendix B, vertical interpolation biases do arise

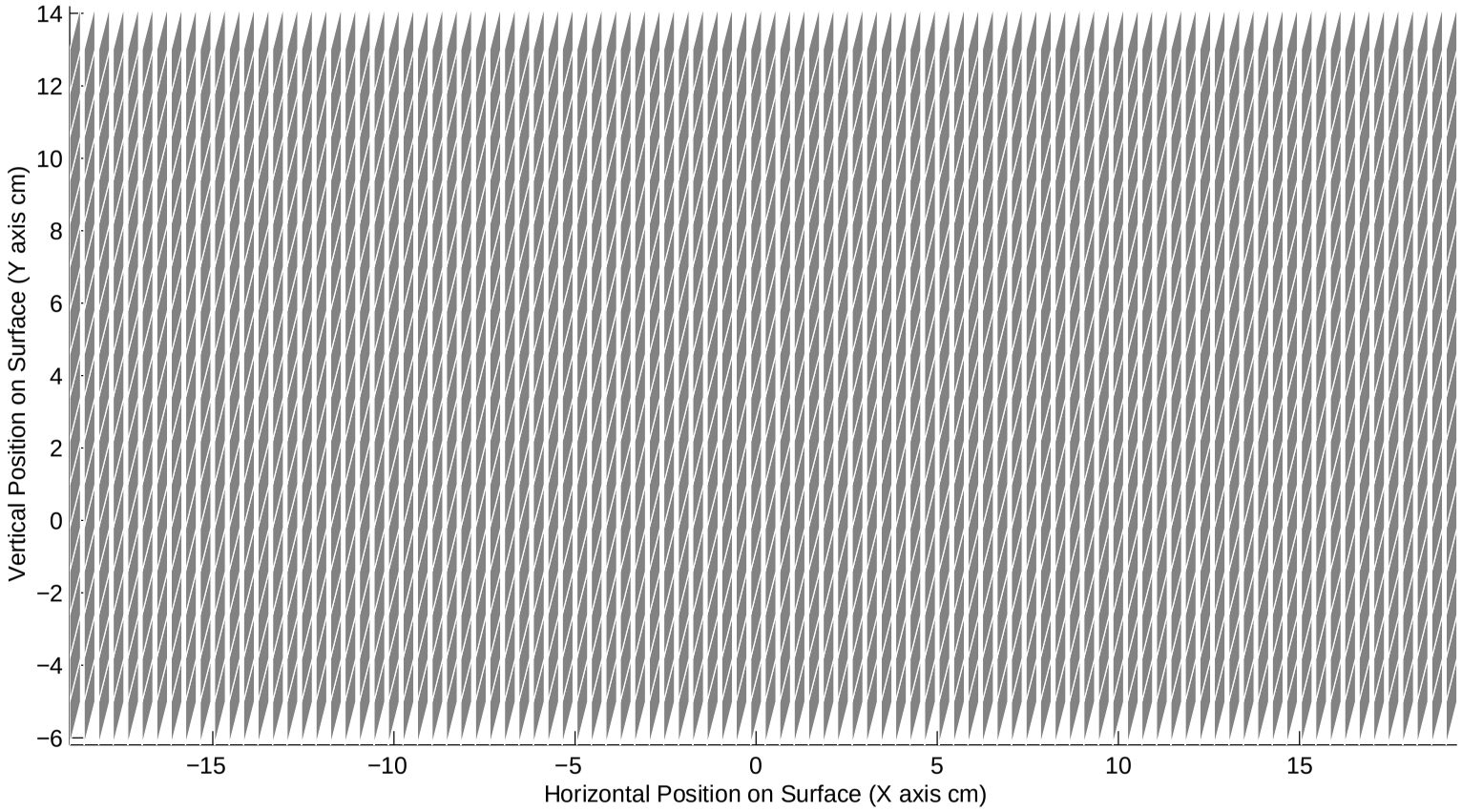


Figure 2.4: Diagram of electrode layout for the entire 16 × 96 parallelogram electrode array. Row pitch is 1.2 cm and column pitch is 0.4 cm, but electrodes are only 0.25 cm wide.

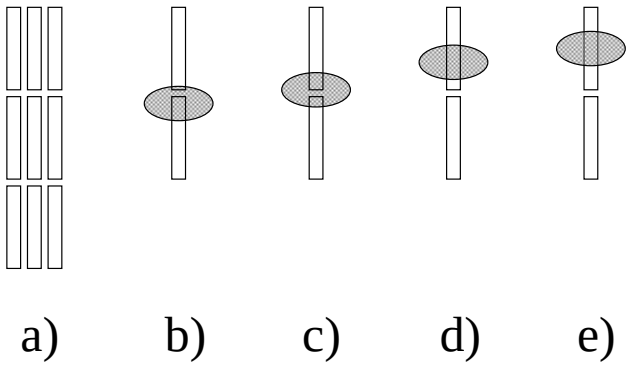


Figure 2.5: A 3×3 section a) of rectangular electrode array. Vertical interpolation between top and bottom electrodes works in b)-c) but not in d)-e).

for small contacts which are not centered on or between columns of the parallelogram electrode array. Thus a commercial product, especially one which attempts to recognize a handwriting grip or stylus, would have to abandon the electrode count savings of this scheme for traditional square electrodes and a smaller row spacing.

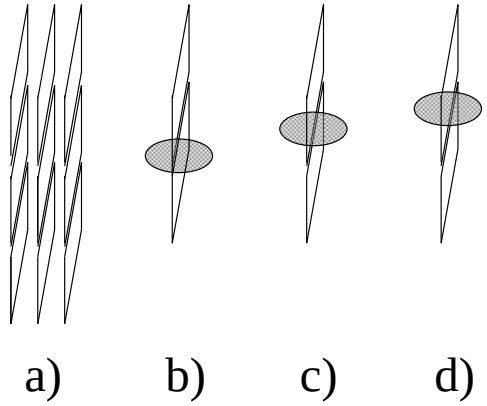


Figure 2.6: Vertical interpolation on the parallelogram electrode array is uniform in a)-d) since ratio of hatched cross sections on top and bottom electrodes changes gradually.

2.1.9 No Motion Blur on MTS

Another important characteristic of the MTS is that the sensing array multiplexes much of the integration, buffering and quantization circuitry. Therefore the capacitance of each electrode is measured over a relatively short period of a few hundred microseconds compared to the total array scanning period of ten to twenty milliseconds. This contrasts with the CCD arrays typically used in video cameras which integrate incoming photons at each pixel over most of the period between readouts. An advantage of the MTS's relatively short integration time is that MTS proximity images do not exhibit motion blur. However, if the scanning rate is not fast enough, quick finger taps over an electrode can occur entirely between measurements of that electrode and be completely missed. When tapping key regions during touch typing, fingers usually remain on the surface for at least 50 ms, but the scan period must be somewhat smaller than this for reliable detection. During the experiments conducted for this dissertation, the array scan frequency or frame rate has been set to 50 fps (corresponding to a period of 20 ms), which ensures that each finger tap shows up in at least one scan. However, at this rate the peak finger pressure as the fingertip bottoms out onto the surface in the middle of the tap cannot be measured accurately because the single scan detecting the tap might occur near the beginning or end of the tap cycle when the finger is barely touching the surface. Minor changes to the scanning hardware can easily push the frame rate to 100 fps, which will allow peak finger pressure to be measured fairly accurately even for extremely quick taps.

2.2 Tactile Image Formation and Background Removal

While designing a tactile sensor array for robotic fingertips nearly 20 years ago, Danny Hillis [59] realized how much easier touch imaging is than computer vision:

> ... analyzing a tactile image is like analyzing a visual image with controlled background, illumination, and point of view ... the properties that we actually measure are very close, in kind, to the properties that we wish to infer.

Comparing background segmentation techniques in vision-based and tactile hand imaging systems will verify his insight.

2.2.1 Optical Image Segmentation

Ahmad's real-time 3D hand tracker [3] segments the background by matching image patches to known skin color histograms, but to keep up with frame rates (30 frames per second) it must limit the skin search region and adaptively subsample the image. Finger positions are obtained by fitting ellipses to the segmented hand patches. The total hand patch area weighted with a centered Gaussian roughly indicates the distance between hand and camera. Ahmad also tries to recover finger joint angles, information which data gloves give directly, by finding fingertips and learning an inverse mapping from fingertip and palm position to intermediate joint angle. This feature of the tracker becomes unstable due to fingertip detection failure if the hand is not roughly normal to the camera.

The Digital Desk [154–157] is a system pioneered at Xerox for combining interaction with paper and digital documents. The system contains both a computer screen projector and zoomable cameras mounted high above the user's desk. The cameras both track hands and recognize text from paper documents lying on the desk. Since the vision system cannot determine exactly when fingers actually touch the desk surface, a microphone is placed under the desk to "hear" finger taps and thus emulate mouse clicks. Crowley and Coutaz [30] consider color, correlation tracking, principal components and active contours for following a pointing object on a digital desk. In the correlation method, a previous image of a fingertip is used as a reference template for correlations with the next image. The new finger

position is indicated by the amount of template image shift which minimizes the sum of squared differences between template and image. Again, the computational costs of the correlation limit the template search region and thus the maximum trackable finger speed.

2.2.2 Methods for Proximity Image Formation

Background segmentation of proximity images from electrode arrays is much easier because extraneous objects are not expected to be visible in the background. Paper or plastic left over the electrodes do not register on capacitive proximity sensors, nor do small metal objects unless they are deliberately grounded. However, spatial non-uniformities in the parasitic capacitances of discrete components and signal lines may cause background measurements at each electrode to differ. Unlike background signals caused by extraneous external objects, such background non-uniformities are not expected to change over time. A local offset calibration or adaptive thresholding scheme can cancel these fixed sensor disparities. Once these sensor offsets are taken into account and electrical noise is filtered, the proximity image can simply be thresholded to identify regions of fleshy contact. Note that single-finger projective touchpads do utilize offset adaptation but do not have to segment the image into fleshy contact regions; they simply compute a global centroid from measurements of all row and column electrodes.

2.2.2.1 Binary Tree Scanning

Lee's binary tree scanning algorithm [88] combines noise filtering and thresholding in hardware by analog grouping and summation of electrode capacitance measurements. The array is recursively subdivided into rectangular electrode groups of decreasing size via bisection starting with the whole array. Thresholds are calibrated during device initialization for each electrode group at each size, or level, in the recursion. During subsequent scanning, subrectangles are scanned only if the parent

rectangle's threshold is exceeded. Once the recursion reaches a measurement which passes threshold at the single electrode level, a finger position is computed as the centroid of the recursed electrode capacitance and its eight neighboring electrode capacitances. Advantages of Lee's scheme are: not every electrode in the array need be separately scanned each pass, and grouping of many electrodes at the beginning of the scan tends to average out noise. The disadvantage is that small, light contacts can be lost among the large electrode groups if the large group thresholds are marginally too high.

2.2.2.2 Brute Array Scanning

Both digital and analog processing speeds have increased enough since Lee's prototype was built that the scanning overhead concerns have become negligible, especially in light of the additional finger tracking and gesture recognition algorithms which the MTS must execute. Keep in mind that though the number of discrete components necessary for an electrode array may make it seem large, the number of "pixels" is still small compared to even a low-resolution digital camera image. For this reason, and to ensure even brief, light finger contacts are captured, the MTS employs a brute force electrode scan to form a complete proximity image before applying standard digital filtering techniques.

2.2.2.3 Sensor Offset Adaptation

Sensor offset calibration will fail during device initialization if the user's hands are already on the board. Since there may not be a time when the fingers are known to be absent, the MTS continuously updates each electrode offset with the minimum of readings from that electrode. Suppose $A_{ij}[n]$ is the raw tactile proximity measured from the electrode at row i, column j during scan cycle n. Then the local offsets O_{ij} can be updated as:

$$O_{ij}[n] = \min(A_{ij}[n], O_{ij}[n-1]) \tag{2.1}$$

The offset-corrected image E is then:

$$E_{ij}[n] = A_{ij}[n] - O_{ij}[n] \; \forall i, j : \; 0 <= i < E_{rows}, 0 <= j < E_{columns} \tag{2.2}$$

Since capacitance measurements always return to baseline when fingers are removed, the offsets will correct themselves by decreasing as soon as fingers are lifted. The danger of this method is that negative electrical noise spikes can cause inadvertent lowering of the offsets. Local offsets which are too low lead to false positive proximity indications, just as offsets which are too high cause finger contacts to be missed. The MTS compromises by decreasing offsets only when at least three low proximities are read consecutively and by allowing very slow recovery, over about a minute, should an offset get lowered too far:

$$O_{ij}[n] = \min(\max(A_{ij}[n], A_{ij}[n-1], A_{ij}[n-2]), (O_{ij}[n-1] + \beta)) \tag{2.3}$$

where the max operation provides immunity to single negative noise spikes and a tiny β gives a slow recovery rate. Even with a tiny β, hands which are left resting on the board a few minutes will appear to fade. To prevent this, β is further decreased for those electrodes which the system confidently identifies as underlying a fleshy contact. These offsets quickly adapt to the minimum baseline capacitance so any readings above the offsets can be modeled as the flesh proximity magnitude plus minor Gaussian background noise.

2.2.2.4 Proximity Image Filtering

While Lee [88] electrically averaged the capacitances of entire rectangular groups of electrodes to combat noise before threshold testing, the MTS electrode array is much less noisy than Lee's device. Furthermore, to take full advantage of the electrode array resolution, groups should conform to finger contact shape electrode by electrode rather than be constrained to rectangular groups which poorly fit the oval shape of most hand contacts. Therefore, the MTS only employs slight spatial

diffusion of each offset-corrected image to combat electrical noise. Then it applies significance threshold and local maximum tests to each diffused pixel to detect the center of each hand contact, as further described in Chapter 3.

2.3 Topology of Hand Proximity Images

To illustrate typical properties of hand contacts as they appear in proximity images, Figures 2.7–2.10 contain sample images captured by the prototype array of parallelogram-shaped electrodes. Shading of each electrode darkens to indicate heightened proximity signals as flesh gets closer to the surface, compresses against the surface due to hand pressure, and overlaps the parallelogram more completely. Notice that the proximity images are totally uncluttered by background objects; unlike optical images, only conductive objects within a couple millimeters of the surface show up at all. Background sensor offsets have already been removed from each image, and background electrical noise levels are so low as to not be visible with the given grayscale intensity map. Certain applications such as handwriting recognition will clearly require finer electrode arrays than indicated by the electrode size in these sample images. In the discussion that follows, the proximity data measured at one electrode during a particular scan cycle constitutes one "pixel" of the proximity image captured in that scan cycle.

In this section and the rest of this dissertation, the term "proximity" will only be used in reference to the distance or pressure between a hand part and the surface, not in reference to the distance between adjacent fingers. "Horizontal" and "vertical" refer to x and y directional axes within the surface plane. Proximity measurements are then interpreted as pressure in a z axis normal to the surface. The direction "inner" means toward the thumb of a given hand, and the direction "outer" means towards the pinky finger of a given hand. For the purposes of this description, the thumb is considered a finger unless otherwise noted, but it does not count as a fingertip. "Contact" is used as a general term for a hand part when it

touches the surface and appears in the current proximity image, and for the group and path data structures which will represent it in Chapter 3.

2.3.1 Flattened Hand Image Properties

Figure 2.7 shows a right hand flattened against the surface with fingers outstretched. This flattened hand image includes all of the hand parts which can touch the surface from the bottom of one hand, but in many instances only a few of these parts will be touching the surface, and the fingertips may roam widely in relation to the palms as fingers are flexed and extended. At the far left is the oblong thumb which tends to slant at about 120°.

The columnar blobs arranged in an arc across the top of the image are the index finger, middle finger, ring finger and pinky finger. Since the fingers are fully extended, the creases at finger joints cause slight undulations in proximity along each column, though smearing by the parallelogram electrodes obscures this effect somewhat. Flesh from the proximal finger joints, or proximal phalanges, appears as the particularly intense undulations at the bottom of the index, middle, and ring finger columns. Since the fingers are fully flattened, flesh from the forepalm calluses is also visible as small clusters below the proximal phalanges, near the vertical level of the thumb.

The inner and outer palm heels cause the pair of very large contacts across the bottom of the image. These palm heels tend to be quite large, mildly oblong, and oriented diagonally. Unless the center of the palm is intentionally pushed against the surface, a large crease or proximity valley clearly separates the inner and outer palm heels. Even though image resolution is fairly low, it is clear that the fleshy contacts from different parts of the hand have subtly contrasting geometric properties. All the hand contacts are roughly oval-shaped, but they differ in pressure, size, orientation, eccentricity and spacing relative to one another.

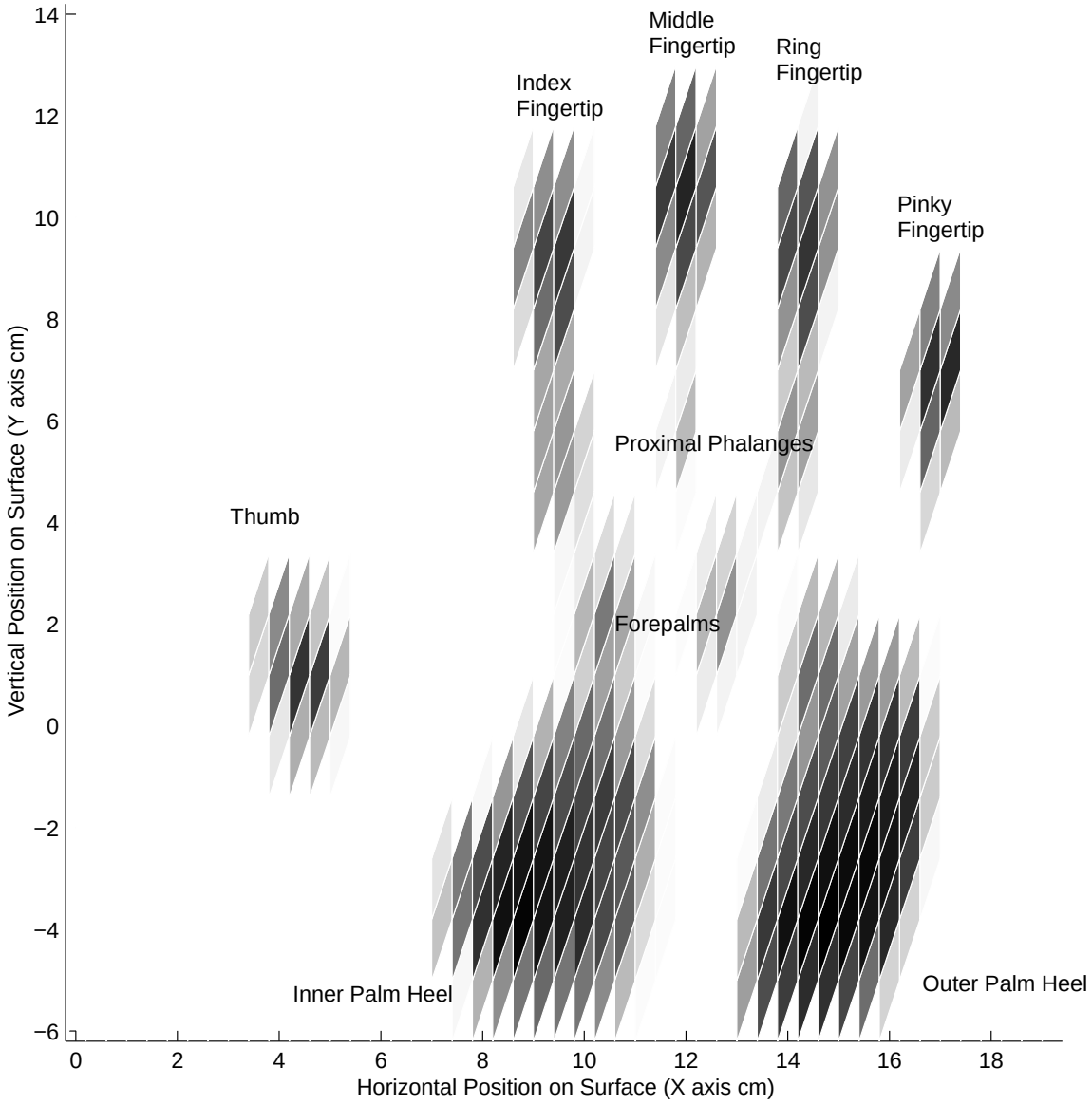


Figure 2.7: Offset-corrected proximity image of right hand flattened onto the surface with fingers outstretched and all hand parts labeled.

2.3.2 Properties of Hands in the Neutral Posture

Figure 2.8 shows a proximity image for all fingers and palms of both hands

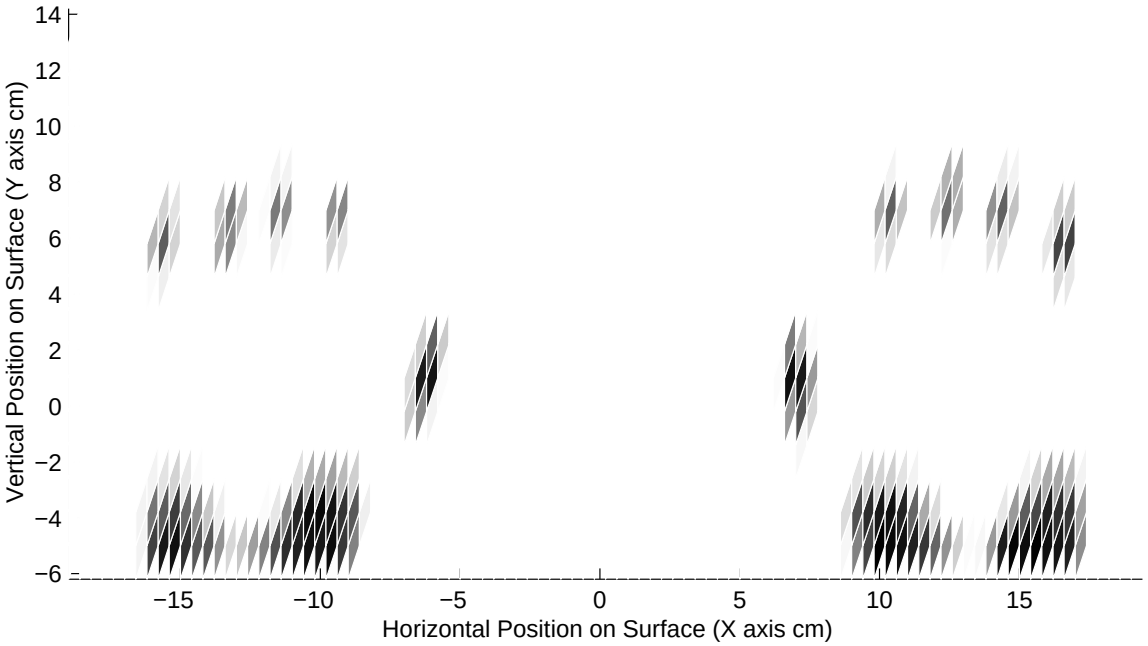


Figure 2.8: Proximity image of both hands resting on the surface in their respective neutral or default postures.

resting in what will be known hereafter as their default positions. Since these positions correspond to the most neutral hand and finger postures, with wrist straight and fingers curled so fingernails are normal to the surface, gestures are most likely to start from this hand configuration. Note that since fingers are curled, the proximal phalanges and forepalms are far above the surface and not visible. Because the fingers are slightly spread in this neutral posture, all fleshy contacts are clearly separated by at least one electrode at the background or zero proximity level. Since only the tips rather than the lengths of the fingers are visible, the fingers appear much shorter than in Figure 2.7, and would appear circular if not for vertical smearing by the parallelogram electrodes. However, the finger widths remain fairly constant

regardless of contact elongation. Also, the electrodes at the center of each fingertip do not appear as dark as the central thumb and palm heel electrodes because, in this case, the fingertips contacts are not tall enough to fully overlap any of the parallelograms, limiting the proximity signal regardless of their distance from the surface. The palm heels appear somewhat shorter than in Figure 2.7 since only the rear of the palm can touch the surface when fingers are flexed, but the separation between the palm heels is unchanged.

The fact that the intermediate finger joints connecting fingertips to palms, *i.e.*, the lengths of the fingers, do not appear in this commonly occurring proximity image has further consequences. While such lack of intermediate hand structure simplifies determination of the fingertip centroid, it is also the main shortcoming of capacitive proximity sensing in terms of hand gesture recognition. Reliably establishing finger or even hand identity when intervening hand structure is missing from the proximity images poses the most challenging problem of the work described in this dissertation. This challenge is the subject of Chapter 4.

2.3.3 Partially Closed Hand Image Properties

For a tracking system to support a wide range of hand gestures, it must tolerate contact shapes and juxtapositions which vary from the default. The two extremes to be considered in this work are the previously discussed flattened hand and the partially closed hand shown in Figure 2.9. Here the thumb is pushed directly behind the index finger, but vertical smearing by the wedge electrodes may cause thumb and index finger to appear as a single unseparable contact. Unlike the default hand posture in Figure 2.8, adjacent fingertips are so close together as to be distinguishable only by slight proximity valleys or saddle points between them. At the given horizontal electrode spacing, the saddle points between adjacent fingertips may only be separated by a single column wide. Any segmentation algorithm must use the partial minima in the horizontal direction to distinguish these fingertips. In

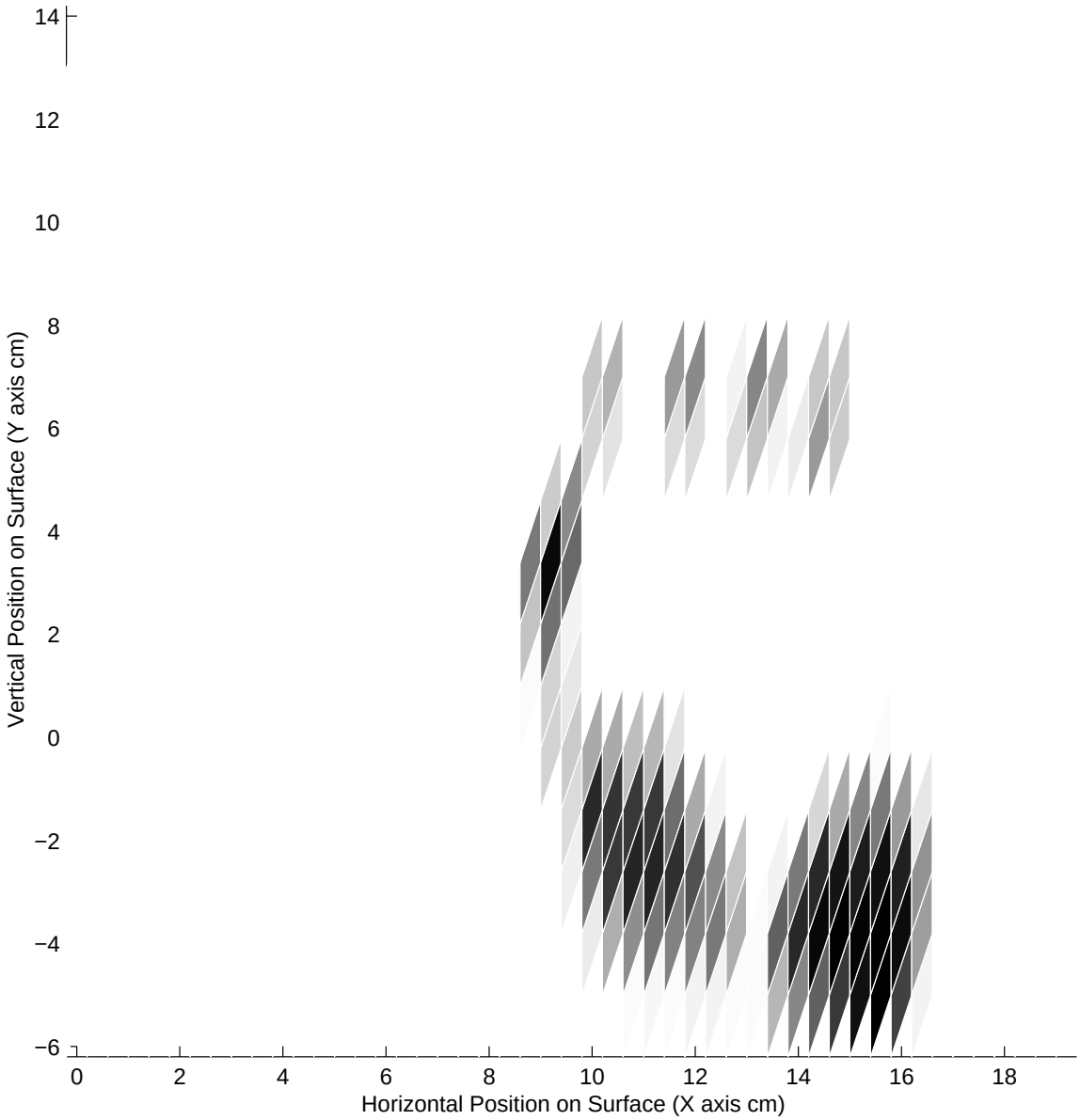


Figure 2.9: Proximity image of a partially closed hand with fingertips squished together.

case the fingertip row is rotated, partial minima in diagonal directions must also be detected. This conflicts with the segmentation needs of palms, which may contain spurious partial minima due to minor variations in sensor gain or flesh proximity across their large areas. All partial minima within palm contacts should be ignored except the large crease between the palm heels.

2.3.4 Pen Grip Image Properties

Figure 2.10 is a proximity image of a right hand in a pen grip configuration, which is particularly comfortable and dexterous for handwriting or freehand drawing. The thumb and index fingertip are pinched together as if they were holding a pen, but in this case they are touching the surface instead. Actually the thumb and index finger appear the same here as in Figure 2.9. However, the middle, ring, and pinky fingers are curled under as if making a fist, so the knuckles from the top of the fingers actually touch the surface instead of the finger tips. The curling under of the knuckles actually places them behind the pinched thumb and index fingertip, very close to the palm heels. The knuckles also appear larger than the curled fingertips of Figure 2.9 but the same size as the flattened fingertips in Figure 2.7. These differences in size and arrangement are sufficient to distinguish the pen grip configuration from the closed and flattened hand configurations. Though the contact segmentation and identification methods presented in this dissertation extend to the pen grip configuration with minimal modification, a higher resolution sensor array without vertically smearing parallelogram electrodes is needed to accurately discern the pinched fingers.

2.3.5 Comfortable Ranges of Hand Motion

Given that the MTS prototype has the form factor of a standard computer keyboard and is similarly placed on a desk, lap or workbench to operate from a sitting or standing posture, the ranges of hand position and rotation expected during

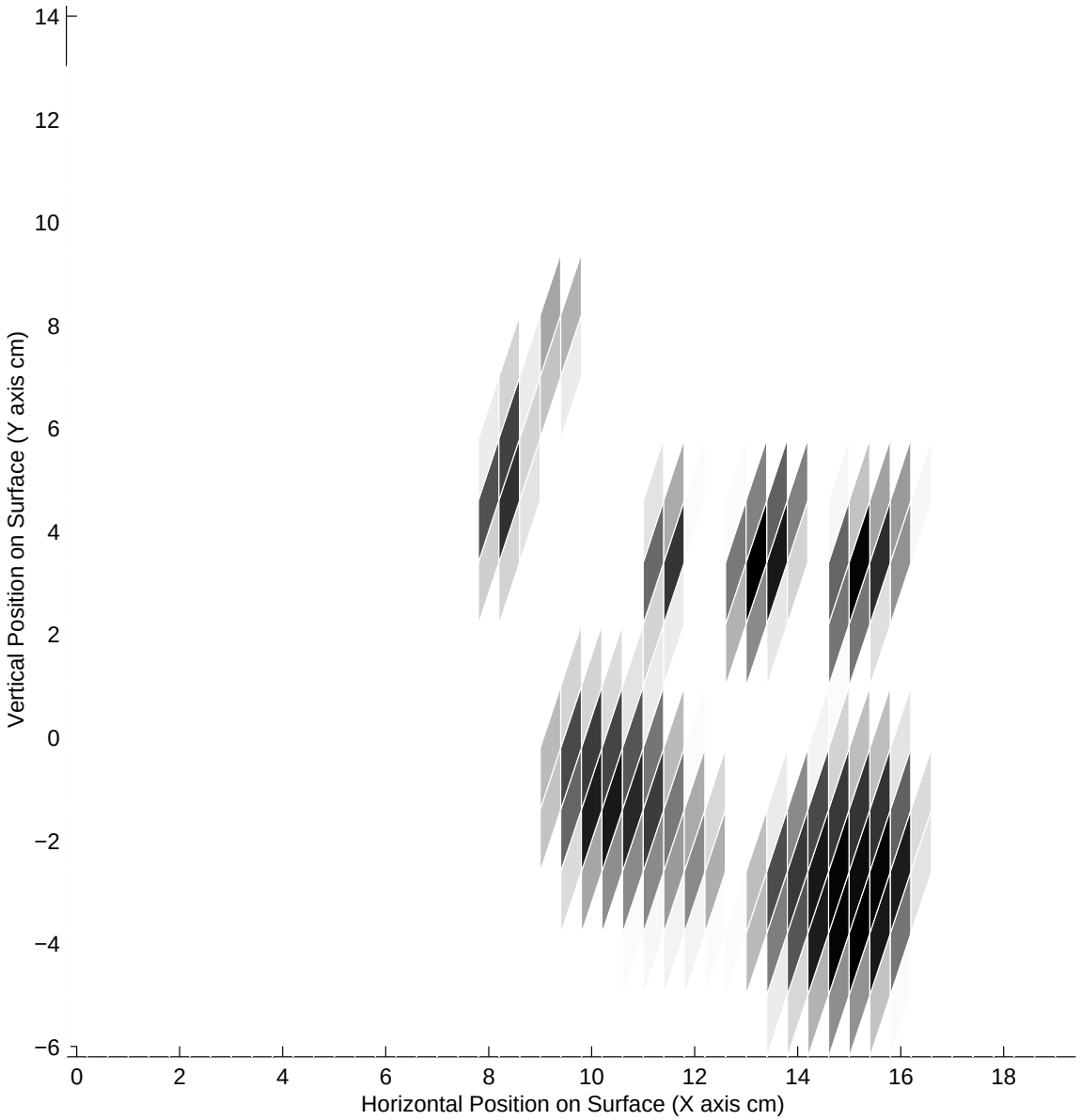


Figure 2.10: Proximity image of a hand with inner fingers pinched and outer fingers curled under towards the palm heels as if gripping a pen.

normal operation are fairly limited. When only one hand is on the surface, its maximum inward rotation can occur when it crosses to the opposite side of the surface, as shown in Figure 2.11. This situation maximizes the inward rotation of both the forearm about the elbow and the hand about the wrist. The maximum

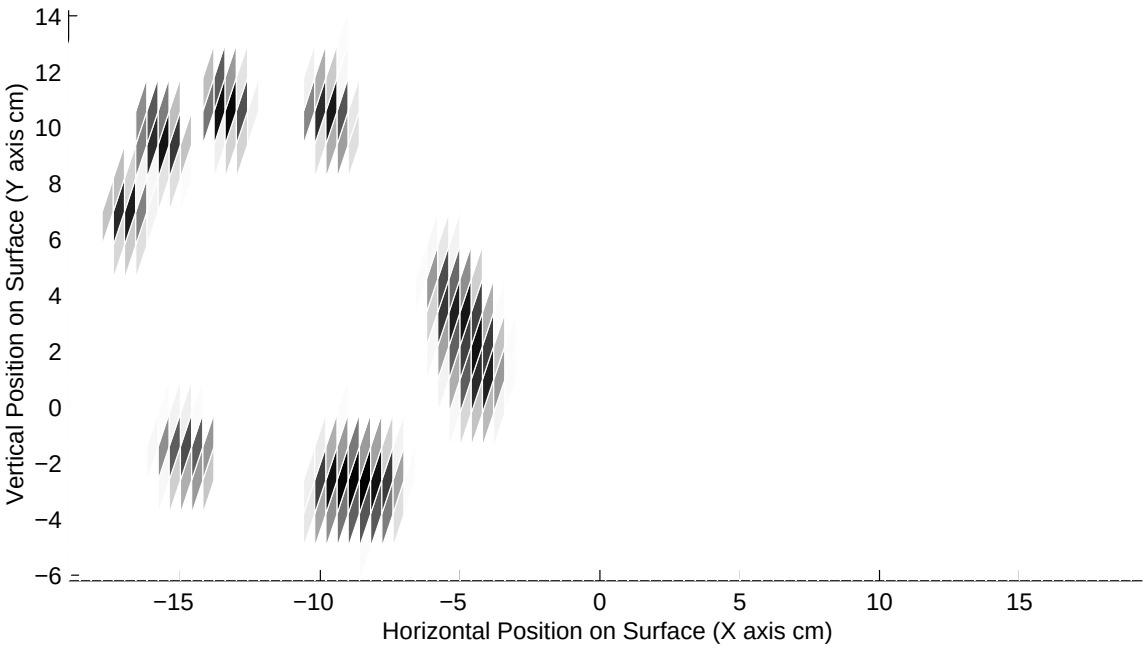


Figure 2.11: Proximity image of right hand at far left of sensing surface and rotated counter-clockwise to its biomechanical limit.

clockwise or outward rotation occurs from the default hand position with forearm parallel to the vertical surface axis, as shown for the right hand in Figure 2.12. Further rotations are only possible through contortions of the whole body or if the operator's torso is not facing the apparatus.

When both hands are on the surface, hand position is even further limited by the fact that operators are not expected to let the hands cross over or overlap one another. Figure 2.13 shows the maximum leftward position of the right hand when the left hand is in its default position. For some operations only part of a hand may remain in the active sensing area, as shown for the row of right hand fingertips at

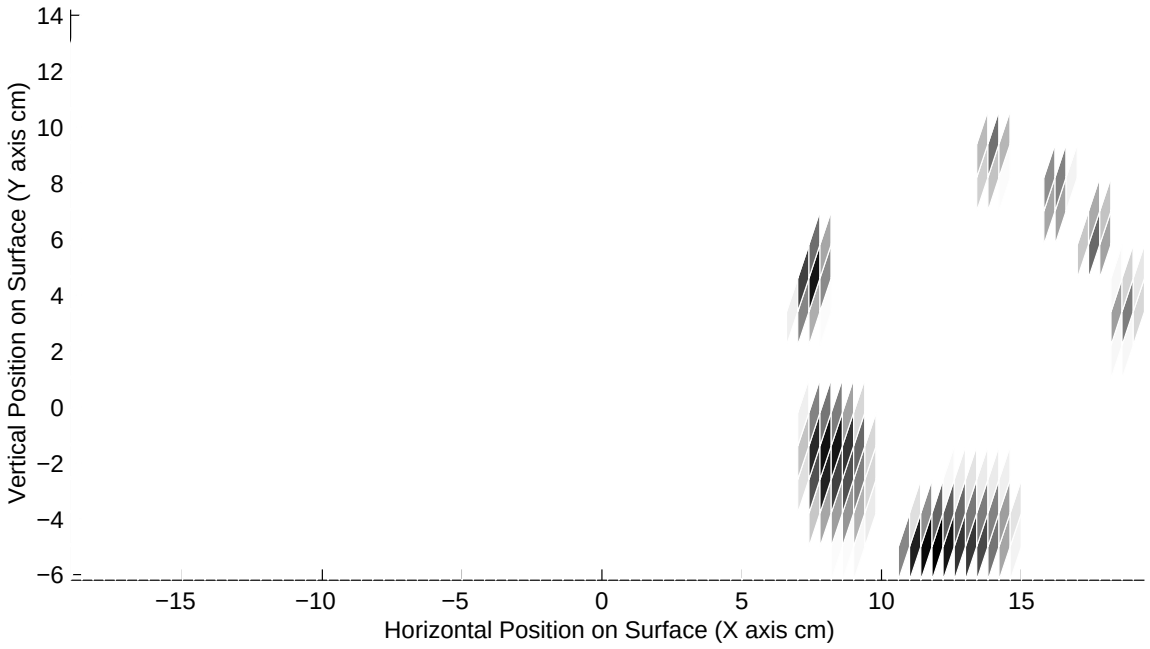


Figure 2.12: Proximity image of right hand at far right of sensing surface and rotated outward to its biomechanical limit.

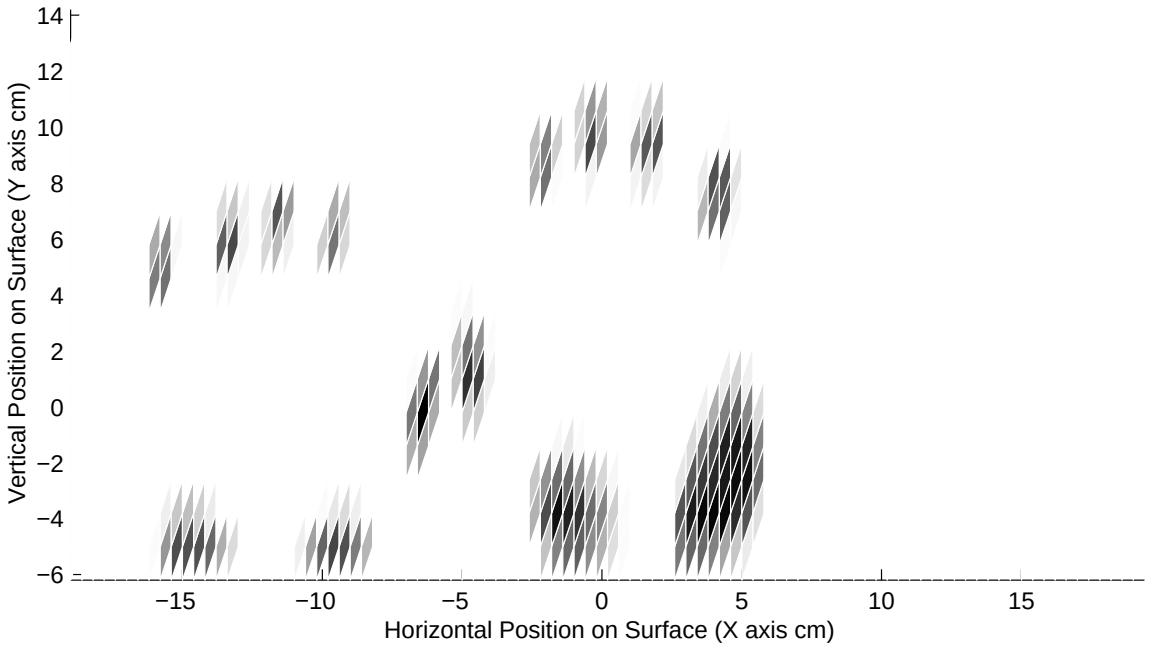


Figure 2.13: Proximity image of left hand in default position and right hand up against it.

the bottom middle of the surface in Figure 2.14. Though it is hard to imagine how

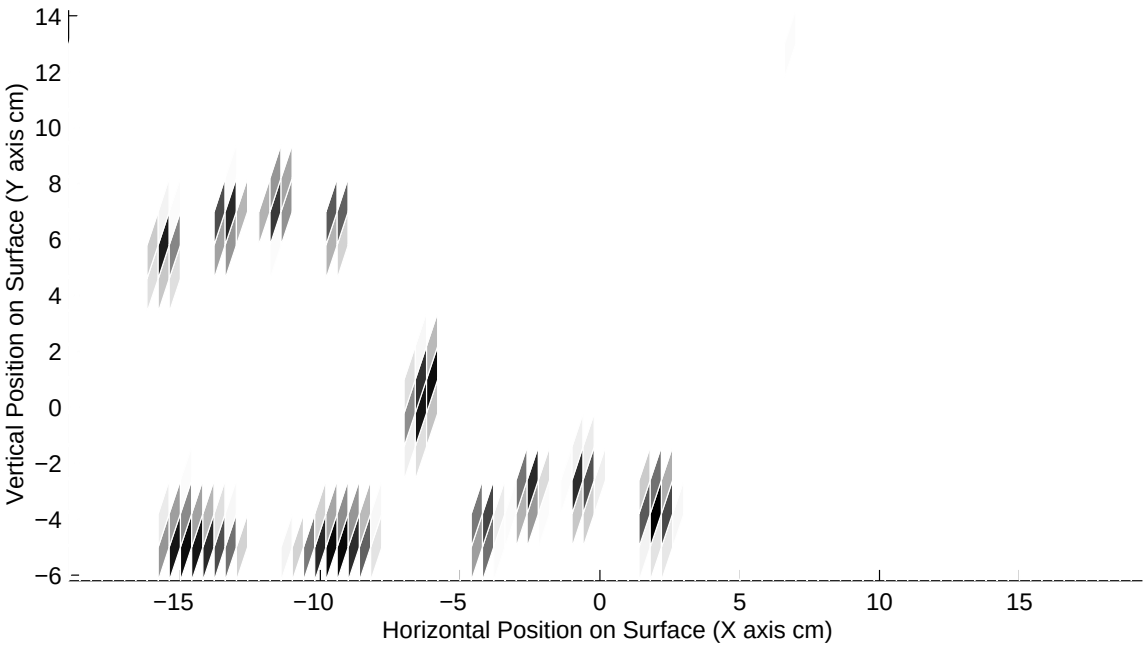


Figure 2.14: Proximity image of left hand in default position and right hand moved down so only fingertips remain in active sensing area.

this would be useful, the fingertips can also lie over the top of the active sensing area as in Figure 2.15, so only the thumb and palms remain visible.

2.4 Conclusion

Capacitance-based proximity sensing has many advantages over other hand motion sensing techniques. These advantages include precise detection of flesh contact with a surface, zero-force activation, avoidance of mechanical encumbrances, prevention of fingertip occlusion, and absence of background scene clutter. An array of a few thousand electrodes is sufficient to detect and uniquely determine the positions of any number of contacts from the undersides of both hands. Though each electrode has a constant sensor offset which must be removed, a large MTS can have signal-to-noise ratios as high as its tiny touchpad cousins.

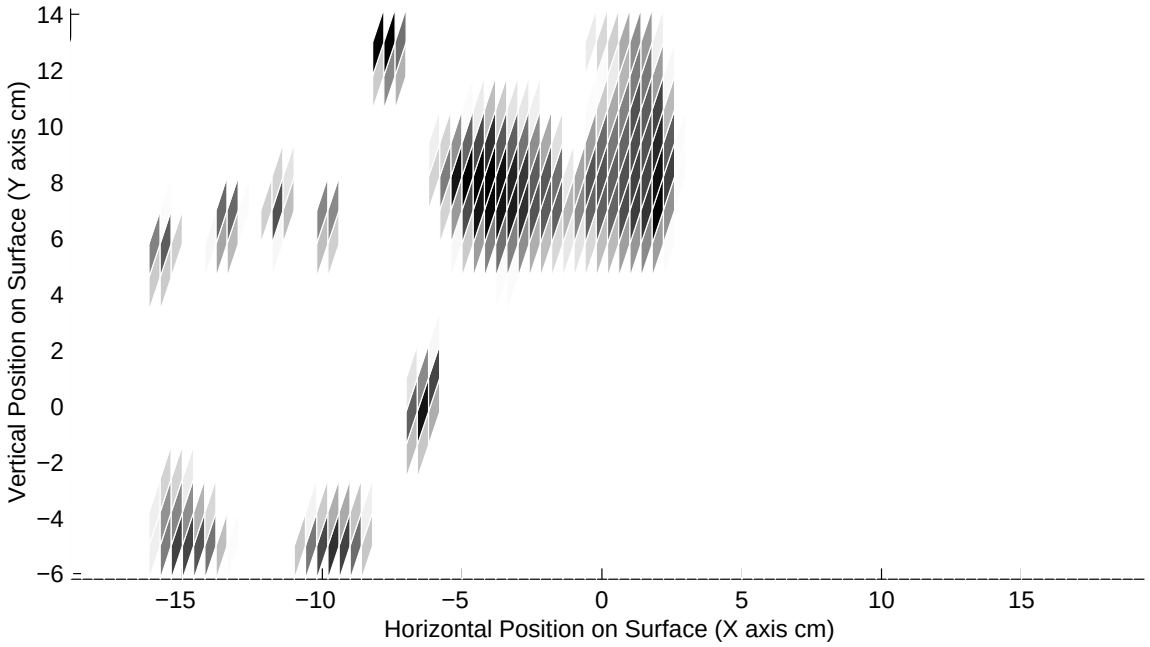


Figure 2.15: Proximity image of left hand in default position and right hand moved up so only thumb and palms remain in active sensing area.

The MTS offers a previously unexplored compromise between the rich tactile and force feedback of a mechanical keyboard or joystick and the feedback void of free space hand gestures. The proximity signals measured by the MTS correspond almost exactly to the operator's own sensations of engaging and sliding the hand across the surface. Even though hand proximity images contain ambiguities due to the lack of sharp edges between flesh contacts and the absence of intervening hand structure, the results of Chapters 3 and 4 will show that these ambiguities are surmountable. Ultimately such a unique, close correspondence between the sensations of the operator and the proximity imaging system can support much faster and more accurate gesture recognition than video-based systems.

Chapter 3

HAND CONTACT SEGMENTATION AND PATH TRACKING

This chapter and the following chapter describe the finger tracking and identification system of the MTS. In order to support resting or simultaneous chordic manipulation by both hands, the tracking system must be quite sophisticated. As the system diagram in Figure 3.1 shows, the overall system architecture resembles that of most computer vision systems, including stages such as image segmentation, feature extraction, path tracking across images, and object recognition, or in this case contact identification. However, the algorithms presented here have been carefully chosen and adapted to meet the unique anatomical, biomechanical, and topological constraints available at each stage. The modules are highly interdependent: the accuracy of path tracking and identification hinges upon consistent, carefully defined segmentations, and ambiguities which cannot be resolved within a module can often be bootstrapped with the help of feedback from other modules. As the feedback paths in Figure 3.1 indicate, the hand position estimates described in Section 4.3 of the following chapter constitute the primary feedback between modules. Thus while this chapter will concentrate on the contact segmentation and path tracking modules, the functioning of the system as a whole can only be understood by study of both chapters.

The segmentation module is responsible for breaking up each proximity image into groups of electrodes underlying identifiable portions of hand flesh. It then

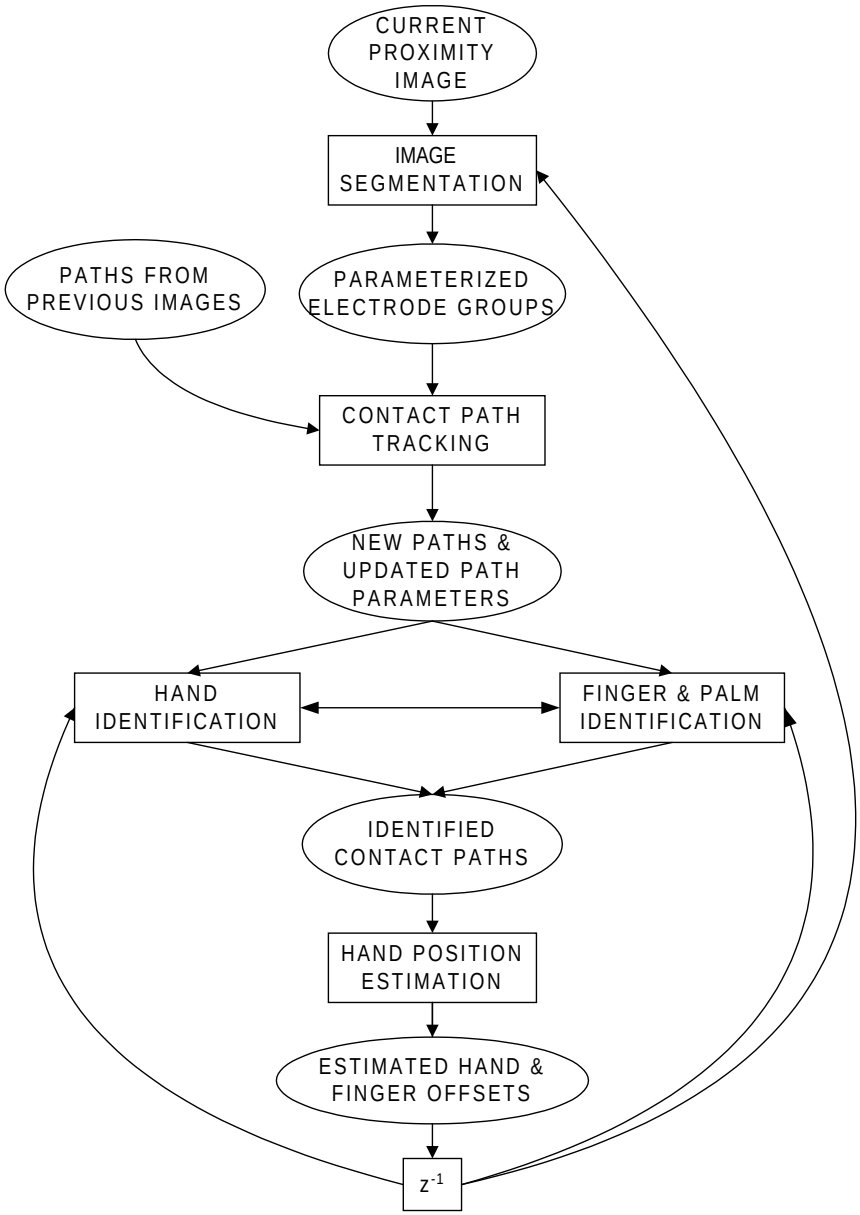


Figure 3.1: System-level diagram for hand and finger tracking and identification modules.

extracts geometric features from each group. The path tracking module links into trajectories the groups from successive proximity images which correspond to the same hand contact. It then computes the velocity along each hand contact trajectory. Though path tracking is a well-known problem and not terribly difficult for finger movements imaged at reasonable frame rates, the path tracking module has the important responsibility of detecting when individual fingers touch down on or lift off the surface. The hand identification module decides which hand causes each surface contact, and the finger identification module assigns a unique thumb, fingertip, or palm heel identity to each contact within a hand. Finally, the hand position estimator maintains a conservative estimate of overall hand and finger position offsets relative to the default finger positions of Figure 2.8.

3.1 Notation and Major Variable Types

The mathematical notation is designed to consistently describe both the conventional arrays which arise in low-level proximity image processing and the complex data types and instantiations needed to organize a variety of high-level hand and finger parameters. Global scaling constants and thresholds are specified in subscripts of K. There are four high-level object data types denoted by G for groups of electrodes, P for contact paths or trajectories, F for parts of the hand such as thumb, fingers, or palms, and H for a hand as a whole. Particular parameters of these structures are specified by names in subscripts, *e.g.*, G_x for a group's x centroid position. Parameters which are updated every scan cycle can include a bracketed time index, such as $P_x[n], P_y[n]$ for the time-filtered path centroid at scan cycle n. Particular group or path object instantiations are distinguished by a numerical index immediately following the capital letter; e.g., $P1_{vx}[n] < P2_{vx}[n]$ compares the horizontal velocities of two distinct paths. The index 0, *e.g.* $G0, P0, F0$ is reserved for a null or default object instantiation with all measurement parameters set to zero and feature parameters at their defaults of 0 or 1 as appropriate.

3.2 Contact Segmentation

The objective of contact segmentation is to group those electrodes or pixels which underly the same distinguishable hand part, *i.e.*, the same finger or palm heel, and then to represent each group to higher levels of the tracking system via a compact parameterization. Though removing background objects from proximity images is trivial because none are expected, accurately segmenting fleshy contacts which are variously shaped and separated from one another remains challenging.

3.2.1 Introduction to the Contact Segmentation Problem

For proper functioning of the hand and finger identification methods of Chapter 4, it is imperative that segmentations be flawless for the entire range of hand configurations which operators are expected to perform. The combinatorial optimization algorithms of the finger identification process assume that the correct mapping between segmentation groups and hand part identities is a one-to-one mapping, and the algorithms attempt to find the optimal one-to-one mapping. If the segmentation process erroneously merges the contacts of two distinguishable hand parts or splits the contact from one hand part into multiple groups, the correct identity mapping will no longer be a one-to-one mapping, and at least some of the contact identifications will be wrong. Designing the identification process to be highly tolerant of erroneous merging or splitting of groups would weaken the constraints on the combinatorial optimization and add undue complexity to identification. Luckily, this is not necessary. The segmentation techniques presented here will demonstrate that with sufficient tuning of segmentation rules, hand contact segmentation can be nearly flawless, even for low-resolution parallelogram electrode arrays. The failures that do occur are usually mild enough that feedback from partially correct identifications is sufficient to adjust the segmentation rules, producing correct segmentations for subsequent images.

Since the electrodes at the center of each hand contact invariably possess locally maximum proximity, an efficient approach to proximity image segmentation is to search outward from local maximum electrodes for contact edges, marking the edges as group boundary pixels. Algorithms which start at a seed pixel and search outward for similar pixels are generally known in computer vision as *region growing* segmentation methods [13, 68, 70, 172]. Local maxima are the obvious candidates for seed pixels. The primary challenge of proximity image segmentation is then to decide exactly what qualifies as an edge between contacts. For reliable segmentation of a wide range of hand configurations, this decision must depend on some sort of contextual feedback about the types of contacts being segmented.

Even if edge detection decisions are complex, this region growing process incurs surprisingly little computational overhead because searching outward from local maxima confines the edge detection computations to relatively small portions of the image containing about a dozen pixels per contact. Furthermore, since edges of convex contacts tend to be oriented perpendicularly to the vector passing from an edge pixel to the center of the contact, partial spatial derivatives only need be computed along the direction of search relative to the contact center, not in all directions. Thus while the MTS segmentation methods are related to watershed merging methods [42, 57, 151], the MTS does not have to compute the gradient of the whole image, and its merging is simply based upon region overlap. Likewise, proximity image segmentation is not nearly as computationally expensive as prior segmentation methods for optical images of hands. These prior methods compute complicated quantities such as spatiotemporal derivatives for optical flow [121], color histograms [3], or Gabor filters [115, 146] across most of each image.

3.2.2 Overview of the Segmentation Process

The data flow diagram in Figure 3.2 provides an overview of the image segmentation process. To avoid spurious extrema, a spatially smoothed copy of

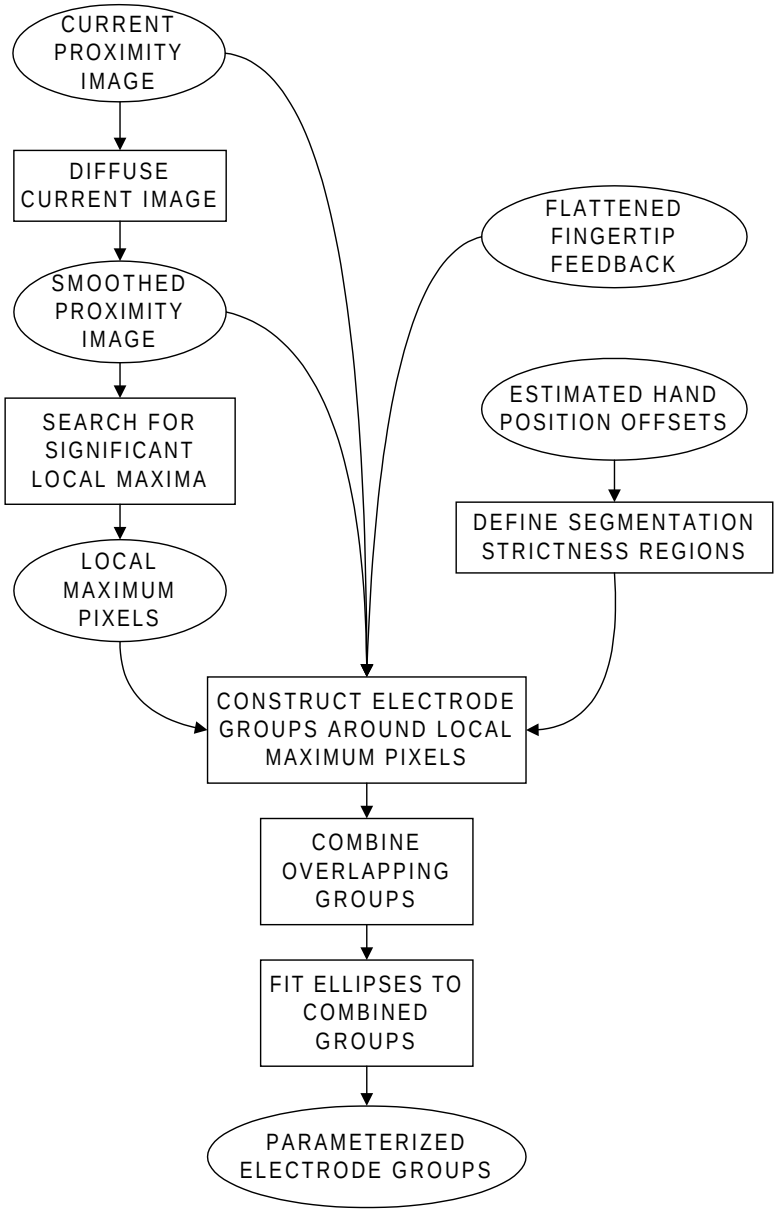


Figure 3.2: Data flow diagram of the proximity image segmentation process.

the current proximity image is utilized in the search for local maximum pixels and in computation of spatial derivatives. Any pixel whose smoothed proximity exceeds a significance threshold and the smoothed proximities of all eight nearest neighbor pixels is considered a local maximum pixel. The system constructs around each significant local maximum a semi-convex group of neighboring electrodes G_E which corresponds to a unit fleshy contact. Edges are always indicated at pixels where the undiffused proximity falls to background levels; but in case fingertips are only separated by a slight proximity valley, a partial minimum or saddle point between adjacent fingers is also considered an edge. Large palm heel contacts, on the other hand, may exhibit partial minima due to minor non-uniformities in flesh proximity or electrode sensitivity across the contact, so it is not sufficient to have the same edge detection rules across the whole image.

To resolve the conflicting edge detection needs of different types of hand contacts, the segmentation system bootstraps off of high-level analyses of previous images. These high-level analyses are supplied in the form of hand position estimates and finger contact height measurements. Given a hand position estimate, the segmentation system applies strict edge detection rules in regions of the image where fingertips and thumb are expected to appear, but it applies sloppy edge detection rules in regions of the image where palms are expected to appear. This ensures that adjacent fingertips are not joined into a single group and that each palm heel is not broken into multiple groups. In case smoothing fails to produce just one local maximum in each palm heel or flattened finger, a merging stage combines overlapping groups which contain the local maxima of other groups.

The last stage of the segmentation process is to extract shape, size, and position parameters from each electrode group. If properly segmented, most groups will roughly resemble ovals, so each group can be meaningfully represented at higher system levels by a few fitted-ellipse parameters: group centroid (G_x, G_y), total

proximity signal G_z, orientation G_θ, and eccentricity G_ϵ.

3.2.3 Proximity Image Smoothing

After the MTS fully scans a new proximity image and subtracts off background sensor offsets, the segmentation process begins by making a smoothed copy of the image via one or more passes with a nearest neighbor diffusion operator, also known as a Gaussian kernel. One pass of the 2-D diffusion operator performs the following computation:

$$\begin{aligned} S_{ij}[n] &= (1 - 2D_x - 2D_y)E_{i,j}[n] \\ &\quad + D_y(E_{i-1,j}[n] + E_{i+1,j}[n]) \\ &\quad + D_x(E_{i,j-1}[n] + E_{i,j+1}[n]) \end{aligned} \tag{3.1}$$

where D_x and D_y are the diffusion rates in the x and y directions. For stability, the rates must satisfy $D_x + D_y <= .5$ [112]. The MTS actually sets $D_x + D_y <= .25$ so that the contribution of the four neighbors is never more than the contribution of the center electrode. Since the electrode row spacing $\delta y \approx 1.2$ cm is much more than the column spacing $\delta x \approx .4$ cm and the wedge electrodes already smear a lot between rows, the proportion of vertical to horizontal diffusion is set so $(D_y/D_x) \approx (\delta x/\delta y) \approx .33$.

The smoothed image $S_{ij}[n]$ is then searched for significant local maxima, *i.e.*, pixels whose proximities are greater than or equal to a global significance threshold, $K_{maxima\_significance}$, and the proximities of all eight nearest neighbor pixels. $K_{maxima\_significance}$ can either be fixed or adapted to the mean and standard deviation of the background electrical noise. The smoothing thus has two benefits: to reduce the chance that an isolated noise spike on a single electrode will result in a local maximum which exceeds the significance threshold, and to consolidate local maxima to about one per distinguishable fleshy contact. Because of the high signal-to-noise ratio of the proximity sensors, only slight smoothing is necessary, as

indicated in Figure 3.6. Too much smoothing obscures the proximity valleys between adjacent fingertips. Nevertheless, because smoothing is the only operation besides the search for local maxima which is applied to the whole image, it incurs more computational cost than all other stages of the segmentation process combined, taking 2-3ms on the 60 MHz TSM320C32 DSP.

3.2.4 Segmentation Strictness Regions

As hinted in Section 2.3.3, segmentation needs are not uniform across the entire image or across all hand configurations. Regions of the image containing fingertips require *strict segmentation*, *i.e.*, establishment of contact edges at all horizontal partial minima. This conflicts with the segmentation needs in palm regions. Palms should be segmented *sloppily*; in other words, only large creases between palm heels or electrodes with background-level proximity should register as palm contact edges. How can the system decide where segmentation should be *strict* versus *sloppy* when contextual features such as relative contact shape and position can only be measured after segmentation has completed? A cumbersome and computationally expensive approach would be to form multiple segmentation hypotheses and try to construct an evaluation procedure to pick the most likely segmentation.

A simpler but effective approach is to utilize contact juxtaposition constraints and expected hand position. The critical observation is that for comfortable ranges of hand rotation, the fingertips always lie above the palms and usually appear above the thumb. If fingers, palm and thumb were all guaranteed to be touching the surface, the system could assume the local proximity maximum nearest the top of the board was a fingertip and the one nearest the bottom was a palm. A horizontal dividing line could be envisioned in between, above which strict fingertip segmentation rules would apply and below which sloppy palm segmentation would apply. In practice the palms may not be touching the board, or in rare cases the palms may touchdown before the fingers, so the placement of this dividing line has to be

guessed at. Since strict segmentation should also apply to the thumbs to prevent thumb contacts from merging with fingertips or palms, the sloppy regions should be further confined by vertical dividers to exclude regions where thumbs are expected. In case the hands are not positioned symmetrically, the sloppy region dividers should be placed independently for each hand.

Position estimates are available as top-level feedback from the hand position estimation module (Figure 3.1) in the form of offsets from the default hand positions (Figure 2.8). Section 4.3 describes in detail how the hand position estimation module derives conservative estimates using contact positions and identities from previous images. Basically each hand position estimate is initialized to the default hand position, tracks the average positions of identified finger contacts relative to their default finger positions, and decays back toward the default hand position when all fingers lift off the surface.

Figure 3.3 illustrates how the segmentation regions translate with the estimated hand offsets. In Figure 3.3a the hands are in their default positions, zeroing the estimated offsets for both hands. Plus signs in the diagram indicate the estimated position of each finger and palm heel in each hand. Rectangular outlines in the lower corners represent the left and right *sloppy segmentation regions*, where partial minima are largely ignored. The T-shaped region remaining is the *strict segmentation region*, where proximity saddle points must serve as contact edges. The best positions of the vertical dividing lines has been found empirically to be just inside of the sensor array columns where the index fingers are expected to lie, and the horizontal dividing lines are placed near the estimated vertical positions of their respective thumbs. The outside edges of the surface form the outer and lower boundaries of the sloppy regions. Due to the decay in estimated offsets after hands leave the surface, the sloppy segmentation regions return to the positions shown in Figure 3.3a after the hands have stayed off the surface a few seconds, regardless of

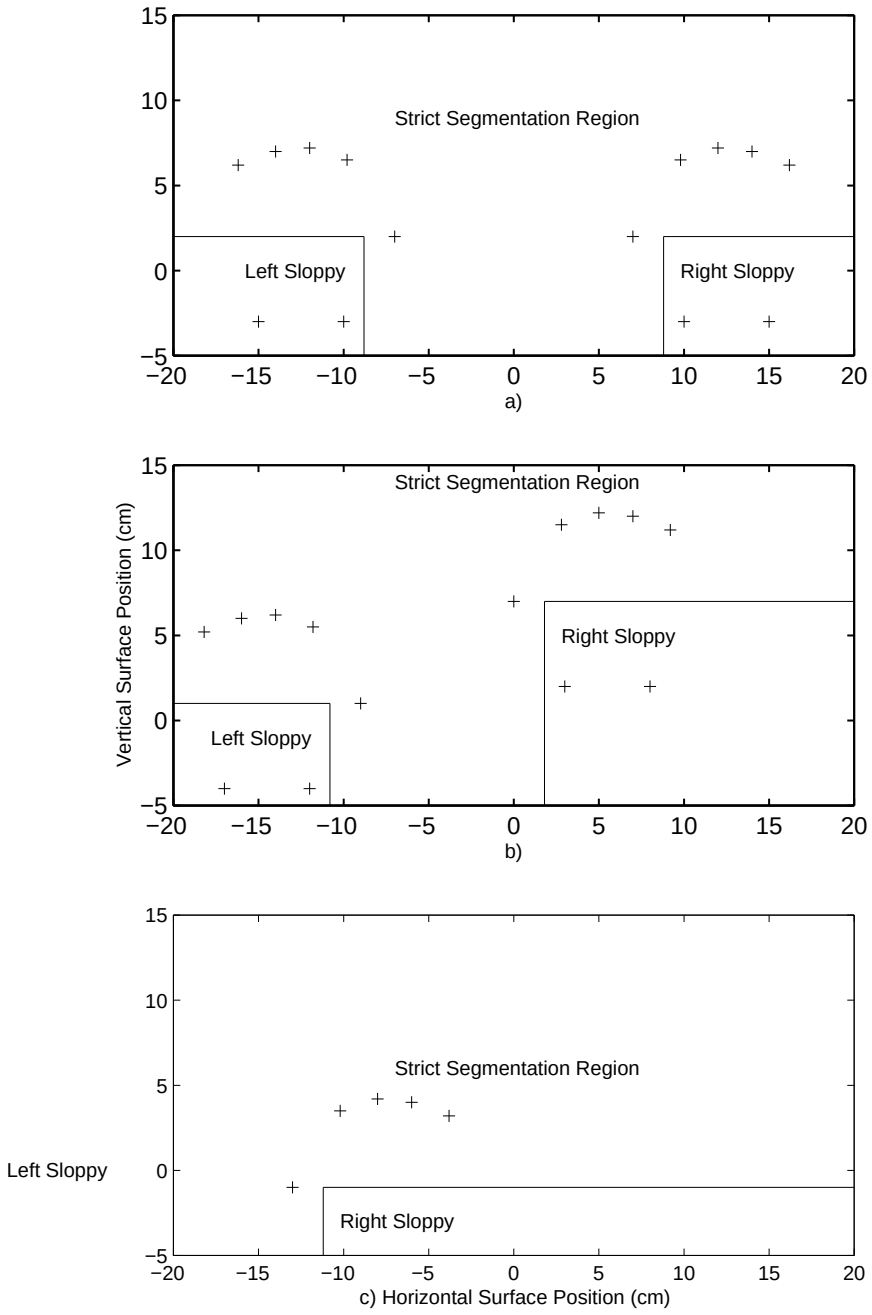


Figure 3.3: The positions of the left and right sloppy segmentation regions (boxes) in relation to estimated finger positions (plus signs) for a) both hands in their default positions or lifted off the board a long time, b) the hands in asymmetrical positions, and c) the right hand crossing to the left side of the board, pushing the left sloppy region out of the way since the left hand is off the board.

hand position at liftoff.

As the hands move away from the default positions, the dividing lines of the left sloppy region are translated by the left hand offset estimate ($LH_{eox}[n-1], LH_{eoy}[n-1]$), and the dividing lines of the right sloppy region are translated by the right hand offset estimate ($RH_{eox}[n-1], RH_{eoy}[n-1]$). Figure 3.3b shows how the sloppy regions follow the estimated hand positions as the right hand moves toward the upper left and the left hand moves toward the lower left. This ensures that the palms and only the palms fall in the sloppy regions as long as the hand position estimates are correct.

Figure 3.3c shows that the left sloppy region and hand position estimate move off the surface entirely when the left hand remains off the surface and the right hand slides to the left side of the surface. This prevents the fingers of one hand from entering the sloppy segmentation region of the opposite hand. This effect is implemented by imposing a minimum horizontal separation between the hand position estimates and, should the left and right regions get too close to one another, letting the hand with the most surface contacts override the estimated position of the hand with fewer contacts.

Though the hand position estimator is generally very good at keeping the sloppy regions under the palms as the hands moves around the board, errors are possible if parts of the hand unexpectedly touch down too far above or below their estimated positions. If only a palm heel touches down in the strict segmentation region at the top of the board, it can be segmented into multiple groups which will be misidentified as a row of overlapping fingertips. However, if the size and shape of these groups becomes so large as to be inconsistent with their identification as fingertips, the finger identification module will reidentify them as palms and eventually correct the hand position estimate. Likewise, if squished together fingertips touch down in the sloppy segmentation region at the bottom of the board, they

can be joined into one contact and misidentified as a palm heel. Section 3.2.9 will further examine the segmentation errors which can occur if the sloppy segmentation regions get out of place.

3.2.5 Segmentation Search Pattern

Once the local maxima pixels have been found and the sloppy segmentation regions defined, the segmentation process searches around each local maximum pixel for pixels which register significant proximity. Figure 3.4 depicts typical outward search patterns which start from a group's local maximum pixel and proceed in four directions testing for the contact boundary. For a given group G, let $G_{localmax_{row}}$ and $G_{localmax_{col}}$ be the row and column indices of group G's local maximum. Searching for contact boundaries starts at these indices (filled circles in Figure 3.4) and initially proceeds along the $G_{localmax_{row}}$ row toward the right and left until one of the applicable horizontal or diagonal boundary tests comes up positive. Each electrode encountered before reaching the boundary is added to the local maximum's group. While searching to the right, the additional indices $j_{prev} = j - 1$ and $j_{next} = j + 1$ are maintained for use referring to neighboring electrodes in horizontal minimum tests. When searching toward the left, $j_{prev} = j + 1$ and $j_{next} = j - 1$. The indices of the columns just inside the left and right columns where boundary tests turn positive are stored in $G_{firstcol[i]}$ and $G_{lastcol[i]}$, where in this case $i = G_{localmax_{row}}$. The electrode within the row with maximum smoothed reading,

$$G_{maxcol[i]} = \underset{G_{firstcol[i]} <= j <= G_{lastcol[i]}}{\operatorname{argmax}} S_{ij} \tag{3.2}$$

is also recorded. In this case $G_{maxcol[i]}$ should equal $G_{localmax_{col}}$, but when searching subsequent rows where $i \neq G_{localmax_{row}}$ the column $G_{maxcol[i]}$ may differ from the column of the local maximum.

Next the search automatically advances to the rows $i = G_{localmax_{row}} + 1$ and $i = G_{localmax_{row}} - 1$ directly above and below $G_{localmax_{row}}$, so electrodes from at

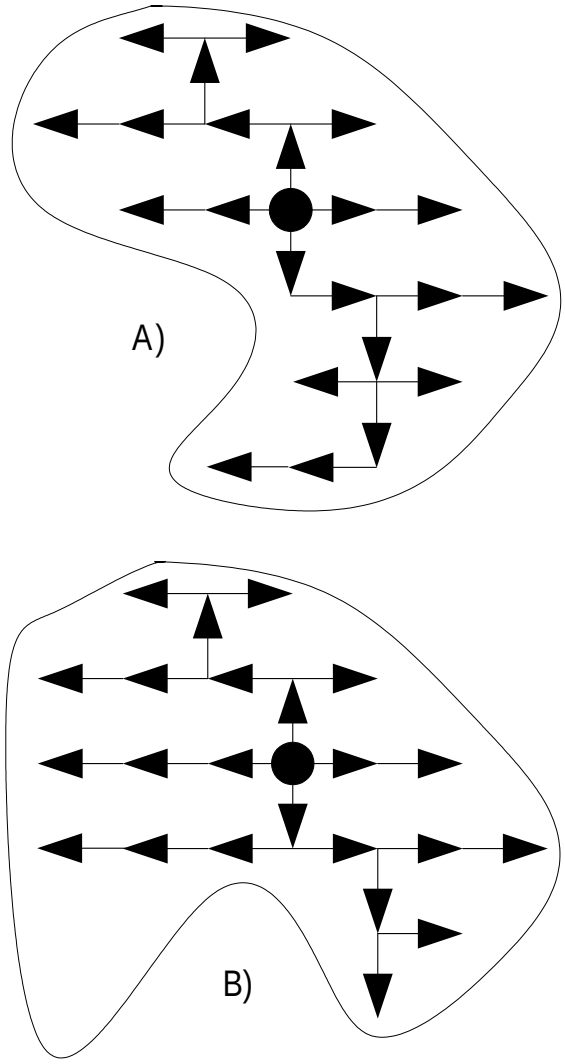


Figure 3.4: Typical search patterns starting at the group's local maximum (filled circle) and proceeding along successive rows towards the contact edge, represented here as the curved, closed boundary. Each arrowhead represents a pixel which is added to the group as the search encounters it. Notice that the search does not proceed outside the boundary. Because the search can only advance into the next row from the pixel with maximum proximity in the current row, it will advance past concavities into multiple horizontal offshoots a) but can only find one of the vertical offshoots b).

least three rows will be available in each group for vertical centroid interpolation. Before advancing rows the index of the previously searched row must be recorded as i_{prev} to aid in vertical and diagonal partial minimum tests. For these rows directly above or below the row of the local maximum, $i_{prev} = G_{localmax_{row}}$. The index of future rows to be searched i_{next} may also be needed, and will in this case be either $G_{localmax_{row}} + 2$ or $G_{localmax_{row}} - 2$. The horizontal search within subsequent rows always starts at the *previous* row's maximum column, *i.e.*, $j = G_{maxcol[i_{prev}]}$, and proceeds horizontally outward in both directions as it does for the row of the local maximum. Thus subsequent rows tend to be entered along the vertical or diagonal ridge of the contact. Advancement of the search to additional rows occurs recursively as long as vertical boundary tests are negative. These tests are only applied in the column $G_{maxcol[i]}$ to electrodes directly above or below. The rows immediately previous to those in which vertical boundary tests first come up positive are recorded as G_{toprow} and G_{botrow}.

Note that partial minima electrodes at the bottom of proximity valleys are never included in any group because it is not known in what proportion the two adjacent contacts contributed their signals. Assuming an equal, 50-50 split in contributions from each contact was found to cause more instabilities in contact centroids as the partial minima shift from one electrode to another than leaving partial minima electrodes out of both adjacent groups altogether.

Let e be an electrode of the electrode set $\mathcal{E}$ with row index e_i and column index e_j. When boundaries have been encountered in all directions, the resulting semi-convex group of electrodes can be described as:

$$G_E = \{e \in \mathcal{E} : G_{botrow} <= e_i <= G_{toprow}, G_{firstcol[e_i]} <= e_j <= G_{lastcol[e_i]}\} \quad (3.3)$$

The groups are called semi-convex because all of the electrodes within a group's row must be connected, *i.e.*, there can not be multiple runs of electrodes within a row

with excluded electrodes in between (Figure 3.4b). On the other hand, multiple, unconnected column segments are allowed within a group (Figure 3.4a) since $G_{firstcol[]}$ and $G_{lastcol[]}$ are arrays whose elements can differ across rows, whereas G_{botrow} and G_{toprow} are scalar. The semi-convexity constraint on the search pattern has minor consequences such as tolerating undulations in finger width. The semi-convexity constraint also discourages multiple vertical offshoots (such as fingers) of wide electrode clusters (such as palms) from being combined into one group. The key point is that since electrode groups need not be rectangular, they can closely fit the typical oval shape of flesh contacts without leaving electrodes out or including those from adjacent contacts.

3.2.6 Segmentation Boundary Tests

The contact edge tests consist of two rule classes: directional minimum tests which differ for sloppy versus strict segmentation regions and the proximity significance test which is applied the same to both strict and sloppy segmentation regions. Positive results on any applicable test cause a pixel to be marked as a group boundary and the search to resume in another direction, as indicated by the flow chart in Figure 3.5. Naturally, the edge of the electrode array always establishes a group boundary.

3.2.6.1 Proximity Significance Tests

The significance threshold test is true for an electrode at row i, column j if the unsmoothed pixel proximity is less than a significance threshold, $K_{pixel\_significance}$:

$$E_{ij}[n] < K_{pixel\_significance} \tag{3.4}$$

This generic pixel significance threshold is set to three or four standard deviations of the measured background noise, or about half of the local maxima significance

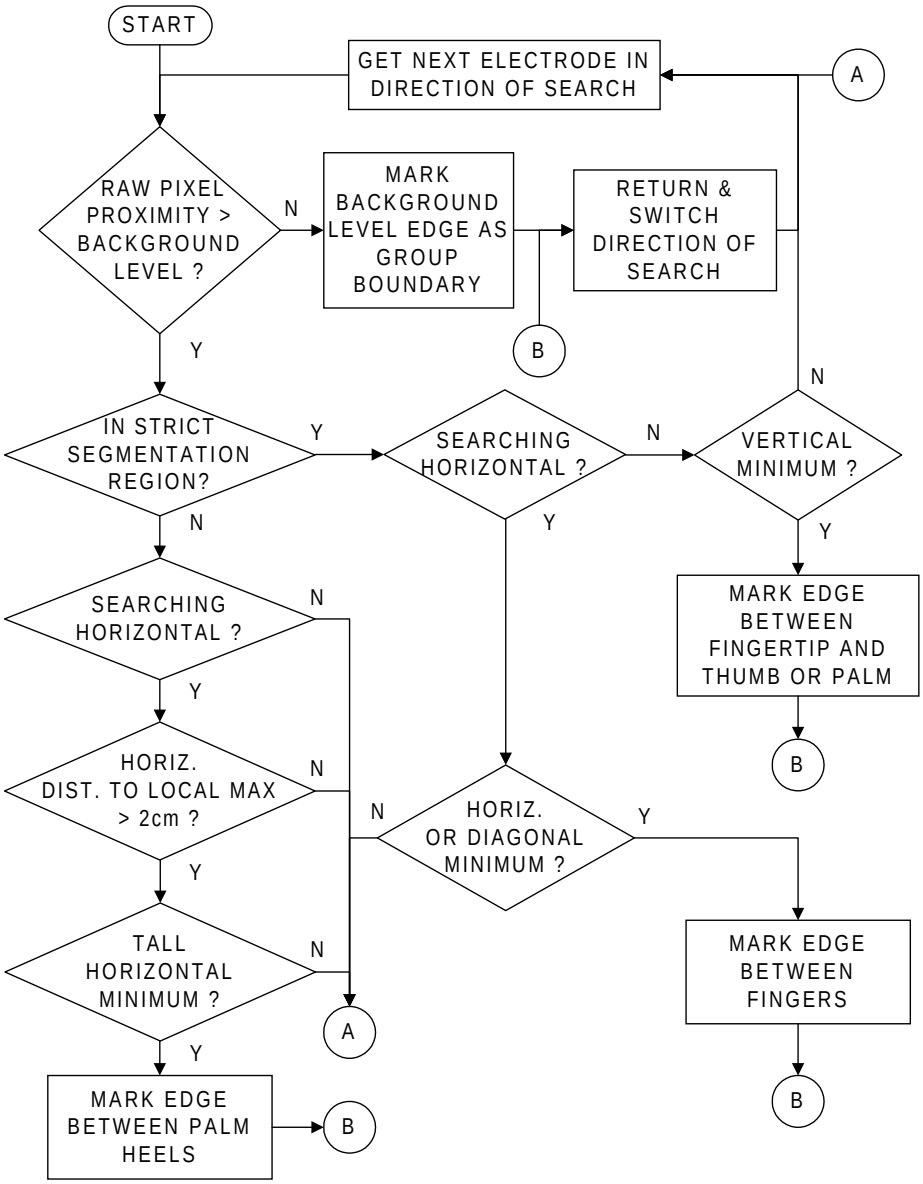


Figure 3.5: Flow chart summarizing the contact edge tests which are applied at each pixel encountered by a group's segmentation search pattern.

threshold $K_{maxima\_significance}$. Using the unsmoothed rather than smoothed proximity in this test ensures that diffusion of proximity outside of the true contact area (see Figure 3.6b) does not cause pixels outside the contact edge to be included in the group.

Using the unsmoothed proximity in the significance test also prevents merging of adjacent fingertips (Figure 2.9) which unexpectedly touch down in a sloppy segmentation region yet still have at least one electrode at background proximity level in the proximity valley between them. Though such valleys are not totally obscured by smoothing, diffusion from the contact centers causes the smoothed proximity of such electrodes to remain above any reasonably low significance threshold. Since sloppy region tests ignore the horizontal partial minima of such valleys, only this thresholding of the unsmoothed proximity image can keep fingertips which wander into sloppy segmentation regions reliably separated. If the adjacent fingertips can be kept segmented into separate groups for a few images, proper identification of them will eventually cause corrections in the hand position estimates to move the sloppy segmentation regions away from the fingertips.

3.2.6.2 Strict Segmentation Region Partial Minima Tests

Strict-segmentation-region partial-minima tests apply to horizontal, vertical and diagonal neighbors depending on the current direction of search. Since the search always proceeds away from a local maximum (Figure 3.4), a partial minimum or saddle point is generally indicated whenever proximity starts increasing in the direction of search. The tests have evolved empirically to detect diagonal minima and segment diagonally adjacent fingertips on the parallelogram electrode array. Therefore they may appear somewhat ad hoc; no doubt they could be simplified for a higher resolution array of square electrodes.

As a horizontal search begins in a row at the column of the previous row's maximum pixel $G_{maxcol[i_{prev}]}$, the current row's maximum pixel is expected to occur

in the same column or a column immediately adjacent to $G_{maxcol[i_{prev}]}$. For these columns j such that $G_{maxcol[i_{prev}]} - 1 <= j <= G_{maxcol[i_{prev}]} + 1$, a full partial minimum test of both next and previous pixels is applied instead of just checking for an increase in the next pixel in the direction of search, *i.e.*, a horizontal boundary is only indicated if $S_{i,j_{prev}}[n] > S_{i,j}[n] < S_{i,j_{next}}[n]$. This tolerates tall contacts which are oriented at a slant and therefore have proximity ridges which are not perfectly vertical. If $S_{i,j_{prev}}[n]$ was not checked and the search started from a column which did not have maximum proximity in the current row, diagonal ridges would be erroneously rejected.

For columns farther away from the previous row's maximum pixel, the horizontal test is slackened to include general increases in proximity in the horizontal, diagonal, and vertical directions of search. The horizontal boundary test simply becomes $S_{i,j}[n] < S_{i,j_{next}}[n]$, and a test $S_{i,j}[n] < S_{i_{next},j}[n]$ of the vertical neighbor in the next row to be searched indicates presence of a diagonal proximity valley caused by a contact diagonally adjacent to the contact being searched. Either test can stop the horizontal search in the current direction, establishing a horizontal boundary for the current row. By only testing that the proximity of the next pixel is greater than the current pixel without testing that the previous pixel is also greater than or equal to the current pixel, a search which starts in a sloppy region and meanders into a strict region will be forced to stop if proximities in the strict region are increasing in the direction of search, even if the actual saddle point occurred in the sloppy region and was ignored. This helps prevent palms from being joined to thumbs or flattened fingers if the sloppy segmentation regions get too close to the fingers.

When deciding whether to advance to the next row i_{next}, vertical partial minimum tests are applied at the column $j = G_{maxcol[i]}$ of the pixel in the current row found to have the maximum proximity. These tests also have a diagonal component, positively indicating a vertical boundary if $S_{i,j}[n] < S_{i_{next},j}[n]$, $S_{i,j}[n] < S_{i_{next},j-1}[n]$

or $S_{i,j}[n] < S_{i_{next},j+1}[n]$.

3.2.6.3 Flattened Finger Segmentation

To allow fingertip groups to merge reliably with the finger's proximal phalange contacts (Figure 2.7) without allowing thumbs, forepalms, or palm heels to merge with fingertips, the system ignores vertical partial minima in the strict segmentation region under a long series of conditions. First, such merging must be enabled by identification system feedback (Figure 3.2) indicating that most of the contacts identified as fingers are flattening onto the surface. This feedback takes the form of a flag which is set when in the previous image the geometric mean of the contact heights of all fingers but the innermost surpasses a threshold. This feedback distinguishes proximal phalanges of flattened fingers (Figure 2.7) from a thumb behind a curled fingertip (Figure 2.9) using the fact that it is biomechanically difficult to flatten one long finger without flattening the other long fingers. The geometric mean of multiple finger sizes and exclusion of the innermost finger ensures that phalange merging is not enabled by a misidentified, long thumb contact. If phalange merging were to be enabled without this multiple finger size requirement, any thumb contact traveling within a few centimeters behind a row of curled fingertips would merge with one of them.

Once the system has established from previous images that the fingers are at least beginning to flatten, several more measurements are made to ensure the fingertips are merged with only the proximal phalanges and not the forepalms or palm heels which could be flattening as well. First, the direction of search must be downward from the current group's local maximum. This helps to ensure that the search is coming from a fingertip, not the local maximum of the proximal phalange or forepalm. Additionally, the vertical location of the partial minimum being ignored must not be more than about 5 cm below the current group's local maximum, which corresponds to the maximum anatomical distance from the center of the distal

phalange (fingertip) to the center of the proximal phalange. The vertical location of the partial minimum must also be a couple centimeters above the horizontal dividing line of the hand's sloppy segmentation region, and only the first vertical partial minimum encountered will be ignored. These conditions all differentiate the proximal phalanges from the forepalms. Finally, the vertical partial minima should be in the same column or a column adjacent to the fingertip local maximum. If all of these conditions are met, search from a fingertip will continue past a vertical partial minimum to engulf a proximal phalange.

3.2.6.4 Contact Height Limitation Test

Despite all of the above precautions to prevent merging of fingers and palms, when the hand is flattened against the board very hard, images occasionally arise in which the only vertical minimum between a fingertip and palm heel is either in the sloppy segmentation region or between the fingertip and proximal phalange. In either case it will be ignored, merging a fingertip and palm heel into one group. The simplest way to prevent this is to observe that no hand part, finger, thumb or palm on the average adult hand is longer than about 8 cm. Therefore regardless of strict or sloppy segmentation region, search should stop and a vertical boundary be established whenever the next row to be searched is farther than about 8 cm from the row of the local maximum, $G_{localmax_{row}}$, where search started. This completely fills the chinks in the armor of the vertical contact boundary tests, preventing search from a palm heel local maximum from ever reaching an outstretched fingertip and vice versa.

3.2.6.5 Sloppy Segmentation Region Palm Heel Crease Test

The goal of sloppy region segmentation is to get as many of the electrodes influenced by the palms as possible into precisely two palm heels. This maximizes palm heel size and guarantees that the relatively fixed separation of about 5 cm

between palm heels can be measured. The relatively large palm heel sizes and the relatively large separations between palm heel centers help the identification system reliably distinguish fingers from palms. As long as the identification system has clear indication of two palm heels it cannot erroneously assign one of the palm heel identities to a thumb or pinky finger; *i.e.*, the combinatorial optimization constraints are stronger if the palm is consistently segmented into two parts. The palm heel separation is particularly critical in distinguishing a pair of adjacent fingertips starting a finger chord from a pair of palm heels when the fingertips are in the lower regions of the surface where palms are also expected. The identification system has means for identifying separate forepalms if necessary, but when possible they should be merged with one of the palm heels. Therefore, all partial minima in sloppy segmentation regions are ignored to maximize palm heel size except the large proximity valley or crease between the palm heels.

When the palm heels only touch the surface lightly, the proximity significance test will detect this crease and keep the two palm heel electrode groups separate. But as more pressure is applied, no electrode between them will remain at the background proximity level, and they will be separated only by a tall proximity valley with partial minima electrodes above the significance threshold. Fairly stringent tests are needed to detect this crease and not any other partial minima. First, anatomical constraints place the crease at least 2 cm from the center of either palm heel, so qualifying partial minima must be at a column j at least 2 cm from the column of the local maximum electrode where the search originated. Second, the proximity of the partial minimum electrode should be less than about half that of the local maximum, which requires that the proximity valley be fairly deep. Finally, the proximity valley must be fairly wide, so the partial minimum must extend past electrodes in next nearest neighbor columns, $S_{i,j-2}[n] > S_{i,j-1}[n] > S_{i,j}[n] < S_{i,j+1}[n] < S_{i,j+2}[n]$. These conditions allow horizontal boundaries between palm heels to be established in all rows which

the crease crosses. The only time the crease is not detected is when the entire center of the palm is pushed hard onto the surface, but when this is done the palm contacts are so large that the palm heel separation measurement is unimportant.

3.2.7 Combining Overlapping Groups

In sloppy segmentation regions it is possible for groups to overlap significantly because partial minima between local maxima do not act as boundaries. Typically when this happens the overlapping groups are part of a large fleshy contact such as a palm which, even after smoothing, has multiple local maxima. However, it is also necessary to get rid of the separate group around the local maxima of proximal phalanges when these are subsumed by fingertip groups. In the interest of presenting only one group per distinguishable fleshy contact to the rest of the system, the group combination stage of Figure 3.2 combines overlapping groups into single supergroups before parameter extraction.

Two groups are defined to be overlapping if the search originating local maximum electrode of one group is also an element of the other group. Two groups are connected if there is a sequence of overlapping groups between and including them; *i.e.*, overlap is transitive. A set of connected groups $\{G1..GN\}$ is combined into a supergroup GS by forming the semi-convex hull of the connected groups:

$$GS_{toprow} = \max_K GK_{toprow} \tag{3.5}$$

$$GS_{botrow} = \min_K GK_{botrow} \tag{3.6}$$

$$GS_{firstcol[i]} = \min_K GK_{firstcol[i]} \; \forall i : \; GS_{botrow} <= i <= GS_{toprow} \tag{3.7}$$

$$GS_{lastcol[i]} = \max_K GK_{lastcol[i]} \; \forall i : \; GS_{botrow} <= i <= GS_{toprow} \tag{3.8}$$

The semi-convex hull can include electrodes which are not part of any of the connected groups if a horizontal concavity lies between two of the connected groups, as occurs in Figure 3.19a). After consolidation into the supergroup is finished, the original connected set of groups $\{G1..GN\}$ is deleted.

3.2.8 Extracting Group Parameters

The last stage of the segmentation process extracts shape, size, and position parameters from each electrode group. Group centroid reflects hand contact position, and changes in contact position between successive images reflect finger velocity. The identification system utilizes geometric features of contacts such as total group proximity, eccentricity, and orientation to help distinguish finger, palm, and thumb contacts.

Though image smoothing does not affect the center of mass of a contact's proximity signal as measured over the whole image, it can diffuse some of the contact's signal outside of a group boundary into adjacent contacts or the image background. Diffusion can also increase the apparent radius or fitted ellipse axis lengths of a group. Therefore, to prevent biases during extraction, the parameterization process utilizes the undiffused image proximities $E_{ij}[n]$ instead of the smoothed proximities $S_{ij}[n]$ for all computations.

3.2.8.1 Centroid Computation

Given that G_E is the set of electrodes in group G, let $e_z = E_{e_i,e_j}[n]$ be the unsmoothed proximity of an electrode or pixel e, and let e_x and e_y be the coordinates on the surface of the electrode center in centimeters. To give a basic indicator of group position, the proximity-weighted center, or centroid, is computed from positions and proximities of the group's electrodes:

$$G_z = \sum_{e \in G_E} e_z \tag{3.9}$$

$$G_x = \sum_{e \in G_E} \frac{e_z e_x}{G_z} \tag{3.10}$$

$$G_y = \sum_{e \in G_E} \frac{e_z e_y}{G_z} \tag{3.11}$$

Note that since the total group proximity G_z integrates proximity over each pixel in the group, it depends upon both the size of a hand part, since large hand parts

tend to cause groups with more pixels, and upon the proximity to or pressure on the surface of a hand part.

Appendix B contains nonlinear interpolation methods used to ameliorate vertical interpolation biases and oscillations caused by the parallelogram electrodes.

3.2.8.2 Ellipse Fitting

While the user typically will not vary contact shape or orientation intentionally, such parameters will assist finger and hand identification in Chapter 4. Since most groups are convex, their shape is well approximated by ellipse parameters. The ellipse fitting procedure requires a unitary transformation of the group covariance matrix G_{cov} of second moments G_{xx}, G_{xy}, G_{yy}:

$$G_{cov} = \begin{bmatrix} G_{xx} & G_{xy} \\ G_{yx} & G_{yy} \end{bmatrix} \tag{3.12}$$

$$G_{xx} = \sum_{e \in G_E} e_z (G_x - e_x)^2 \tag{3.13}$$

$$G_{yx} = G_{xy} = \sum_{e \in G_E} e_z (G_x - e_x)(G_y - e_y) \tag{3.14}$$

$$G_{yy} = \sum_{e \in G_E} e_z (G_y - e_y)^2 \tag{3.15}$$

The eigenvalues λ_0 and λ_1 of the covariance matrix G_{cov} determine the ellipse axis lengths and orientation G_θ:

$$G_{major} = \sqrt{\lambda_0} \tag{3.16}$$

$$G_{minor} = \sqrt{\lambda_1} \tag{3.17}$$

$$G_\theta = \arctan\left(\frac{\lambda_0 - G_{xx}}{G_{xy}}\right) \tag{3.18}$$

where G_θ is uniquely wrapped into the range $[0, 180°)$.

For convenience while distinguishing fingertips from palms at higher system levels, the major and minor axis lengths are converted via their ratio into an eccentricity G_ϵ:

$$G_\epsilon = \frac{G_{major}}{G_{minor}} \tag{3.19}$$

Note that since the major axis length is always greater than or equal to the minor axis length, the eccentricity will always be greater than or equal to one. Finally, the total group proximity is empirically renormalized so that the typical curled fingertip will have a total proximity around one:

$$G_z := G_z / Z_{averageFingertip} \tag{3.20}$$

On low resolution parallelogram electrode arrays, the total group proximity G_z is a more reliable indicator of contact size as well as finger pressure than the fitted ellipse parameters. Therefore, if proximity images have low resolution, the orientation and eccentricity of small contacts are set to default values rather than their measured values, and total group proximity G_z is used as the primary measure of contact size instead of major and minor axis lengths.

3.2.9 Performance of the Segmentation Methods

Daily typing and chordic manipulation on the MTS by the author has verified that segmentations are flawless for the most commonly utilized hand configurations, at least as performed by a skilled operator. Any quantitative evaluation of segmentation performance would depend highly on the relative frequency of easily segmented versus difficult to segment hand configurations which arise in the application being studied. Rather than attempt an application and operator-dependent quantitative evaluation, this section presents a sampling of hand configurations which illustrate the importance of the more complex segmentation rules and the rare cases in which segmentation fails.

A flattened hand with fingers squeezed together includes most of the contact juxtapositions which require proper alignment of the sloppy segmentation regions for correct segmentation. Figure 3.6 includes both the unsmoothed and diffused

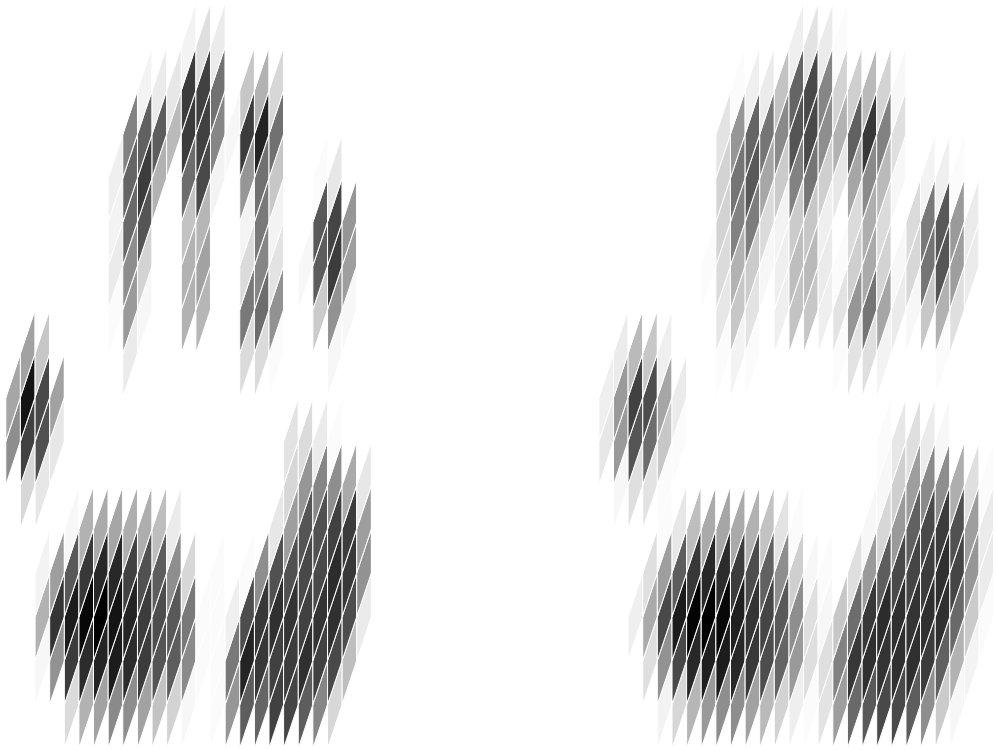


Figure 3.6: Unsmoothed a) and diffused b) proximity images of a flattened hand with the fingers squeezed against one another rather than spread out. Note that smoothed proximities of pixels near the center of each contact are more even but proximity has also bled outward around contact edges, as is especially noticeable for the thumb contact.

images of a flattened, squeezed hand. Note that the fingers are very close together, the proximal phalanges are touching the surface, and the outer palm heel causes two local maxima (darkest electrodes at lower left and upper right of contact) which are not merged by smoothing. Only the center of the palm and the forepalms remain suspended above the surface.

Note that usually each palm heel will only cause one local maximum, especially in the smoothed image. Multiple maxima which are distinct enough to remain separate during diffusion only appear sporadically due to odd distributions of hand pressure. Several images which only had one local maximum per palm heel were discarded before a hand posture was found which produced the dual maxima of Figure 3.6. Nevertheless, multiple maxima appear often enough that it is worthwhile to combine overlapping groups whenever possible. This avoids confusing the identification system with more than two palm heel groups per hand and avoids additional smoothing which could jeopardize segmentation of adjacent fingertips.

To demonstrate the types of segmentation failures which can occur when the hand position estimate is incorrect, the image of Figure 3.6 was segmented once with the strict segmentation region covering the whole image and again with the sloppy segmentation region covering the whole image. The former case will show what can happen when the palms unexpectedly touch down in the strict segmentation region, and the latter case will show what happens when fingertips unexpectedly touch down in the sloppy segmentation region, contradicting the last known hand position retained by the hand position estimator. Figure 3.7 displays segmentation maps in which electrodes numbered the same are members of the same segmentation group; the ordering of the numbers is arbitrary. Note that these segmentation maps and all that follow include only the final combined supergroups, not individual overlapping groups before they are combined.

Figure 3.7a shows the segmentation map obtained when the whole hand is processed with strict segmentation rules, *i.e.*, when the sloppy segmentation region is moved off the palms down to the lower right corner of the image. Since strict segmentation rules establish boundaries along the proximity valley between the dual outer palm heel local maxima, preventing each local maximum's group from overlapping the other, the outer palm heel contact remains erroneously broken into two

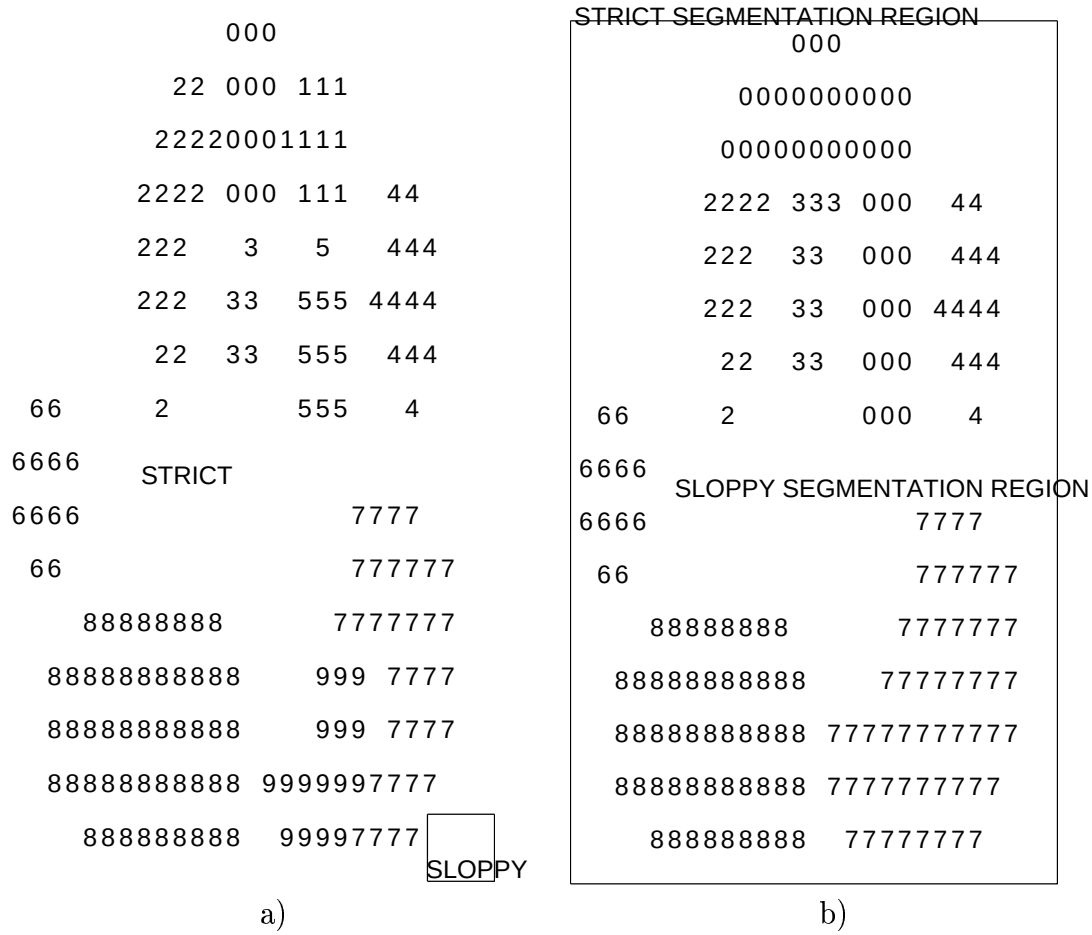


Figure 3.7: Segmentation results for the flattened hand using either a) strict segmentation rules for both fingers and palm heels or b) using sloppy segmentation rules for the whole hand. In a) the outer palm heel contact erroneously maintains two separate groups corresponding to the dual local maxima, and some of the electrodes at the edge of the palm heel contacts are excluded from all groups. Tolerance of flattened finger vertical minima is also disabled, causing splits in the segmentation between proximal and distal portions of the middle and ring fingers. In b) the lack of horizontal partial minimum detection causes the index, middle, and ring fingertips to erroneously merge into one group. Electrodes influenced by the thumb or pinky are grouped correctly in both a) and b).

groups. Comparing the size of the palm contacts of the original image (Figure 3.6a) to the segmentation map suggests that the strict segmentation rules have caused a few palm contact electrodes to be entirely left out of all the palm groups. Though not many electrodes have been excluded for this example image, sometimes this can cause the extracted group size and total proximity parameters to be erroneously low, making it harder for the identification system to distinguish fingertips from palms when the palms unexpectedly escape the sloppy segmentation region. The improper alignment of the sloppy segmentation region also disables tolerance of flattened finger vertical minima. This in turn causes proximal phalanges with distinct local maxima (those from the middle and ring fingers in this case) to fail to merge with their respective fingertip groups.

Figure 3.7b shows the opposite extreme in which the whole hand lies within the sloppy segmentation region and is segmented with sloppy segmentation rules. In this case the palm is properly segmented into two large heel groups which contain all palm contact electrodes. This occurs because sloppy segmentation rules allow the groups from each of the outer palm heel dual local maxima to overlap and thus be combined via their semi-convex hull. However, since sloppy segmentation rules do not include horizontal partial minima tests, the horizontally adjacent index, middle, and ring fingertips merge. Such incorrect fingertip merging can occur any time the fingertips touch down squeezed-together in a sloppy segmentation region. The proximal phalanges of the index and middle fingers remain separate from the merged fingertip group not because of the vertical minima between fingertips and proximal phalanges but because the semi-convex search pattern can only follow one vertical offshoot from each row, and in this case the search always traversed the proximal phalange of the ring finger.

Figure 3.8 shows the segmentation map when the sloppy segmentation region is properly aligned on the palm contacts. This segmentation region alignment or

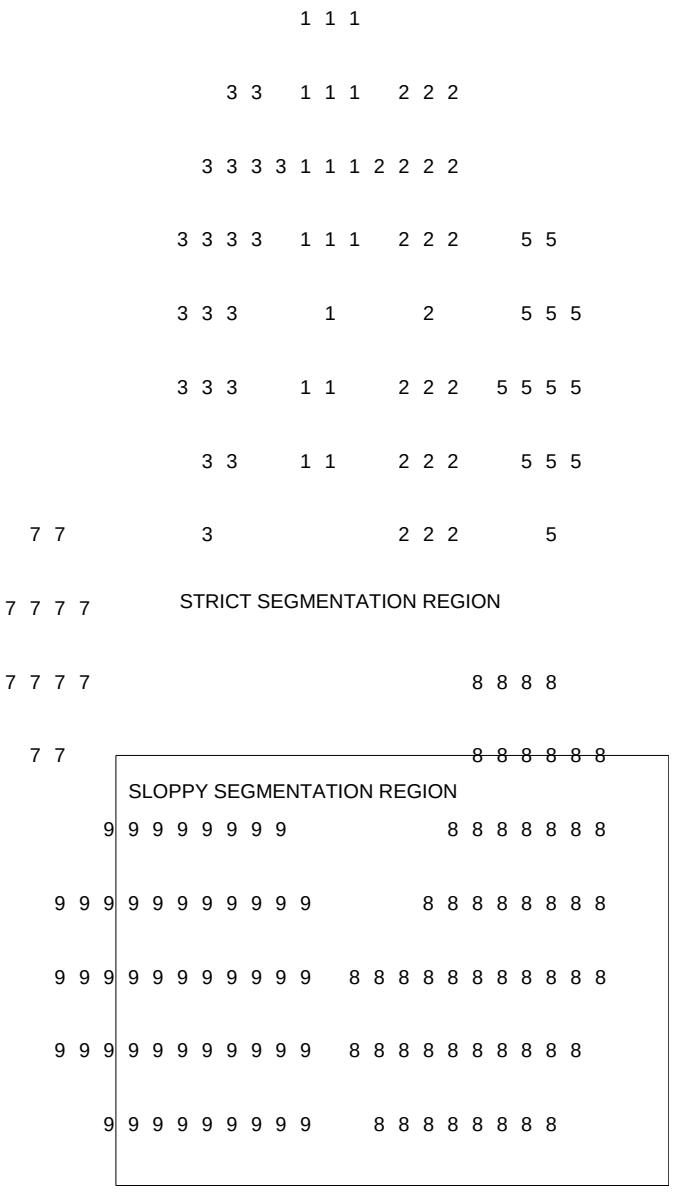


Figure 3.8: The correct segmentation for the flattened hand obtained by applying sloppy segmentation rules in the box around the palm heels and strict segmentation rules for the fingers. All electrodes proximal to the palm heels are included in their groups, yet the palm groups are split by the crease halfway between the palm heels. The whole of each finger, *i.e.*, both the distal tip and proximal phalange, combines into one group separate from the adjacent fingers.

any alignment within a couple centimeters of it achieves the correct segmentation for both fingers and palm heels. By the time the whole hand has flattened onto the surface, finger identifications and thus the hand position estimate will surely have stabilized to their correct values, invariably producing this correct segmentation under actual operating conditions.

In contrast to the squeezed, flattened hand which requires proper alignment of the sloppy region for correct segmentation, the default or neutral hand posture of Figure 2.8 is segmented correctly regardless of where strict or sloppy rules are applied. The segmentation map for the whole hand in the sloppy region is shown in Figure 3.9, but the same map is obtained with the sloppy region over palms only or with strict segmentation everywhere, with the exception that with strict segmentation of the palm heels a few of the electrodes on the periphery of the palm contacts can be excluded from the palm heel groups. The image can be segmented correctly regardless of segmentation region alignment because each distinguishable contact has only one local maximum and the contacts are all separated from one another by electrodes at background proximity levels. Thus the proximity significance test common to both strict and sloppy segmentation regions is sufficient to establish group boundaries. Note also that this easily segmented hand posture and variations upon it comprise the most commonly performed hand configurations.

A rotated variation (Figure 3.10) of this neutral hand configuration demonstrates the need for diagonal partial minima detection. Sloppy segmentation (Figure 3.11a) of the diagonally adjacent fingertips causes them to be merged into two groups. Strict segmentation of them (Figure 3.11b) detects the diagonal partial minima between them and keeps them in four separate groups. As will be shown in Figure 3.12, any further rotation causes fingertips to merge despite strict segmentation.

In Figure 3.12 the hand is rotated a full 90° from the neutral or default pos-

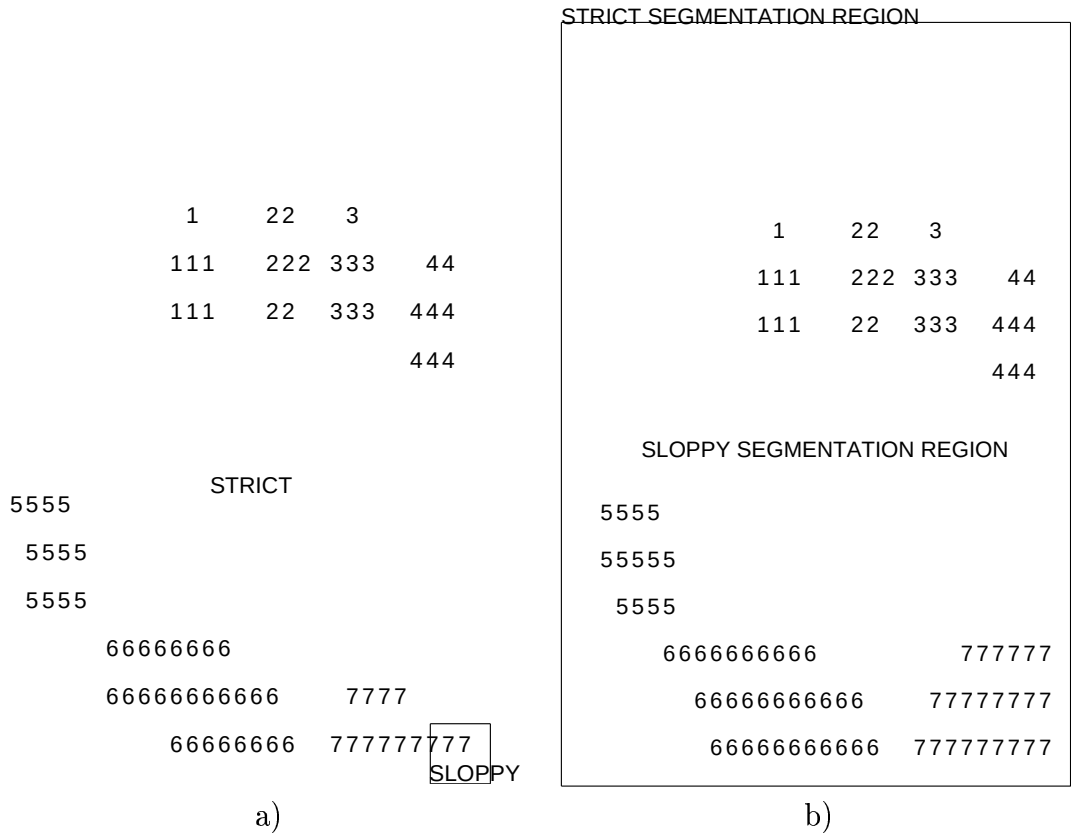


Figure 3.9: This correct segmentation of the neutral hand posture (Figure 2.8) is obtained regardless of where strict a) or sloppy b) segmentation rules are applied since contacts are relatively small and well-separated.

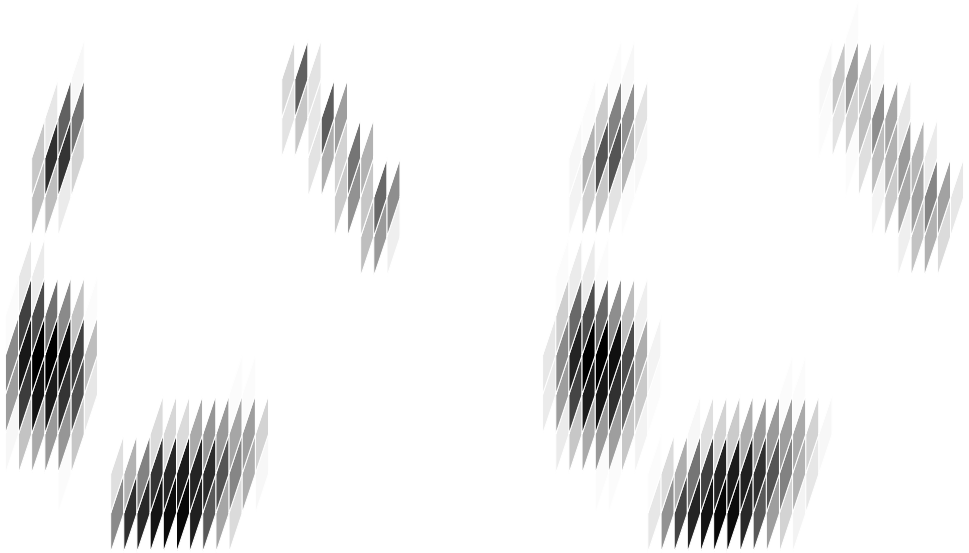


Figure 3.10: Unsmoothed a) and diffused b) proximity images of a partially closed right hand rotated clockwise 45°.

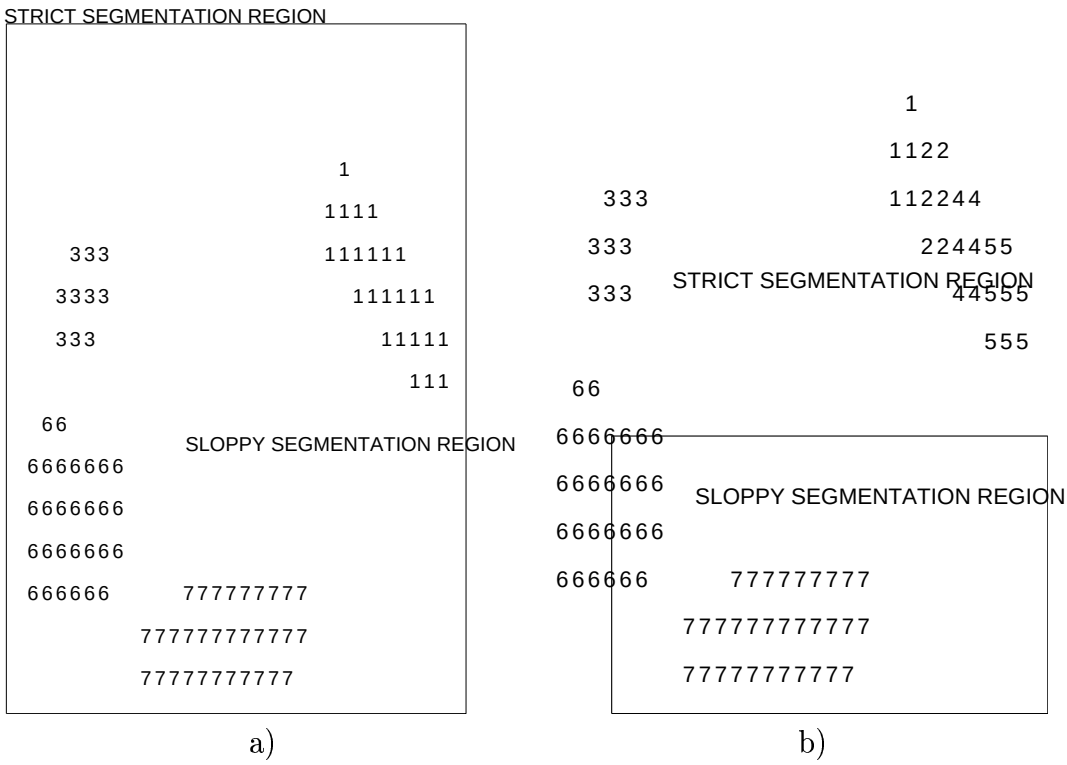
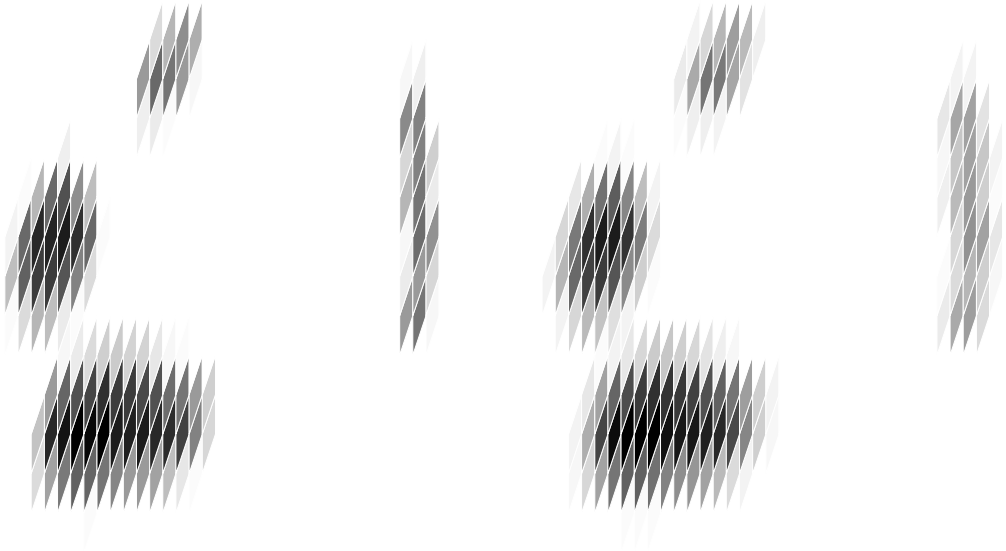


Figure 3.11: Sloppy segmentation a) of fingertips in a slanted row causes some of them to be merged. However, the diagonal minima tests of strict segmentation b) keep fingertip groups properly separated even when the row of fingertips is slanted as much as 45°.



a) b)

Figure 3.12: Unsmoothed a) and diffused b) proximity images of a partially closed right hand rotated clockwise 90°, fully sideways.

ture. As the segmentation map with properly aligned sloppy regions in Figure 3.13 indicates, vertical smearing by the parallelogram electrodes hides the proximity valleys between the fingertips, causing unavoidable merging even with strict segmentation rules. Note that performing this sideways hand configuration from the normal

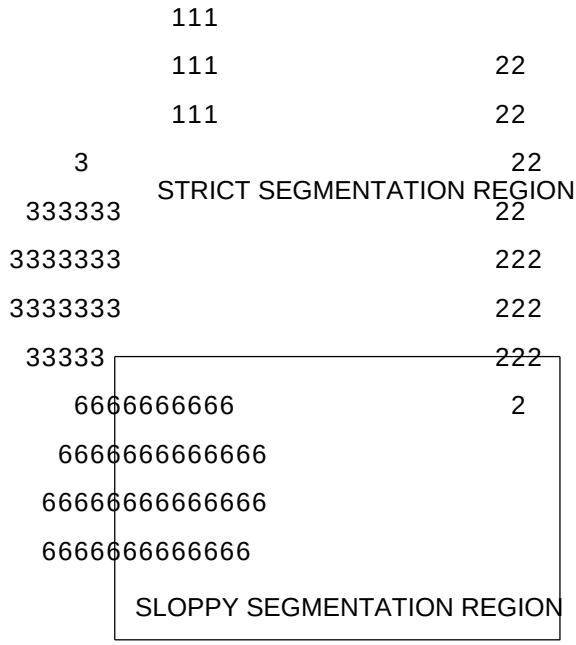


Figure 3.13: All segmentation rules fail to segment the column of fingertips because the vertical smearing by vertically interleaved parallelogram electrodes obscures the local proximity maxima normally caused by each fingertip.

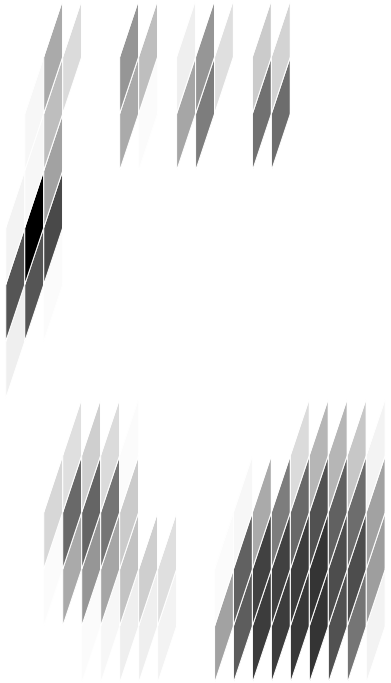
sitting posture requires awkward contortions of the body, so the configuration would only be encountered regularly if the MTS was mounted on a pedestal so it could be approached from all sides.

The parallelogram electrode smearing also causes segmentation errors when the thumb passes just behind a fingertip. Figure 3.14a shows the thumb as close as

it can get to the index fingertip and still be segmented correctly with strict rules (Figure 3.14b). Figure 3.15a is an image of a closed hand with the thumb tip fully pushed up against the back of the fingertip. In this case the thumb tip and fingertip are so close that parallelogram electrode smearing obscures the vertical partial minima between them, causing them to be merged into one group (Figure 3.15b). Since the pen grip hand configuration (Section 2.3.4) contains this same juxtaposition of the thumb and index fingertip, it cannot be segmented properly with the current parallelogram electrode row spacing either. Note that if proximal phalange merging for flattened finger segmentation (Section 3.2.6.3) was erroneously enabled during segmentation of Figure 3.14a, the vertical partial minimum would have been ignored, causing thumb and fingertip to be merged as in Figure 3.15b.

Figure 3.17 is the correct segmentation map for the flattened hand with outstretched fingers of Figure 3.16. This map illustrates that a forepalm contact gets its own group if it has a separate local maximum and cannot be combined with the palm heels through mutual overlap. This is not considered an error because the identification system can reliable identify forepalms when the rest of the hand is flattened onto the surface. Note also that even though the proximity images show that the index finger is connected to the inner palm heel and the pinky finger is connected to the outer palm heel, the segmentation groups for fingers and palms remain separate.

Figure 3.18 exposes the shortcomings of flattened finger segmentation when the fingers are not oriented near vertical. If the fingers lie at a slant of more than about 25° off vertical, each finger begins breaking into multiple groups due to detection of diagonal partial minima instead of vertical partial minima. Figure 3.19b shows the resulting segmentation map with fingertip groups separated from proximal phalange groups even when the sloppy segmentation region is aligned properly. This failure is a consequence of the assumption that proximity ridges will always be



a)

```
 1    2
 11   22  333  44
55    22  33   44
55          3    4
555
555
```

STRICT SEGMENTATION REGION

SLOPPY SEGMENTATION REGION

```
 6666          777777
66666         77777777
6666666     777777777
  66666     777777777
```

b)

Figure 3.14: Unsmoothed proximity image and properly aligned segmentation map of a thumb passing about a centimeter behind the index fingertip.

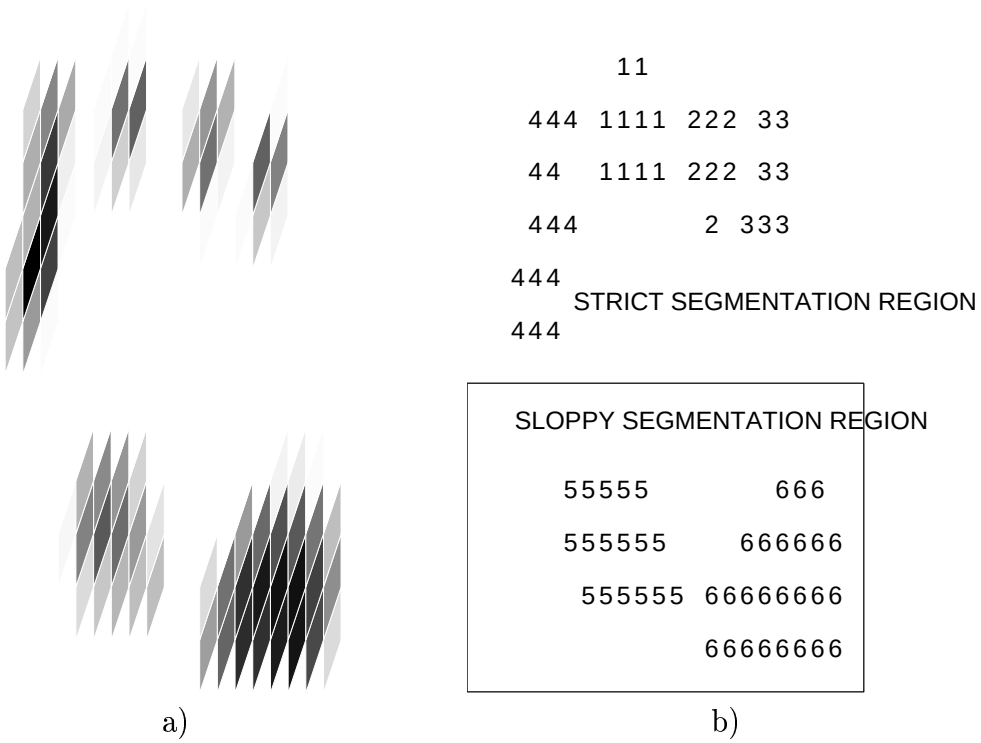


Figure 3.15: Unsmoothed proximity image and properly aligned segmentation map of a thumb touching the back of the index fingertip.

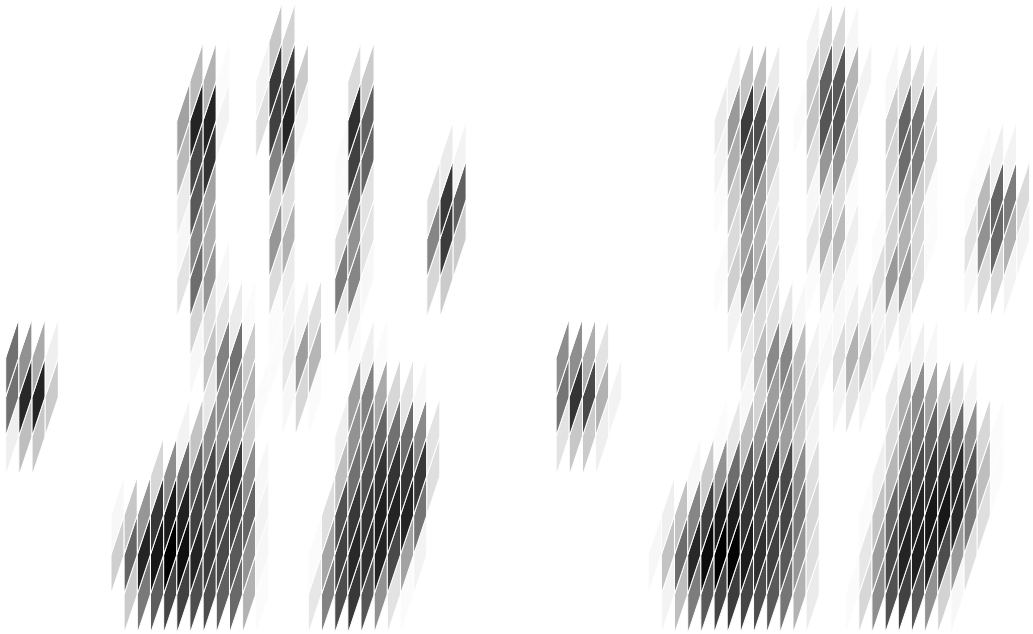


Figure 3.16: Unsmoothed a) and diffused b) proximity images of a right hand flattened onto the surface so hard that the forepalms are touching.

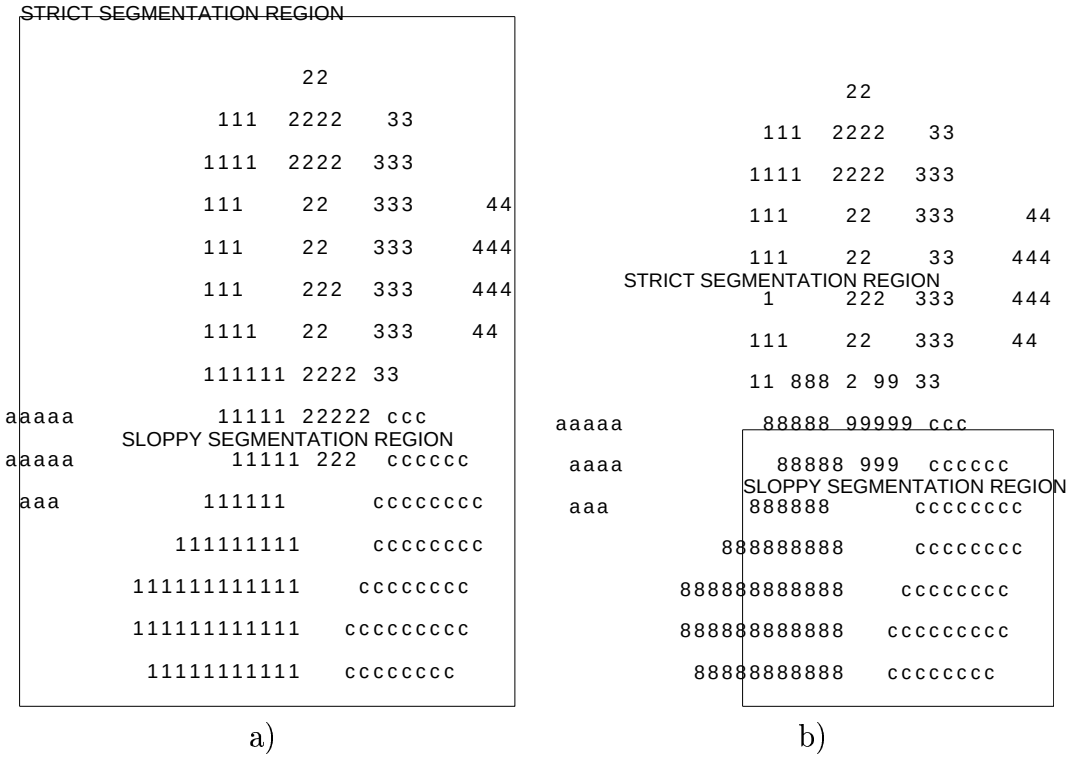


Figure 3.17: All sloppy segmentation a) of the flattened right hand of Figure 3.16, and the correct segmentation using properly aligned sloppy regions b).

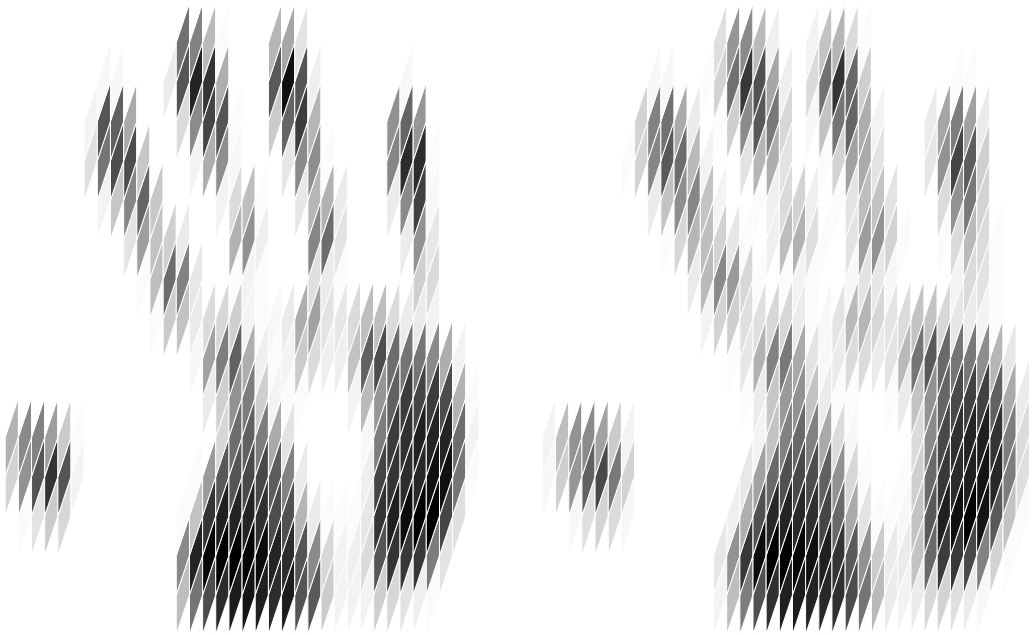


Figure 3.18: Unsmoothed a) and diffused b) proximity images of a flattened right hand rotated counter-clockwise about 30°.

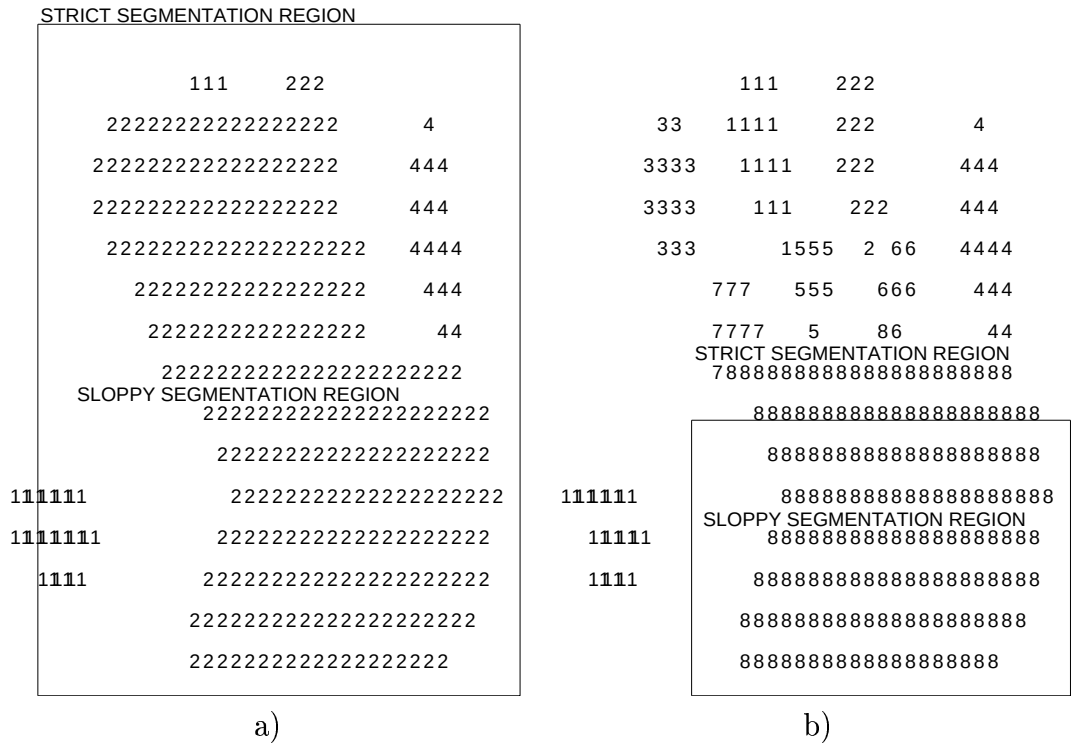


Figure 3.19: Segmentation maps for the rotated, flattened hand. Sloppy segmentation everywhere a) causes all groups to overlap. A single supergroup forms from the semi-convex hull around these groups and encloses the whole hand. With sloppy segmentation only over the palms b), the diagonal partial minima detection of strict segmentation keeps the fingertip and proximal phalange groups erroneously split.

connected through nearest diagonal neighbors. Sufficiently oblique ridges such as these which cause maxima in every other column are broken up by diagonal minima tests. Occasionally this type of failure splits a diagonally-oriented thumb pressed hard onto the surface into two groups. This failure would be less likely with a smaller row spacing which increases the angle between nearest diagonal neighbors to 45°. Combination via semi-convex hull of the entire flattened palm into a single group (Figure 3.19b) can be handled by the identification system and is not considered an error as long as the merging remains stable across successive proximity images.

Another measure of segmentation performance is the consistency of group membership across successive proximity images of a stationary hand. Segmentation consistency has a major impact on the path tracking to be discussed in section 3.3. If segmentations of a hand contact are not consistent across images, the measured hand contact centroid can erroneously shift between images and cause jitter in the contact velocity computed by the path tracking module. If large contacts suddenly merge or break apart in a new image the centroids may move so much that the path tracker concludes a hand part has lifted off or newly touched down. As long as these unstable hand parts are identified as forepalms or palm heels such spurious touchdowns will have no deleterious effects, but if the unstable hand parts are identified as fingers whose centroids are near keys of the key layout, the system can falsely interpret them as keypresses.

The segmentation of most unflattened hand postures is perfectly consistent, but when palms are fully flattened, forepalm-influenced electrodes can unstably shift between finger, palm heel, and independent forepalm groups, or the palm heels can unstably merge together for one image and break up in the next due to subtle changes in palm contact topology. Though such instabilities have been observed to perturb finger centroids enough to cause spurious key activations during hand flattening tests, full hand flattening does not occur during normal MTS operation

or hand resting. Therefore, like the other rare cases of segmentation failure, this problem will not be investigated further until it is known whether lower noise, higher resolution sensor arrays will make it disappear entirely.

3.3 Persistent Path Tracking

3.3.1 Introduction to the Path Tracking Problem

Electrode groups are transitory in the sense that the segmentation algorithm reconstructs them from scratch for each proximity image. It is then the responsibility of the path tracking process to chain together those groups from successive proximity images which correspond to the same physical hand contact. To determine where each hand part has moved since the last proximity image, the tracking algorithm must decide which current groups should be matched with which existing contact paths. As a general rule, a group and path arising from the same contact will be closer to one another than to other groups and paths. Also, biomechanical constraints on lateral finger velocity and acceleration limit how far a finger can travel between images. Therefore a group and path should not be matched unless they are within a distance known as the tracking radius of one another. Tracking performance can be improved by incorporating velocities measured along existing paths in previous images into the prediction of current surface contact location.

The path tracking process must also determine when a physical hand contact newly touches down or lifts off the surface. Since the typical lateral separation between fingers is greater than the tracking radius for reasonable image scan rates, finger touchdown and liftoff are easily detected by the fact that touchdown usually causes a new group to appear outside the tracking radii of existing paths, and liftoff will leave an active path without a group within its tracking radius. To prevent improper breaking of paths at high finger speeds, each path's tracking radius P_{rtrack} is made dependent on its existing speed and contact size.

Rubine [130] addressed the path tracking problem for up to three fingers detected by a "Sensor Frame" [107, 108]. He did not find it necessary to predict current contact locations from past path velocity or acceleration. He simply used the last known path position as the predicted position. Since Rubine only dealt with 3 fingers, *i.e.*, six possible one-to-one assignments of groups to paths, he simply evaluated all possible pairings and picked the one which minimized the sum of the distances between each path and its assigned group. However, counting all fingers, thumbs, and palm heel and forepalm contacts from both hands, up to 20 groups can be present at any one time on the MTS. Clearly the MTS cannot do a brute force enumeration of all 20! possible assignments, but the same assignment optimization algorithms invoked for finger identification in Section 4.4 could be utilized to efficiently minimize the assignment sum.

Any group-path pairing method must perform a global optimization in the sense that it must handle both the case that several groups are clustered around a single path, wherein only the closest group in the cluster should be assigned to the path, and the case that several paths are clustered around a single group, wherein only the closest path in the cluster should be assigned to the group. Thus the decision to pair a group and path cannot be made based solely upon the distance between them without considering the pairing distances of other groups or paths nearby. The pairing method presented here utilizes a rule related to the shared near neighbors clustering technique introduced by Jarvis and Patrick [72] instead of explicitly minimizing the sum of pairing distances. This rule only accepts assignments between groups and paths that are closest to one another. The nearest neighbor clustering rule and assignment sum minimization may produce slightly different pairings for very tight clusters of groups and paths, but given reasonable frame rates, adult finger spacings, and velocity-based path prediction, groups remain so much closer to their actual paths than other paths that the behavior of the

pairing method for marginal cases does not matter.

Consult Figure 3.20 for a summary of the steps in the path tracking process described below.

3.3.2 Prediction of Contact Location

Whether the prediction of path contact locations needs to include past path velocity depends on the minimum possible finger separation, maximum expected finger speed and acceleration, and the array scan or frame rate. If segmented properly, group centroids are usually separated by at least 1.5 cm. The MTS frame rate is currently 50 fps, but can easily be raised to 100 fps. Because hand and finger motions tend to be smooth, acceleration is quite low except at the beginning and end of slides. At typical mouse manipulation speeds a finger will travel less than 1 mm between frames. However, consideration of Fitts' Law [25] suggests hand pointing speeds can reach 180 cm/s. The fastest hand slides observed across the MTS have only reached 100 cm/s, but these still cause each finger to travel up to 2 cm between frames, a value comparable to the nominal fingertip separation. If a row of fingers quickly slides horizontally, one fingertip can thus appear right over the last known location of an adjacent fingertip unless this last known location is updated with past finger velocities.

Including previously measured velocity in the location prediction improves the prediction except when a finger suddenly starts or stops or changes direction. Since such high acceleration events occur less often than zero acceleration events, the benefits of velocity-based prediction outweigh the potentially bad predictions during finger acceleration. Let $P_x[n-1], P_y[n-1]$ be the position of path P from time step $n-1$ and $P_{vx}[n-1], P_{vy}[n-1]$ the last known velocity. The velocity-predicted path continuation is then:

$$P_{predx}[n] \;=\; P_x[n-1] + \triangle t P_{vx}[n-1] \tag{3.21}$$

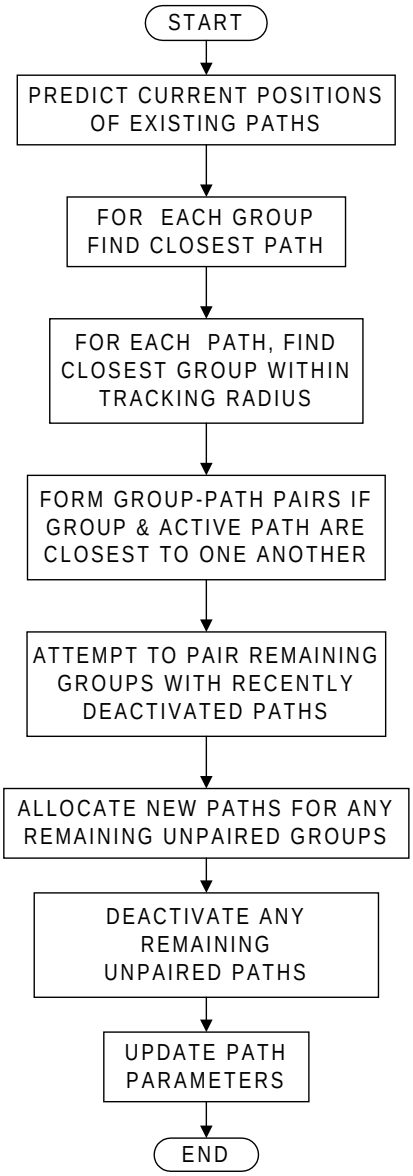


Figure 3.20: Flow chart summarizing the contact path tracking algorithm.

$$P_{predy}[n] \;=\; P_y[n-1] + \triangle t P_{vy}[n-1] \tag{3.22}$$

Possibly the reason Rubine found velocity-based predictions unnecessary was that the Sensor Frame had a higher frame rate or that he was only tracking finger motion with the hand in a relatively fixed position over a small area. The MTS is large enough for lateral slides of the whole hand and forearm. These easily reach speeds several times higher than individual finger motions from a stationary hand.

3.3.3 Mutually Closest Pairing Rule

Let the set of paths active in the previous image be $\mathcal{PA}$, and let the set of electrode groups constructed in the current image be $\mathcal{G}$. For each active group Gk, find the closest active path and record the distance to it:

$$Gk_{closestP} = \underset{Pl \in \mathcal{PA}}{\operatorname{argmin}}\, d^2(Gk, Pl) \qquad \forall\, Gk \in \mathcal{G} \tag{3.23}$$

$$Gk_{closestPdist^2} = \min_{Pl \in \mathcal{PA}} d^2(Gk, Pl) \qquad \forall\, Gk \in \mathcal{G} \tag{3.24}$$

where the squared Euclidean distance is an easily computed distance metric:

$$d^2(Gk, Pl) = (Gk_x - Pl_{predx})^2 + (Gk_y - Pl_{predy})^2 \tag{3.25}$$

Then for each active path Pl, find the closest active group and record the distance to it:

$$Pl_{closestG} = \underset{Gk \in \mathcal{G}}{\operatorname{argmin}}\, d^2(Gk, Pl) \qquad \forall\, Pl \in \mathcal{PA} \tag{3.26}$$

$$Pl_{closestGdist^2} = \min_{Gk \in \mathcal{G}} d^2(Gk, Pl) \qquad \forall\, Pl \in \mathcal{PA} \tag{3.27}$$

A group Gk and path Pl are only paired with one another if they are closest to one another, *i.e.*, $Gk_{closestP}$ and $Pl_{closestG}$ refer to one another, and the distance between them is less than the tracking radius. The nominal tracking radius is about 1 cm. All of the following conditions must hold:

$$Gk_{closestP} \;\equiv\; Pl \tag{3.28}$$

$$Pl_{closestG} \;\equiv\; Gk \tag{3.29}$$

$$Pl_{closestGdist^2} \;<\; Pl^2_{rtrack} \tag{3.30}$$

Any active group which cannot be paired with an active path under these constraints is allocated a new path, representing touchdown of a new finger onto the surface. Any active path which cannot be so paired with an active group is deactivated, representing hand part liftoff from the surface.

To aid finger tap debouncing and detection of repetitive finger taps, it is useful to preserve continuity of path assignment between taps over the same location. When a new path needs to be allocated for an isolated group, *i.e.*, the group cannot be assigned to any active path, priority is given to any recently deactivated paths within the tracking radius of the group. The closest path which has been deactivated within the last second or so is reactivated, assigned to the group, and specially marked as reactivated. It's release time is stored in $P_{tpreviousrelease}$ to aid in detection of double-click timing. If no recently deactivated paths exist nearby, a totally new path is started.

3.3.4 Path Parameters

The final step of path tracking is to incorporate the extracted parameters of each group into its assigned path via standard filtering techniques. The MTS applies the simple autoregressive filter equations shown below to update the path position ($P_x[n], P_y[n], P_z[n]$), velocity ($P_v x[n], P_v y[n]$), and shape $P_\theta[n], P_\epsilon[n]$ parameters from corresponding group parameters. If a path P has just been started by group G at time step n, *i.e.*, a hand part has just touched down, its parameters are initialized as follows:

$$P_{press_t} = t \tag{3.31}$$

$$P_{press_x} = G_x \tag{3.32}$$

$$P_{press_y} = G_y \tag{3.33}$$

$$P_x[n] = G_x \tag{3.34}$$

$$P_y[n] = G_y \tag{3.35}$$

$$P_z[n] = G_z \tag{3.36}$$

$$P_\theta[n] = G_\theta \tag{3.37}$$

$$P_\epsilon[n] = G_\epsilon \tag{3.38}$$

$$P_{vx}[n] = 0 \tag{3.39}$$

$$P_{vy}[n] = 0 \tag{3.40}$$

$$P_{vz}[n] = G_z/\triangle t \tag{3.41}$$

else if group G is a continuation of active path $P[n-1]$ to time step n:

$$P_x[n] = G_\alpha G_x + (1-G_\alpha)(P_{predict_x}[n-1]) \tag{3.42}$$

$$= G_\alpha G_x + (1-G_\alpha)(P_x[n-1] + \triangle t P_{vx}[n-1]) \tag{3.43}$$

$$= G_\alpha G_x + (1-G_\alpha)(2P_x[n-1] - P_x[n-2]) \tag{3.44}$$

$$P_y[n] = G_\alpha G_y + (1-G_\alpha)(P_{predict_y}[n-1]) \tag{3.45}$$

$$= G_\alpha G_y + (1-G_\alpha)(P_y[n-1] + \triangle t P_{vy}[n-1]) \tag{3.46}$$

$$= G_\alpha G_y + (1-G_\alpha)(2P_y[n-1] - P_y[n-2]) \tag{3.47}$$

$$P_x[n] = G_\alpha G_x + (1-G_\alpha)(P_{predx}[n-1]) \tag{3.48}$$

$$P_y[n] = G_\alpha G_y + (1-G_\alpha)(P_{predy}[n-1]) \tag{3.49}$$

$$P_z[n] = G_\alpha G_z + (1-G_\alpha)P_z[n-1] \tag{3.50}$$

$$P_\theta[n] = G_\alpha G_\theta + (1-G_\alpha)P_\theta[n-1] \tag{3.51}$$

$$P_\epsilon[n] = G_\alpha G_\epsilon + (1-G_\alpha)P_\epsilon[n-1] \tag{3.52}$$

$$P_{vx}[n] = (P_x[n] - P_x[n-1])/\triangle t \tag{3.53}$$

$$P_{vy}[n] = (P_y[n] - P_y[n-1])/\triangle t \tag{3.54}$$

$$P_{vz}[n] = (P_z[n] - P_z[n-1])/\triangle t \tag{3.55}$$

It is also useful to compute the magnitude $P_{speed}[n]$ and angle $P_{dir}[n]$ from the velocity vector $(P_{vx}[n], P_{vx}[n])$. The filters are first or second order autoregressive where the amount of filter memory increases with $(1-G_\alpha)$. The second-order position

filter has zero tracking delay during constant velocity motion, but position filter output can lag behind actual finger position during accelerations or slightly overshoot during decelerations. Since the reliability of position measurements increases considerably with total proximity P_z, the low-pass filter pole G_α is decreased for groups with total proximities lower than normal:

$$G_\alpha \approx \begin{cases} .8 & \text{if } P_z > 1 \\ .4 + .4P_z & \text{else if } P_z <= 1 \end{cases} \tag{3.56}$$

Thus when signals are weak, the system relies heavily on the previously established path velocity, but when the finger firmly touches the surface causing a strong, reliable signal, the system relies entirely on the current group centroid measurement.

Since the MTS does not yet recognize complex path gestures or handwriting, it does not need to store all past points of each finger trajectory. However, to aid detection of finger chords and taps, the MTS does retain temporal markers with finger position and size parameters from important stages of the path life cycle such as finger touch down ($P_{press_t}, P_{press_x}, P_{press_y}$), stabilizations in finger proximity ($P_{peak_t}, P_{peak_x}, P_{peak_y}, P_{peak_z}$), and finger liftoff ($P_{release_t}, P_{release_x}, P_{release_y}$). As a finger taps the surface, proximity quickly rises and then plateaus until the finger lifts off. To ensure the peak proximity marker is available fairly soon after touchdown, its parameters are actually captured when the rate of change of proximity $P_{vz}[n]$ falls below a very small positive threshold, indicating the beginning but not necessarily the peak of a proximity plateau. These important finger activity markers would become inconsistent if a faulty path tracking algorithm inadvertently reshuffled paths.

3.3.5 Path Tracking Results

The path tracking process described above performs flawlessly for normal operating hand speeds and a frame rate of 50 fps. It will chain groups caused by the same hand contacts indefinitely as they slide across the surface, starting new

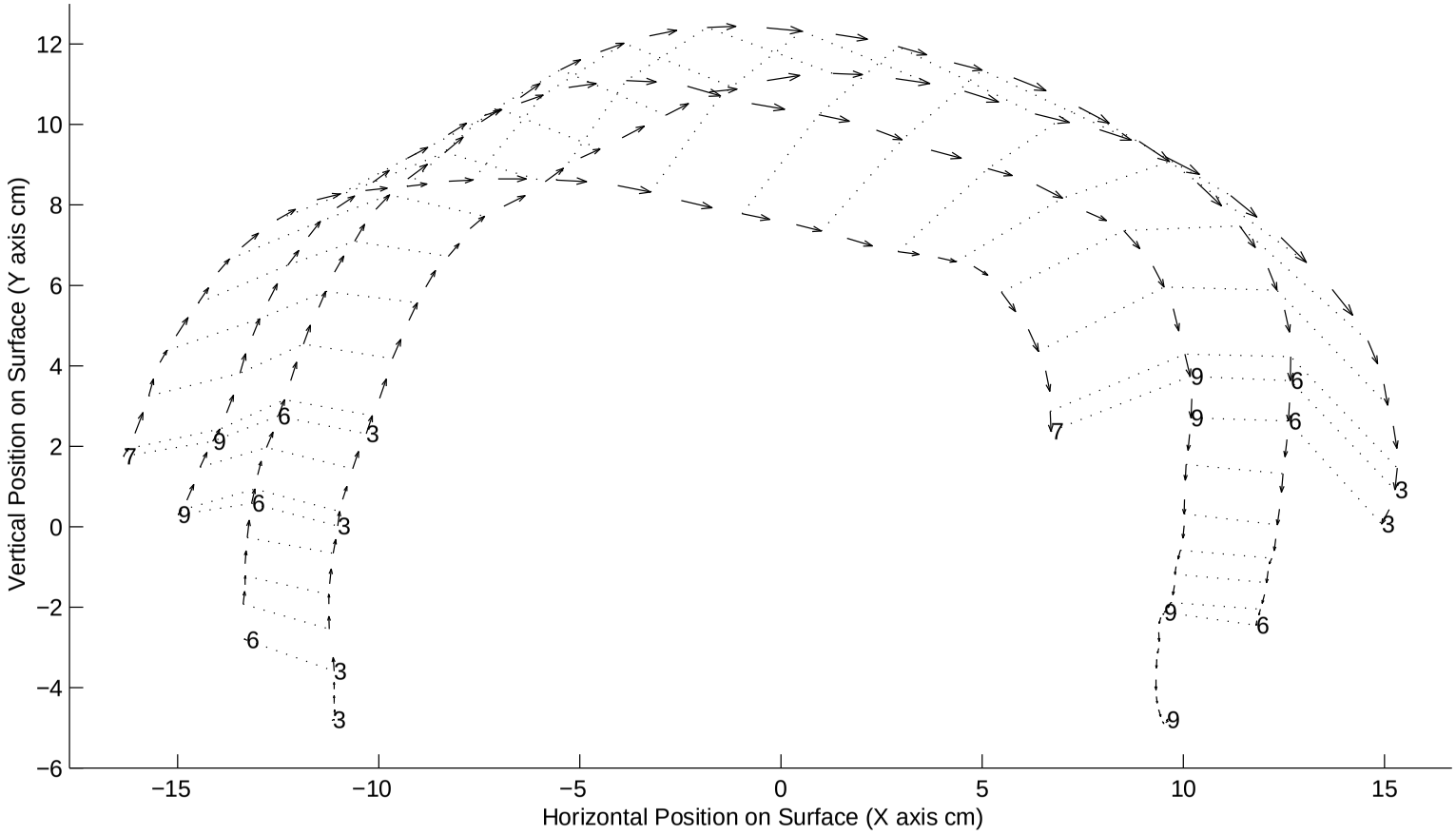


Figure 3.21: Trajectories of four left hand fingertips touching down asynchronously on the lower left of the surface and sliding in an arc to lift off at the lower right. Arrow tips and tails represent filtered contact positions $(Pi_x[n], Pi_y[n])$ in successive proximity images n. The four path indices i are printed at touchdown and liftoff to prove continuity of the tracking algorithm. Index order is arbitrary. To indicate relative timing of the trajectories, dotted lines connect all current contacts at certain timesteps.

paths only due to extreme deceleration or finger touch down. Figure 3.21 gives a representative example of tracking for four sliding fingers. The only case in which paths are known to get reshuffled or erroneously broken is when forepalm contact segmentations for fully flattened hands become unstable.

3.4 Summary

The centers of hand contacts invariably cause local maxima in smoothed proximity images, so proximity image segmentation algorithms can be very efficient by starting segmentation group growth at significant local maxima. Since most contacts are convex, semi-convex group search patterns and data structures can efficiently encapsulate each contact's electrodes without including electrodes from neighboring contacts. Though in the most commonly used hand configurations contacts are well-separated and easy to segment, other important configurations contain contacts separated only by ambiguous partial minima. Contextual feedback of past hand position and flattened finger status is necessary to decide which of these partial minima should be treated as contact edges. Careful tuning of the edge detection rules and segmentation regions so that electrode groups reliably correspond to functionally distinct hand parts such as fingertips, thumbs, or palm heels allows the contact identification system of the following chapter to be relatively simple yet robust. Vertical smearing by interleaved parallelogram electrodes causes most of the rare segmentation failures, so these failures can be best addressed by improvements in vertical sensor density which phase out parallelogram electrodes, rather than by improvements in the segmentation algorithms.

Path tracking is not terribly difficult given typical finger spacings and frame rates above 50 fps, especially if velocity-based prediction is utilized. Nevertheless, its flawlessness also simplifies the jobs of the identification system in the following chapter and the synchronization system in Chapter 5. Once the finger and hand identification system has correctly assigned an identity to a contact, continuation

of that contact's path by the path tracking system will be sufficient to retain the identity until the contact lifts off, thus avoiding computationally expensive and potentially destabilizing reidentification in every successive proximity image. Accurate contact tracking also ensures the validity of the path life cycle markers used to debounce key presses and detect synchronous touchdowns or liftoffs of multiple fingers.

Chapter 4

FINGER IDENTIFICATION AND HAND POSITION ESTIMATION

This chapter will tackle the main shortcoming of capacitive proximity sensing, establishing fingertip and palm heel identities for surface contacts when most of the intervening hand structure is missing from the proximity images. Finger identification has not been necessary for interactions with commercial touchpads or touchscreens because most cannot track more than one finger at a time, and those that can detect two or three fingers only utilize the contact count. If the MTS was only to be used for typing, trying to identify each surface contact might not be worthwhile because key taps should be distinguished by their spatial location, not which finger strikes the key. But recognition of the rich, bimanual chordic manipulations demonstrated in Chapter 5 demands reliable clustering of surface contacts with their originating hand as well as reliable finger ordering and thumb identification within each hand. Also, palm contacts must be properly identified so that operators can safely rest the entire hand anywhere on the surface without palm motion being misinterpreted as typing or chordic manipulation.

This chapter begins with a discussion of how the identification problem for the MTS relates to previous research on recognition of hand gestures captured by optical sensing systems. Then the chapter presents the algorithms and biomechanical constraints which the MTS utilizes for hand position estimation, finger identification, and hand identification. Finger identification is posed as an assignment

problem which must optimally match finger contacts to a ring of finger attractor points. The identification of single, isolated contacts is solely determined from the weighted Voronoi diagram formed by the attractor points. It is shown that with a contact-attractor distance-squared cost metric, the assignment algorithm effectively sorts multiple hand contacts around the ring with respect to the attractor orderings and inter-attractor angles, regardless of whether the attractor ring is at all centered on the contact cluster. Results will illustrate how intricate feedback between tracking system modules over successive scanning cycles causes quick convergence upon correct identifications for the comfortable range of hand motions.

Remember during the following discussions that the term "fingertip" can refer to any finger except the thumb. Thus the thumb never counts as a "fingertip," though the thumb is considered one of the five fingers. Also, palm heels never count as fingertips or fingers.

4.1 Hand Gesture Recognition

Techniques as diverse as rule-based inference [121], elastic graph matching [146], and Kohonen Feature Maps [16] have been utilized to recognize the free-space hand postures of various sign languages. Because the objective of these systems has been to recognize an alphabet of communicative gestures, most of the systems have only been designed to recognize representative static hand postures, though Wexelblat [163] and Boehm *et al.* [16] have concentrated on recognition of dynamic, continuous gestures sensed by DataGloves [148].

4.1.1 Communicative Gestures versus Manipulative Gestures

In contrast to these communicative gesture recognizers, the primary objective of the MTS is to recognize a variety of simple control gestures for manipulating graphical objects in two or three dimensions. The recognition task can then be separated into three parts: recognizing which graphical manipulation channel the

operator is selecting, tracking the manipulative hand and finger motions, and continuously extracting multiple degree-of-freedom (DOF) object control signals from the tracked motions. Compared to a mouse, recognizing the channel selection corresponds to checking which buttons are being held down, tracking hand motions corresponds to measuring the rotation of the mouse ball, and extracting independent motion components has no analogue since measurements from orthogonal ball rotation sensors are already independent. This chapter will tackle the channel selection and tracking problems in the context of proximity imaging devices by reliably attaching finger and hand identities to the contact paths constructed in the previous chapter. Multi-DOF extraction will not be addressed until Chapter 5.

The graphical manipulation objective places very different requirements on the recognition algorithm than the symbolic communication objective. A central tenet of this dissertation is that operator efficiency in rich graphical environments can increase tremendously by having several manipulation channels which the operator can switch between instantaneously with simple changes in hand configuration. However, not nearly as many channels or distinct hand configurations are necessary for improved graphical manipulation as are necessary to support a symbolic gesture language containing dozens of distinct signs. Operators may only be willing to memorize the hand configurations which select a few different graphical manipulation channels, and it may be hard to memorize mappings for or even imagine uses for more than a dozen channels. Fundamental manipulations such as mouse cursor pointing, mouse cursor dragging, text cursor pointing, text cursor selection, and window scrolling only demand five channels. A few more hand configurations might be utilized to select particular paintbrush, stylus, or eraser tools in drawing programs, avoiding frequent excursions to the drawing tool palette to select different tools. Any remaining hand configurations not needed for graphical manipulation

channels can still be reserved for symbol or command gestures. The MTS can recognize up to eight distinct symbol or command gestures per channel by mapping opposing motions in the rotational, scaling, and two translational DOFs to different symbols or commands.

4.1.2 Locating Fingers within Remote Optical Images

Another important difference between recognition of manipulation versus symbolic gestures is that symbol recognizers only need to discern the pattern of each hand configuration as a whole, while manipulation recognizers must also be able to precisely extract hand motion in several dimensions. To do this, manipulation recognizers must locate and track specific points on the hand such as the fingertips. In the context of passive optical sensing, several researchers have adapted their systems to recognize index finger pointing gestures and track the center of a fingertip. Crowley and Coutaz [30] track a finger on a digital desk by finding the peak in cross-correlation between each video image and a reference fingertip template image.

Ahmad [3] first locates the palm by assuming the palm center is the center of mass of the video image. Then he fits a circle around the palm, finds the center of the wrist from overall hand orientation and the palm-enclosing circle, and applies a Hough transform [9] to the angles with respect to wrist center of pixel groups outside the palm circle. The peaks of the Hough transform histogram then represent the finger angles, and the pixel groups farthest from the palm along these angles are assumed to represent the fingertip locations. Clearly this approach only works for an outstretched hand which approximately faces the camera.

Nolker and Ritter [115] successfully train a local linear mapping network to locate all five fingertips in a wide variety of hand configurations, including when the fingers are curled in touching the palms so that the background of the fingertips

consists of low contrast palm flesh. Nolker and Ritter also successfully extract the pointing direction of an index finger.

4.1.3 The Feasibility of Identification from Proximity Images

The problem of locating and identifying fingertips from proximity image information is substantially different than fingertip location from remote optical images. As described in the previous chapter, accurately measuring contact centroids and tracking contacts across proximity images are relatively easy once the contacts are properly segmented. However, determining which contact comes from which fingertip is greatly complicated by the invisibility of the intermediate finger structure which connects fingertips to the center of the hand.

4.1.3.1 Rubine's Encounter with Finger Identification

The only researcher known to have encountered the finger identification problem for proximity sensing systems is Rubine [130], who was trying to recognize complex multi-path gestures on the Sensor Frame. As an active optical system which sensed obstruction of light beams, the Sensor Frame (see further description on Page 38) [107, 108, 129] suffered the same limitations in detecting intermediate finger structure as finger capacitance sensing systems. Rubine's objective was to recognize complex gestures involving two or three fingers by feeding each finger's path into his path classifier and identifying the gestures by the resulting combination of path classifications. He considered general finger identification infeasible:

> For multi-path input devices which are actually attached to the hand or body, such as the DataGlove, there is no problem determining which path corresponds to which finger. Thus, it would be possible to build one classifier for thumb paths, another for forefinger paths, etc. The characteristics of the device are such that the question of path sorting does not arise.
>
> However, the Sensor Frame (and multifinger tablets) cannot tell which of the fingers is the thumb, which is the forefinger, and so on. Thus there is no *a priori* solution to the path sorting. The solution adopted here was

> to impose an ordering relation between paths. The consistency property is required of this ordering relation: the ordering of corresponding paths in similar gestures must be the same. *Rubine* [130], Page 81.

Rubine resorted to simple sorting of paths according to the temporal and spatial coordinates of their starting points. He claimed this was sufficient for most of the multi-path gestures he was trying to recognize, though it could be confused by similar gestures whose path starting points did not always have the same ordering.

Rubine went on to develop a path clustering technique for complex multi-path gesture recognition which avoided path sorting [130]. This clustering technique computed global features from the sums and differences of all combinations of path pairs. Instead of training a different path classifier for each sorted path, a single classifier was applied to all the global features. During training of the classifier, hierarchical cluster analysis grouped similar feature vectors, irrespective of path sorting.

4.1.3.2 Summary of Constraints on Contact Identity

Upon closer examination, there actually turn out to be quite a few anatomical and biomechanical constraints on the relative features and positions of hand contacts. Many have already been discussed in Section 2.3 on proximity image topology. The challenge will be that sometimes very few constraints are available in the current proximity image or system tracking state. Since some constraints are weak and ambiguous, they can only make up for the invisibility of hand structure when several are combined.

For example, the sizes and orientations of thumb and palm contacts are usually but not always unique and distinguishable from the fingertips. Sometimes the thumb and palm heels are not touching the surface at all and therefore do not appear in the proximity image. Though thumb and palm heels are usually larger than fingertips normal to the surface, they do not become larger instantaneously.

Unless they impact the surface impulsively, the thumb and palm heels can be just as small as normal fingertips for the first few images after touchdown, until their flesh compresses and flattens out. Likewise, if the operator intentionally touches the thumb or palm heel on the surface only lightly, actively suspending their weight, the contacts will never reach a large size. Thumb and palm contact orientations, in turn, cannot be measured reliably until a contact is as big or bigger than a normal fingertip, and these orientations will not always differ from fingertip orientation.

The rest of the constraints apply to inter-contact relationships and cannot simply be measured from geometric features of individual contacts. The interaction of finger joint kinematics and a surface greatly constrains the locations of fingers relative to one another. As is apparent in most of the sample images of Section 2.3, the fingertips tend to settle into a horizontal arc whose radius varies as fingers are flexed and extended (Figures 2.7–2.10). Although it is possible to make some of the fingers flex while the others remain extended, concurrent, uniform flexion as when gripping a foam ball is more natural. Most people can only cross their fingers when the fingers are fully extended, so finger crossover should not occur during typing and chordic manipulation because these activities are normally performed with the fingers partially flexed. The only possibility of partial crossover occurs under extreme rotation of the whole hand in the surface plane. Thus the ordering of fingertip identities within their arc should coincide with the ordering of their horizontal coordinates.

However, the thumb and palm heels can be interspersed nearly anywhere in this horizontal ordering. When considered all together, the thumb, fingertips, and palm heels from one hand tend to be arranged in a circular cluster with a limited radius. As more hand parts touch the surface, more inter-contact relationships become available to evaluate. By the time the whole hand, *i.e.*, five fingers and two palm heels, touches the surface, the inter-contact constraints alone are enough to

reliably identify all of a hand's contacts.

As discussed in Section 2.3.5, the position and rotation of each hand as a whole are also fairly well constrained. Although hand crossover occurs during advanced piano performance, it is hard to imagine how it would be of use in MTS gestures. Therefore hands are not expected to cross over one another, though they may slide to the other side of the surface. A split key layout and a slight arch in the surface about the vertical axis encourage operators to keep the hands well separated. Assuming the operator's torso faces the MTS at a fixed angle, the maximum expected range of hand rotation is about 90°.

A final important observation is that when not actively engaged in graphical manipulation, both hands tend to return to neutral postures with wrists straight and fingers slightly flexed. Thus the variation in posture or deviation from neutral tends to be less at the start of a manipulation than towards the end. A system which can extend throughout a gesture the correct identifications of the initial, neutral posture may get by without needing to identify the extreme, confusing finger arrangements which occur at the end of the gesture.

In summary, the following constraints are *sometimes* available for identification:

- diagonal orientation of thumb and inner palm heel.
- moderate size of flattened thumb contact and large size of flattened palm heels relative to fingertips.
- expected angles and separations between multiple contacts depend on identity of involved hand parts.
- each identifiable hand part can only cause one surface contact.
- crossover or overlap of fingers or hands unlikely.

- range of comfortable hand rotations fairly limited.
- gestures usually start from neutral postures rather than extreme postures.

4.1.3.3 Underconstrained Cases

As this chapter will show, the identification problem is not insurmountable, at least not with the help of the constraints noted above. However, there will still be instances in which the system will not be able to tell fingertips apart, usually because only some parts of a hand are touching the surface, limiting the efficacy of inter-contact constraints. For example, unless the hand is assumed to be hovering over home row, *i.e.*, the default hand position, there is no sure way to tell which fingertip caused an isolated tap on the surface when no other hand parts are touching the surface. However, as long as isolated finger taps are only interpreted as keystrokes on a conventionally distributed key layout, there is no real need to know the finger identity. The intended key symbol should be indicated by the position of the finger tap relative to the key regions in the layout, not by finger identity. Indeed, it would be unnecessarily restrictive to activate keys only if the operator struck them with a particular finger. Only chord typing schemes make rigid associations between single finger identities and key symbols, but in such schemes the hand remains in a relatively static position over the home row keys. Given the reasonable assumption that overall hand positions are fairly fixed during chord typing, the finger identification methods presented in this chapter are accurate enough to support chord typing on the MTS.

Since the actual purpose of finger identification on the MTS is to support chordic manipulation, not chord typing, the contrasting demands each places on an identification system should be further emphasized. Since chordic manipulations will involve slides over the entire surface, chordic manipulations cannot be assumed to start from any one hand position, though they will often start from the neutral

or default hand position. While contact size, orientation, and relative velocity features will be sufficient to distinguish the thumb from fingertips even in the face of such initial hand displacements, these features tend not to differ between fingertips. Therefore, unless all fingertips on a hand are touching the surface, the MTS will have no sure way to tell exactly which fingertips are touching if the fingertips deviate more than a centimeter from the horizontal locations predicted by the hand position estimate. Such deviations occur often enough that the MTS will not be able to reliably distinguish those chordic manipulations consisting of the same number of fingertips but different combinations of them, such as the index and middle fingertips versus ring and pinky fingertips.

4.1.4 Pooling of Fingertip Combinations

Careful design of the chord gesture set in Chapter 5 will pool potentially ambiguous finger combinations so that minor identification failures do not affect the MTS operator. As already discussed, far fewer chords are necessary to cover all conceivable graphical manipulation channels than to cover an entire symbolic alphabet. While the identification system will do its best to identify fingers correctly in all cases, the chordic manipulation recognizer will only depend upon correct detection of thumb presence and proper ordering and counting of the other fingertips. This will support selection of seven different chordic manipulation channels (see Table 5.1 on Page 246) on each hand by multiple-finger chords, as opposed to 26 which would be available on each hand if all combinations of two or more fingers were distinguished.

This pooling of performable fingertip combinations should not be considered an unhappy compromise; consideration of human factors also argues for it. Memorization and control of particular combinations of fingertips rather than particular numbers of fingertips may be just as cognitively and biomechanically demanding for the operator as identification is for the MTS. Given the freedom to choose and vary

which fingertips are used in chords composed of only one, two or three fingertips, operators naturally prefer to pick combinations in which all touching fingertips are adjacent rather than combinations in which a finger such as the ring finger is lifted but the surrounding fingers such as the middle and pinky must touch. In a chord typing study, Fukumoto and Tonomura [41] found that users can tap these finger chords in which all touching fingertips are adjacent twice as fast as other chords. Trained pianists could perform all chords about twice as fast as normal subjects, but awkward chords still took twice as long as chords composed of a contiguous group of fingertips.

Allowing the MTS operator to interchange fingertip combinations when accessing common manipulation channels such as pointing also helps avoid overuse of the finger muscles which press or suspend a particular set of fingers. The uniquely opposable motion of the thumb and relatively large region of sensory-motor cortex devoted to the thumb also suggest that the thumb may be cognitively distinct from the fingertips. The gesture set will abide by this distinction: chords including the thumb will be reserved for selection of command gesture channels, and chords initially composed solely of fingertips will select graphical manipulation channels, though the thumb can be added to these chords after the initial channel selection to capture its unique opposable motions.

4.2 Overview of the Hand Tracking and Identification System

For the reader's convenience, the tracking system flow diagram of Figure 3.1 is repeated in Figure 4.1. The image segmentation module described in Chapter 3 segments the current proximity image into groups of electrodes corresponding to the distinguishable hand parts. The path tracking module, also described in the previous chapter, links the groups from successive proximity images corresponding to the same hand part into persistent contact paths. It also computes parameters

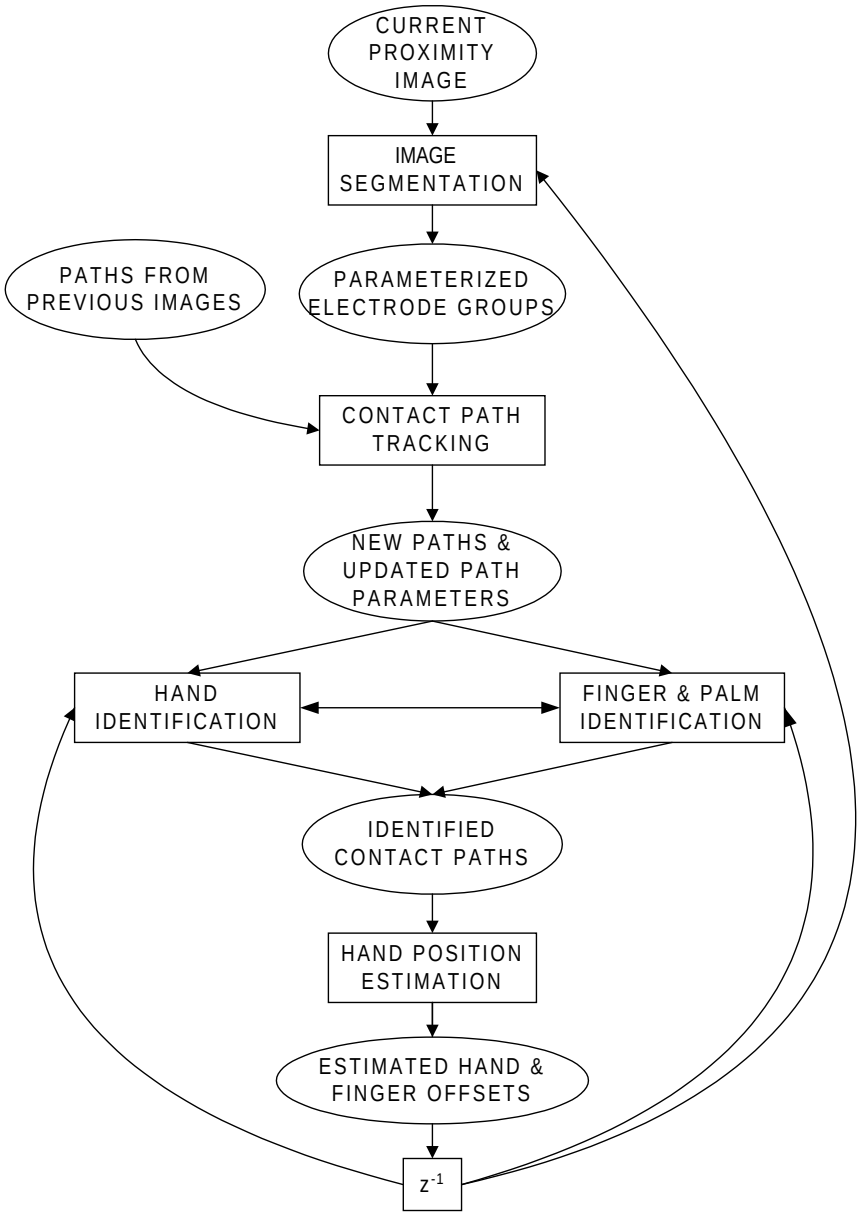


Figure 4.1: System-level diagram for hand and finger tracking and identification modules.

Table 4.1: Finger identity notation for identified path data structures.

Notation	*Common Hand Part Name*
$F0$	The Null/Dummy Hand Part
$F1$	Thumb Finger (not a fingertip)
$F2$	Index Finger
$F3$	Middle Finger
$F4$	Ring Finger
$F5$	Pinky Finger
$F6$	Outer (lateral) Palm Heel
$F7$	Inner (medial) Palm Heel
$F8 \ldots F11$	Forepalm Calluses/Overflow Contacts

related to the contact trajectories such as the instantaneous lateral velocity along each path.

The finger identification, hand identification, and hand position estimation modules are the focus of this chapter. The hand identification module will determine which hand causes each contact, and the finger identification module will establish a unique finger or palm heel identity for each contact within a hand. The finger and hand identification modules are hierarchically related, and both will employ combinatorial optimization methods to find the most biomechanically and anatomically consistent set of contact identifications. The output of the identification modules will be non-zero hand and finger indices attached to all contact paths. The identified contact paths will be referred to with the notation of Table 4.1. To denote a particular hand identity this notation can be prefixed with an L for left hand or R for right hand; for example, RF2 denotes the right index finger path. When referring to a particular hand as a whole, LH denotes the left hand and RH denotes the right hand.

The hand position estimator will use contact positions as well as the assigned contact identities to maintain a conservative estimate of overall hand position even

when a hand is not touching the surface. Feedback of the estimated hand positions to the finger identification process will help identify contacts when so few contacts appear in the image that the hand's contact cluster has no apparent structure. To aid hand identification, the estimated hand position will temporarily retain the last measured hand and finger positions after a hand completely lifts off the surface. Then, if the fingers quickly touch back down in the same region, they will more likely regain their previous identifications. Because this estimated hand position feedback is essential to the accurate functioning of the hand and finger identification algorithms, the hand position estimation algorithm will be described first.

Again, this overall system architecture grossly resembles that of most computer vision systems, with the hand and finger identification modules corresponding to the object recognition stage. In this case, the object recognition stage is more constrained than in most computer vision applications because upper limits are known on the number and type of objects which can appear in the image, *i.e.*, thumb, fingertip, or palm contacts from a left or right hand. Rather than picking the best match from a library of object templates, the object recognition problem reduces to an assignment problem of finding the optimal one-to-one mapping between surface contacts and finger identities. When fewer contacts are present on the surface than possible identities, dummy contacts will be created to keep the mapping one-to-one. Fingers whose identities end up mapped to a dummy contact are assumed to be lifted off the surface.

Unlike the path tracker, which is expected to create and maintain perfect continuity of each finger path from the first image frame in which the finger appears, the identifications are not required to be correct in the first frame. As a hand touches down, its individual parts may gradually appear over several image frames, gradually increasing the constraints available for identification. Given that

the segmentation, identification, and hand position estimation modules may be underconstrained without the feedback between them, they can only be expected to converge on a coherent solution over a few iterations of feedback between one another. Reshuffling of identifications early in a hand gesture is allowed so the system does not commit to an assignment before hand configuration clues have accumulated.

However, reshuffling back and forth during ambiguous cases is still undesirable, so the system must also have some assignment hysteresis. For images which contain no new information pertinent to identification, the identities of existing contacts are simply extended via path continuation. The identification algorithm need not even execute for such images, avoiding the risk that identities will get reshuffled. As soon as a proximity image appears with new constraints such as an additional contact or clearer features, the identification algorithm executes again. As will become apparent, the hand position estimates provide a weaker, analog form of hysteresis while fingers are temporarily lifted off the surface.

4.3 Hand Position Estimation

As indicated in Figure 4.1, the hand position estimator provides important biasing feedback to the identification and segmentation processes. The hand position estimates are intended to be a conservative guess of lateral hand position under all conditions, including when a hand is floating above the surface with no flesh visible in the proximity images. In case a hand is not touching the surface, its position estimate represents a best guess of where it will touch down again. When a hand partially touches the surface, the estimate combines current hand position measurements based upon the centroids and identities of its contacts with previous hand position estimates based upon earlier identifications which may have been more or less reliable than the current measurements.

4.3.1 Measuring Current Hand Position

Figure 4.2 shows the individual steps of the hand position estimation process, which must be repeated for each hand separately. First, an overall hand position must be measured from the contact positions in the current proximity image. The simplest method of obtaining a hand position measurement would be to average the positions of all the hand's contacts regardless of identity. If all hand parts were always touching the surface, as in the flattened hand of Figure 2.7, the resulting centroid would be a decent estimate, lying somewhere under the center of the palm since the fingers and palm heels typically form a ring around the center of the palm. However, consider when only one hand contact is available for the average. The measurement would wrongly assume the hand center is at the position of this lone contact. A lone hand contact is very unlikely to be from the hand center because the hand center usually does not even touch the surface until the rest of the hand is flattened onto the surface. If the lone contact is actually from the right thumb, the true hand center would be 4-8 cm to the right, or if the contact is actually from a palm heel, the true hand center would be 4-6 cm higher, or if the lone contact is from the middle finger, the true hand center would be 4-6 cm lower.

Instead of assuming each contact comes from the center of the hand when computing the average, the MTS's hand position measurement utilizes the within-hand identifications to compute for each contact an offset between its measured position, $(Fi_x[n], Fi_y[n])$, and the default position (Fi_{defx}, Fi_{defy}) of the particular finger or palm heel with its identity i. These default positions correspond to finger and palm positions when the hand is in a neutral posture with fingers partially closed (see Figure 2.8), as when resting on home row of the key layout. *With this identity-dependent offset computation, the position of a single, properly identified surface contact will be sufficient to sustain a fairly accurate hand position estimate.*

The next step averages the individual contact offsets to obtain the measured

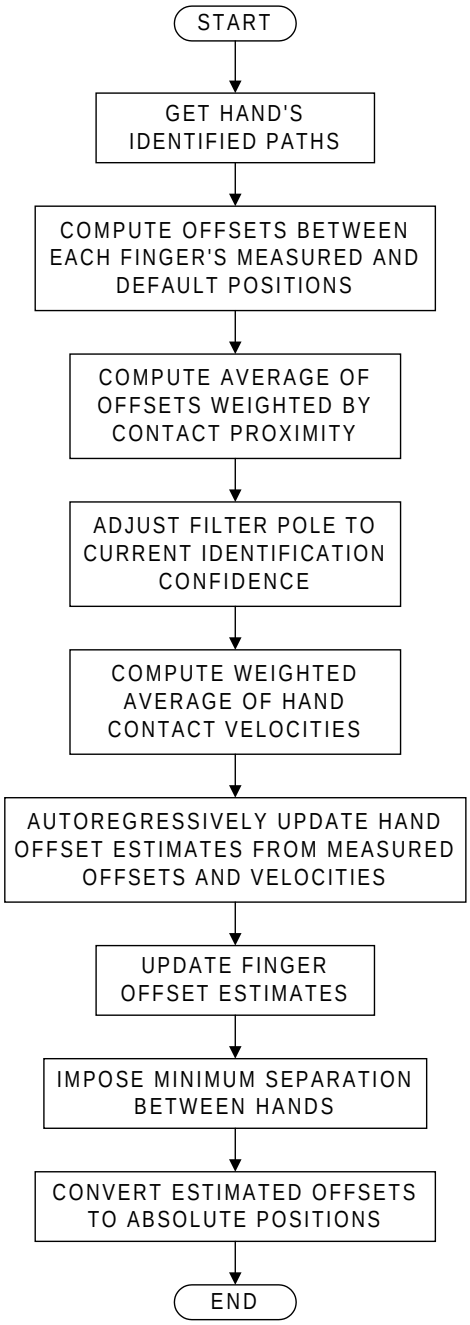


Figure 4.2: Flow chart of hand position estimation process.

hand offset $(H_{mox}[n], H_{moy}[n])$:

$$H_{mox}[n] = \frac{\sum_{i=1}^{i=7} Fi_{mow}[n](Fi_x[n] - Fi_{defx})}{\sum_{i=1}^{i=7} Fi_{mow}[n]} \quad (4.1)$$

$$H_{moy}[n] = \frac{\sum_{i=1}^{i=7} Fi_{mow}[n](Fi_y[n] - Fi_{defy})}{\sum_{i=1}^{i=7} Fi_{mow}[n]} \quad (4.2)$$

Preferably the weighting $Fi_{mow}[n]$ of each finger and palm heel is approximately its measured total proximity, *i.e.*, $Fi_{mow}[n] \approx Fi_z[n]$. This ensures that lifted fingers, whose proximity is zero, have no influence on the average, and that contacts with lower than normal proximity, whose measured positions and identities are less accurate, have low influence. Furthermore, if palm heels are touching, their large total proximities will dominate the average. This is beneficial because the palm heels, being immobile relative to the hand center compared to the highly flexible fingers, supply a more reliable indication of overall hand position.

When a hand is not touching the surface, *i.e.*, when all proximities are zero, the measured offsets are set to zero. This will cause the filtered hand position estimate below to decay toward the default hand position.

4.3.2 Identification Confidence and Filter Delay

As long as the contact identifications are correct, this identification-dependent method for hand position measurement eliminates the large errors caused by assuming lone contacts originate from the center of the hand. Flexing of fingers from their default positions will not perturb the measurement more than a couple centimeters. However, this method is susceptible to contact misidentification. It assumes identifications are always correct, which is not always the case, especially if the identification system only has a couple of contacts to optimize over. For example, if only one hand part is touching the surface and it is assigned the wrong finger or palm identity, the hand position measurement can be off by as much as 8 cm. If the lone contact is attributed to the wrong hand, it can easily cause $H_{mox}[n]$ to be

in error by 20cm. Therefore the current measured offsets are not used directly, but are averaged with previous offset estimates ($H_{eox}[n-1], H_{eoy}[n-1]$) using a simple first-order autoregressive filter, forming current offset estimates ($H_{eox}[n], H_{eoy}[n]$).

The filter pole $H_{o\alpha}[n]$ should be adjusted according to confidence in the current contact identifications. Since finger identifications accumulate reliability as more parts of the hand contact the surface, one simple measure of identification confidence is the number of fingers which have touched down from the hand since the hand last left the surface. Contacts with large total proximities also improve identification reliability because they have strong disambiguating features such as size and orientation. Therefore $H_{o\alpha}[n]$ is set roughly proportional to the sum of contact proximities for the hand:

$$H_{o\alpha}[n] \quad = \quad \min(1, \beta + \frac{1}{7} \times \sum_{i=1}^{i=7} Fi_z[n]) \tag{4.3}$$

$H_{o\alpha}[n]$ must of course be normalized to be between zero and one or the filter will be unstable. Thus when confidence in contact identifications is high, *i.e.*, when many parts of the hand firmly touch the surface, the autoregressive filter favors the current offset measurements. However, when only one or two contacts have reappeared since hand liftoff, the filter emphasizes previous offset estimates in the hope that they were based upon more reliable identifications.

To encourage correct segmentation and identification upon touchdown for a hand which has temporarily lifted off the surface, the filtered offsets must hold a conservative estimate of hand position while the hand is floating above the surface. If a hand lifts off the surface in the middle of a complex sequence of operations and must quickly touch down again, it will probably touch down close to where it lifted off. However, if the operation sequence has ended, the hand is likely to eventually return to the neutral posture, or default position, to rest. Therefore, the β term in Equation 4.3 is made small enough that while a hand is not touching the surface, the estimated offsets gradually decay to zero at about the same rate as a hand lazily

returns to default position. Alternatively, the estimated hand offsets can be made to decay toward zero at a constant speed rather than exponentially.

4.3.3 The Filter Equations

When $H_{o\alpha}[n]$ is small due to low identification confidence, the filter tracking delay becomes large enough to lag behind a pair of quickly moving fingers by several centimeters. The purpose of the filter is to react slowly to questionable changes in contact identity, not to smooth contact motion. Measurement of the current contact velocities $(Fi_{vx}[n], Fi_{vy}[n])$ occurs in the path tracking process (3.3.4) independent of finger identity. Therefore this motion tracking delay can be safely eliminated by adding the contact motion measured between images to the old offset estimate. Again the hand motion $(H_{mvx}[n], H_{mvy}[n])$ is averaged over the individual contact velocities:

$$H_{mvx}[n] = \frac{\sum_{i=1}^{i=7} Fi_{mow}[n] Fi_{vx}[n]}{\sum_{i=1}^{i=7} Fi_{mow}[n]} \quad (4.4)$$

$$H_{mvy}[n] = \frac{\sum_{i=1}^{i=7} Fi_{mow}[n] Fi_{vy}[n]}{\sum_{i=1}^{i=7} Fi_{mow}[n]} \quad (4.5)$$

The estimated hand offsets $(H_{eox}[n], H_{eoy}[n])$ can now be computed using the complete filter equations:

$$H_{eox}[n] = H_{o\alpha}[n] H_{mox}[n] + (1 - H_{o\alpha}[n])(H_{eox}[n-1] + H_{mvx}[n] \triangle t) \quad (4.6)$$

$$H_{eoy}[n] = H_{o\alpha}[n] H_{moy}[n] + (1 - H_{o\alpha}[n])(H_{eoy}[n-1] + H_{mvy}[n] \triangle t) \quad (4.7)$$

The overall filter structure captured by these equations is also illustrated in Figure 4.3.

4.3.4 Enforcing Hand Separation

While the offset computations for each hand have been independent as described so far, it is advantageous to impose a minimum horizontal separation between

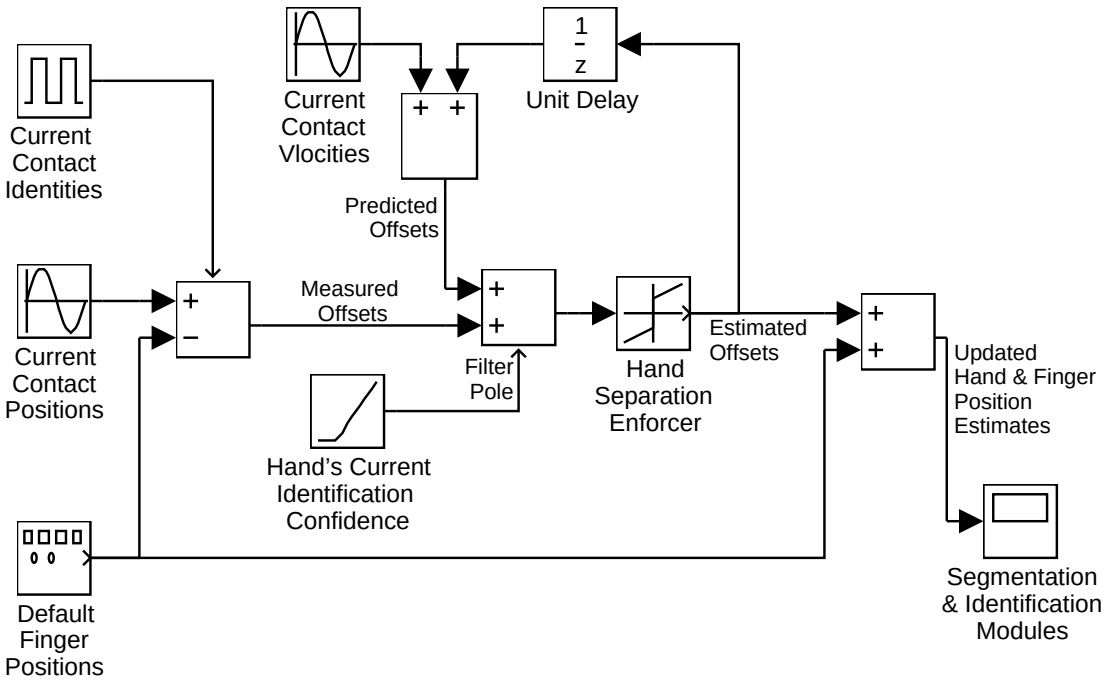


Figure 4.3: Filter diagram for hand position estimator. Note that the default finger positions are subtracted off in the first stage and added back in at the last stage so that the signals at intermediate stages are in the form of relative offsets from default.

the estimated left hand position and estimated right hand position such that when a hand such as the right hand slides to the opposite side of the board while the other hand is lifted, the estimated position of the other hand is displaced. In this case the estimated position of the lifted left hand would be forced from default to the far left of the surface, possibly off the surface completely. If the right hand is lifted and the left is not, an equation like the following can be applied to force the estimated right hand position out of the way:

$$RH_{eox}[n] := \min(RH_{eox}[n], (LF1_{defx} - RF1_{defx}) + LH_{eox}[n] + min\_hand\_sep) \quad (4.8)$$

where $(LF1_{defx} - RF1_{defx})$ is the default separation between left and right thumbs, min_hand_sep is the minimum horizontal separation to be imposed, and $LH_{eox}[n]$ is the current estimated offset of the left hand.

4.3.5 Interactions with Segmentation and Identification Modules

The updated hand position estimates $(H_{eox}[n], H_{eoy}[n])$ are fed back to the segmentation and identification processes during analysis of the next proximity image. If the other processes need the estimate in absolute coordinates, they can simply add the supplied offsets to the default finger positions, but in many cases the relative offset representation is actually more convenient.

The updated hand position estimates tend to move so as to reinforce the current contact identifications. Assuming the current identifications are correct, this tends to stabilize them, to move the attractor ring so that the contacts line up better with their assigned attractors. But it can also reinforce incorrect identifications, which is why it is so important to limit the rate of change in estimated hand position with the filter pole $H_{o\alpha}[n]$ when the confidence in identifications is low. On the other hand, when confidence in identifications is high but the position estimate contradicts them, the position estimate should be allowed to change quickly to become consistent with the identifications.

4.4 Finger Identification

On surfaces large enough for multiple hands, the contacts of each hand tend to form a circular cluster, and the clusters tend to remain separate because operators like to avoid entangling the fingers of opposite hands. Because the arrangement of fingers within a hand cluster is usually independent of the location of and arrangement within the other hand's cluster, the contact identification system is hierarchically split. The hand identification process first decides to which cluster each contact belongs. Then a within-cluster identification process analyzes the arrangement of contacts within each hand's cluster, independent of the other hand's cluster. Because within-cluster or finger identification works the same for each hand regardless of how many hands can fit on the surface, it will be described first. The description below is for identification within the right hand; mirror symmetry must be applied to some parameters before identifying left hand contacts.

4.4.1 The Basic Attractor Ring

For the contacts assigned to each hand, the finger identification process (Figure 4.4) attempts to match contacts to a template of hand part attractor points, each attractor point having an identity which corresponds to a particular finger or palm heel. This matching between contact paths and attractors should be one-to-one, but in the case that some hand parts are not touching the surface, some attractors will be left unfilled, *i.e.*, assigned to dummy paths.

The relative locations of the attractor points are set to the approximate positions of their corresponding fingers and palms when the hand is in the default posture with fingers partially curled (Figure 2.8). These should be the same default finger and palm locations (Fi_{defx}, Fi_{defy}) employed in hand position estimation. The default fingertip positions should also match the centers of the home row keys in the key layout. Setting the distances and angles between attractor points from the half-closed hand posture causes the attractors to lie in a ring with a radius

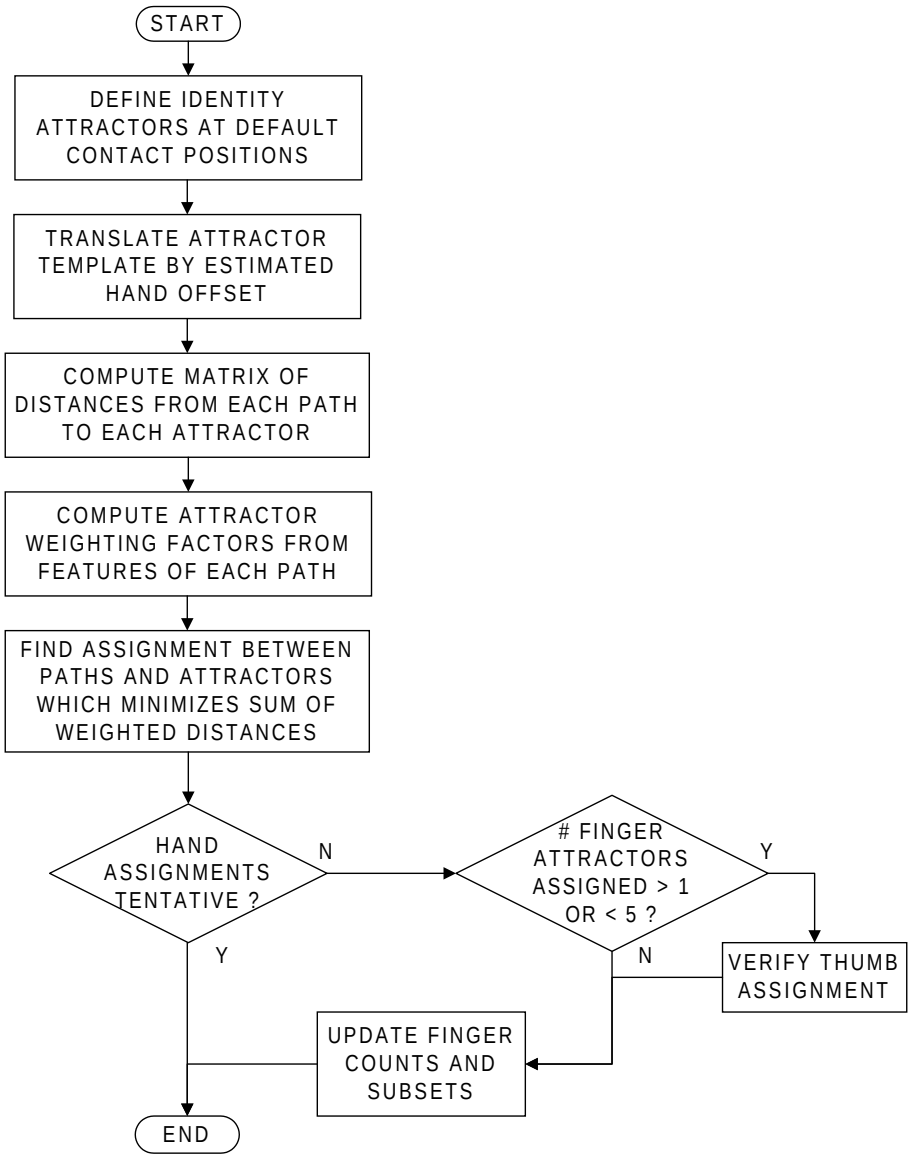


Figure 4.4: Flow chart of the finger and palm (within-hand) identification algorithm.

about halfway between that of an outstretched, flattened hand and a fist. This will allow the matching algorithm to perform well for a wide variety of finger flexions and extensions.

For optimal accuracy of contact-attractor matching, the ring should be kept roughly centered on the hand cluster. Therefore, the attractor ring for a given hand is translated as a whole by the hand's estimated position offset. The final attractor positions $(Aj_x[n], Aj_y[n])$ are then:

$$Aj_x[n] = H_{eox}[n] + Fj_{defx} \tag{4.9}$$

$$Aj_y[n] = H_{eoy}[n] + Fj_{defy} \tag{4.10}$$

4.4.2 Voronoi Diagram for Single Contact Identification

Figure 4.5 displays both the ring-like structure formed by the attractor points for the right hand and the Voronoi polygon or cell around each attractor. If the given hand is a left hand, the attractor ring must be mirrored about the vertical axis from that shown. Every geometric point within an attractor's Voronoi cell is closer to that attractor than any other attractor in the ring [116]. When there is only one contact in the hand cluster and its features are not distinguishing, *i.e.*, when only a single small part of a hand is touching the surface, the identification algorithm can simply determine which Voronoi cell the contact lies within and assign the contact to that cell's attractor. Thus the size and shape of each Voronoi cell indicates the range over which a particular hand part can touch down with respect to estimated hand center and still be identified correctly. Given that the Voronoi cells for fingertips are rather tall and narrow, one can conclude that the Voronoi cells will tolerate a high degree of finger flexion and extension but not so much hand rotation or unexpected horizontal displacement.

In the unweighted Voronoi diagram of Figure 4.5, the palm attractors are actually a couple centimeters lower than the measured default palm heel positions.

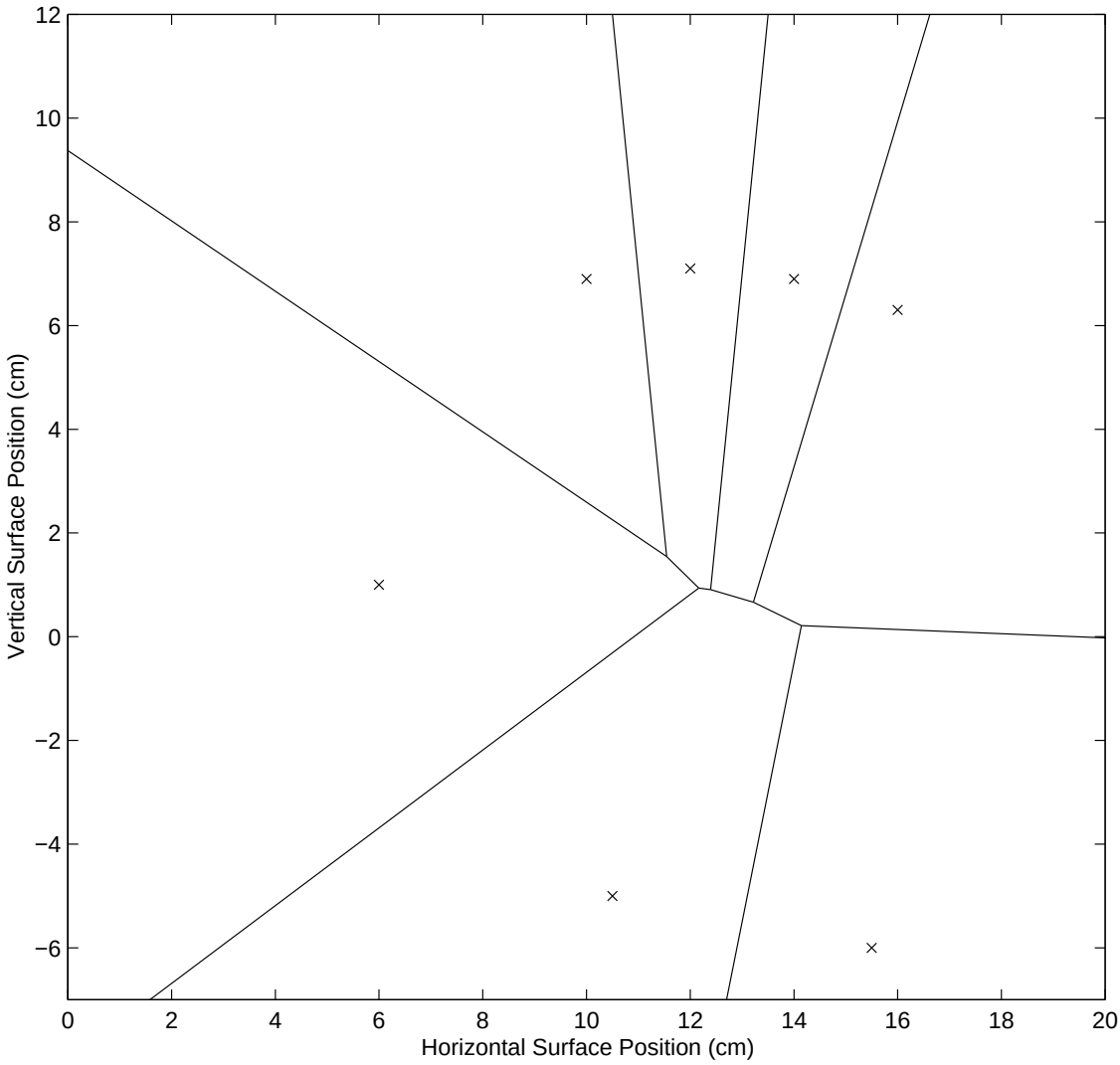


Figure 4.5: Voronoi cell diagram constructed around ring of hand part attractor points (x's labeled with finger identity indices). In this figure, the estimated right hand position offsets are zero, so the ring is not translated from the default finger positions.

Moving these attractors forward to their proper vertical positions of -4 cm would have enlarged the palm heel Voronoi cells too much at the cost of thumb and pinky Voronoi cell size. The palm heel Voronoi cells are so large that an abducted thumb or flexed pinky is occasionally misidentified as a palm heel. This can actually cause typing errors when hitting pinky keys in the lowest row such as $< Ctrl >$ or $< Alt >$ (see Figure 1.1 on Page 6) because the tapping finger can be misidentified as a palm and therefore ignored. Section 4.4.6.2 will introduce palm heel weightings which shrink the palm heel Voronoi cells, allowing the true default palm heel positions to be used.

4.4.3 Multiple Contacts Compete for Voronoi Cells

When multiple parts of a hard touch the surface, more than one may lie within the same Voronoi cell. *Since only one of the contacts can be assigned to any Voronoi cell's attractor at one time, the contacts lying in the same cell must compete for the cell's attractor and neighboring attractors.* A global optimization is necessary to determine which of the contacts goes to the attractor of the occupied Voronoi cell and which goes to neighboring attractors.

This global optimization finds the one-to-one assignment between attractors and contacts which minimizes the sum of weighted, squared distances between each attractor and its assigned contact. When there are fewer surface contacts than attractors, *i.e.*, when any hand parts are floating above the surface, the null path $P0$, which has zero distance to each attractor, acts as a dummy contact in place of any missing contacts. This ensures a one-to-one mapping can be found by making the total number of surface plus dummy contacts the same as the number of attractors. Let the squared distances in the surface plane between each contact path Pi and each translated attractor point Aj form a square matrix $[d_{ij}^2]$:

$$d_{ij}^2 = (Aj_x[n] - Pi_x[n])^2 + (Aj_y[n] - Pi_y[n])^2 \tag{4.11}$$

The optimization can then be stated as finding the permutation $\{\pi_1, \ldots, \pi_7\}$ of integer hand part identities $\{1, \ldots, 7\}$ which minimizes:

$$\sum_{i=1}^{7} d_{i\pi_i} \tag{4.12}$$

where contact i and attractor j are considered assigned to one another when $\pi_i \equiv j$. This combinatorial optimization problem, known more specifically in mathematics as an assignment problem, can be efficiently solved by a variety of well-known mathematical techniques. The following sections will review the solution techniques for the assignment problem, the reason the squared Euclidean distance is preferred over other distance metrics for finger identification, and how feature-dependent weightings of particular contact-attractor distances sustains correct identification over wider ranges of finger motion.

4.4.4 The Assignment Problem

The assignment problem is a special case from the classes of linear programming problems and integer programming problems. Its more general formulation as a linear programming problem [35] is to minimize over the parameters x_{ij} the sum:

$$\sum_{i=1}^{M} \sum_{j=1}^{M} c_{ij} x_{ij} \tag{4.13}$$

subject to the constraints:

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad \forall i = 1, ..M \tag{4.14}$$

$$\sum_{i=1}^{M} x_{ij} = 1 \qquad \forall j = 1, ..M \tag{4.15}$$

$$x_{ij} = 0 \quad or \quad 1 \qquad \forall i, j = 1, ..M \tag{4.16}$$

where c_{ij} is an arbitrary cost of assigning contact i to attractor j, and contact i is considered assigned to attractor j only when $x_{ij} = 1$. Since there are $M!$ possible solution matrices x_{ij}, brute force enumeration is inappropriate unless M is very

small. Since this is a linear programming problem, it also has a dual formulation as a maximization problem, and specialized versions of the primal and dual simplex method can find the optimal solution [8, 10, 109]. However, the specially constrained structure of the solution matrix x_{ij} supports other efficient solution techniques.

Kuhn [83] was the first to develop the Hungarian Method for solving the assignment problem. This method was based on special matrix properties discovered by Egerváry [34] and König [80] which guide manipulations of matrix rows and columns similar to Gaussian elimination. Its worst case performance is $O(M^3)$, though performance varies widely for large M depending on implementation tricks [27, 102]. The Hungarian Method and improvements upon it have been the preferred solution method in the field of operations research, where the assignment problem often arises when trying to match workers' various skills to a variety of tasks or machines requiring different skills. In most of these applications the cost matrix is not derived from the distances between points in a plane. However, some operations research problems do construct a cost matrix from a set of matching distances. An example would be trying to assign M taxis to M passengers given the location where each passenger is to be picked up, the starting location of each taxi, and the assumption that the overhead for each taxi increases in proportion to the distance from last drop off (starting location) to next pick up. In these real-world operations research problems the unsquared rather than squared distance is usually the more relevant cost parameter.

As an integer programming problem, the assignment problem can be solved with the branch and bound heuristic [58], relaxation [11, 12] or network flow minimization techniques based upon the Shortest Augmenting Path Method [7, 31, 33, 75]. As a combinatorial optimization problem, localized [1] combinatorial search heuristics can be applied, though these do not guarantee the global minimum will be found. For the relatively small size of the finger assignment problem, any of these

solution techniques should be efficient enough for real-time finger identification.

Localized combinatorial search was the first technique implemented in the MTS software, and once methods to avoid convergence failures for it were understood (see Appendix C), there seemed to be no need to try an alternate implementation with any of the other well-known techniques. Moreover, localized combinatorial search has a couple advantages within the context of real-time finger identification. First, it can verify very quickly that a previous set of identifications is still at least locally optimum. Though its worst case performance will turn out to be $O(M^3)$ as well, it finds the global minimum fairly fast given an initialization near the global minimum. Second, it offers important insights into the design and analysis of the attractor ring when assignments costs are proportional to contact-attractor distances squared.

4.4.4.1 Localized Combinatorial Search

Combinatorial problems such as the assignment problem can be incrementally optimized using k-exchange neighborhoods [1]. In the case of the finger identification problem, a k-exchange neighborhood is a subset of attractors from the attractor ring. If k is 2, the subset will usually be a pair of adjacent attractors. Minimizing the sum of assignment distances within this subset by conditionally swapping the two contact-attractor assignments will always improve the total assignment sum. This is a consequence of the fact that the total cost, *i.e.*, the assignment sum, is a monotonic increasing function of the individual costs, *i.e.*, contact-attractor distances. Picking a sequence of k-exchange neighborhoods, *i.e.*, successive adjacent attractor pairs, and optimizing them with conditional swapping causes the total assignment sum to decrease toward a local minimum. Whether the local minimum found is actually the global minimum depends on the initial assignments and the ordering of the exchange neighborhood sequence.

4.4.4.2 Choosing Initial Assignments

After each sensor array scan, the paths of hand parts which have newly touched down are assigned to the closest available attractors in the ring. This operation is equivalent to picking a new path, constructing a Voronoi diagram from only the unfilled attractors, and assigning the path to the unfilled attractor whose Voronoi cell the contact lies within. This does not ensure that the initialization puts the new contact with the best attractor since the best attractor might be one that is already filled with a pre-existing path, but this usually puts the new path close to its correct attractor and therefore close to the global minimum.

4.4.4.3 The Swapping Condition

Let Aa and Ab be adjacent attractors in the ring, *i.e.*, $a = b \pm 1$, and let Pg and Ph, correspondingly, be their currently assigned contact paths. The one-to-one nature of the matching assignments is enforced with the double-links $Aa_{assignedpath} \Longleftrightarrow Pg_{assignedfinger}$ and $Ab_{assignedpath} \Longleftrightarrow Ph_{assignedfinger}$. These assignments are swapped, making $Aa_{assignedpath} \Longleftrightarrow Ph_{assignedfinger}$ and $Ab_{associated} \Longleftrightarrow Pg_{assignedfinger}$, if swapping reduces the sum of the contact-attractor distances:

$$d_{ha}^2 + d_{gb}^2 \stackrel{?}{<} d_{hb}^2 + d_{ga}^2 \tag{4.17}$$

In the method of simulated annealing, a noise term is added to the right side of inequality 4.17 so that some swaps are taken even when they do not decrease the assignment sum. In large problems this allows the optimization search to jump back out of local minima. The noise variance or annealing temperature is decreased over time to lock in the global minimum. As Appendix C shows, there are simpler ways than simulated annealing to avoid non-global minima in finger identification.

4.4.4.4 The k-exchange Sequence

The exchange neighborhood sequences utilized by localized combinatorial search heuristics are often application specific [1]. For finger identification, the roughly circular structure of the attractor ring supports a simple nearest neighbor traversal around the ring, as would a linear structure. The sequence simply proceeds around the attractor ring in either clockwise or counter-clockwise order, pausing for a conditional swap at each attractor encountered. Each swap test is applied between the current attractor and the next adjacent attractor around the ring. Travel around the ring repeats until one complete traversal is made without accepting any swaps.

This basically amounts to a bubble sort on the ring, utilizing the swapping condition for the current attractor pair as the bubble sort ordering relation. Notice that though a bubble sort has $O(M^2)$ performance [79], *i.e.*, M traversals of the ring are required in the worst case, a single traversal of the ring can verify that the sorting of contacts based on their positions in previous proximity images is still the correct sorting for their current positions. Also, the initialization of new contacts assignments as described in Section 4.4.4.2 tends to put contacts within a couple attractors of their proper attractor, so usually not more than two or three traversals of the ring are necessary. Worst case performance can be improved by utilizing a bidirectional bubble sort [79], *i.e.*, reversing the direction of the exchange sequence after each complete traversal of the ring.

Because transitivity of the ordering relations described in the next section will not always extend between opposite sides of the ring, contacts can get stuck in attractors on the opposite side of the ring. Such convergence failures are explained further in Appendix C. They can be avoided and the global minimum reached by expanding the exchange neighborhoods to include contact pair swaps between attractors on opposite sides of the ring instead of only between adjacent attractors on the ring. This expansion of the exchange neighborhood reduces worst case

performance to $O(M^3)$.

4.4.5 Geometric Interpretations of the Swapping Condition

This section explores geometric interpretations of the pair swapping condition of Equation 4.17 which aid in design of the attractor ring and understanding of its assignments.

4.4.5.1 Geometric Interpretation of Single Contact Swapping

The simplest case is when one of the contacts, say Ph, is a dummy or null contact. By definition of dummy contact, $d_{ha} = 0$ and $d_{hb} = 0$. Then the swapping inequality reduces to:

$$d_{gb}^2 \stackrel{?}{<} d_{ga}^2 \tag{4.18}$$

which is equivalent to:

$$d_{gb} \stackrel{?}{<} d_{ga} \tag{4.19}$$

since all distances are non-negative. As shown in Figure 4.6, the geometric interpretation of this swapping condition is that after the conditional swap, real contact Pg will be assigned to the closest attractor, *i.e.*, the attractor which is on the same side of the perpendicular bisector between the attractors as Pg is. The walls of Voronoi cells are actually composed of such perpendicular bisectors between various attractors, Figure 4.6 being the special case of a Voronoi diagram of only two attractors. The fact that Equation 4.18 is equivalent to Equation 4.19 simply means that squaring the distances has no impact on the swapping condition a dummy contact is involved; Voronoi diagrams constructed for a Euclidean distance *squared* metric are the same as Euclidean distance metric diagrams [116].

Another interpretation will be useful for comparison to the case when the attractors compete for two contacts. By projecting the contact g onto the line $\overline{ab}$ between the attractors, one can see that the swapping condition depends on the

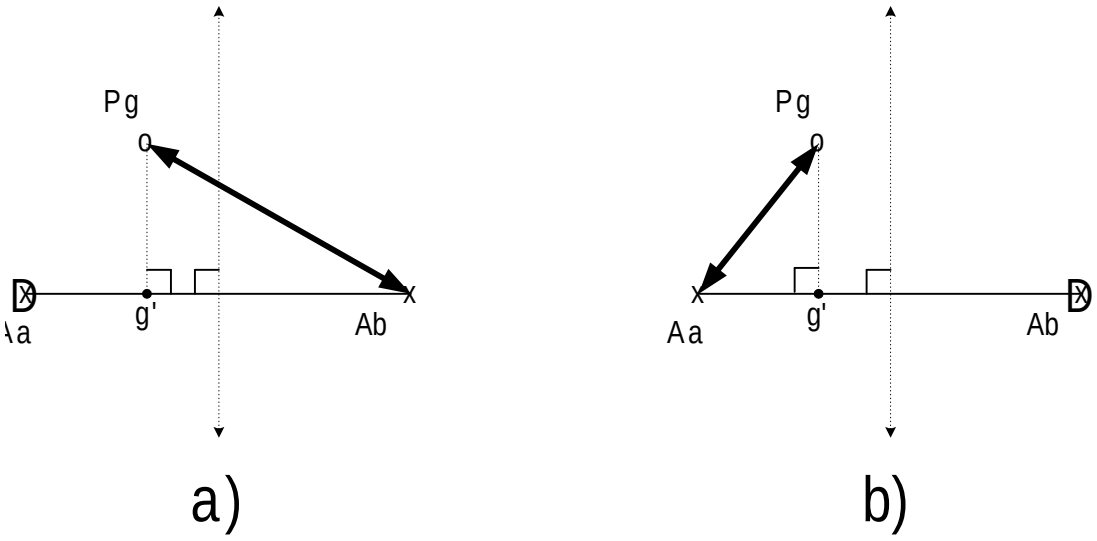


Figure 4.6: Geometric construction showing possible assignments (heavy arrows) when two attractors (crosses) compete for one real surface contact (circle). The losing attractor is always assigned a dummy contact (D) which has zero distance to every attractor. The swapping condition prefers case b) in which the assignment link does not cross the perpendicular bisector (dotted arrows) of the attractors.

horizontal position of the projection g' relative to the center point between the attractors:

$$g'_x \stackrel{?}{<} \frac{(a_x + b_x)}{2} \tag{4.20}$$

The comparison of the contact projection with absolute attractor pair position in Equation 4.20 should be contrasted to the ordering relation which will arise for contact pairs in Equation 4.35. Note Equation 4.35 does not depend upon absolute horizontal alignment of the attractor pair.

4.4.5.2 Geometric Interpretation of Contact Pair Swapping

Though Figure 4.6 and its generalization to Voronoi diagrams explain swapping behavior when two or more attractors are only competing for one real surface contact, the case when two attractors are competing for two contacts, as when the attractor ring is full of contacts, is not as straightforward unless a Euclidean distance squared metric is used. With the distance-squared metric, the swapping decision depends only on the ordering of the contacts as projected onto the line connecting the attractor pair.

This interesting property of the distance-squared metric can be proved with the help of the geometric constructions in Figure 4.7. Without loss of generality, assume attractors a and b lie along a horizontally oriented line. Now, form perpendiculars from each contact g and h to the line $\overline{ab}$ and mark the projected intersections g' and h', respectively. The swapping Inequality 4.17 can be restated in terms of the lengths of the vectors connecting pairs of these points:

$$||\vec{ah}||^2 + ||\vec{bg}||^2 \stackrel{?}{<} ||\vec{ag}||^2 + ||\vec{bh}||^2 \tag{4.21}$$

where the left side of the inequality represents the sum of the squared assignment lengths (heavy arrows) of Figure 4.7a, and the right side of the inequality represents the sum of the squared lengths for the opposite assignments in Figure 4.7b. By the

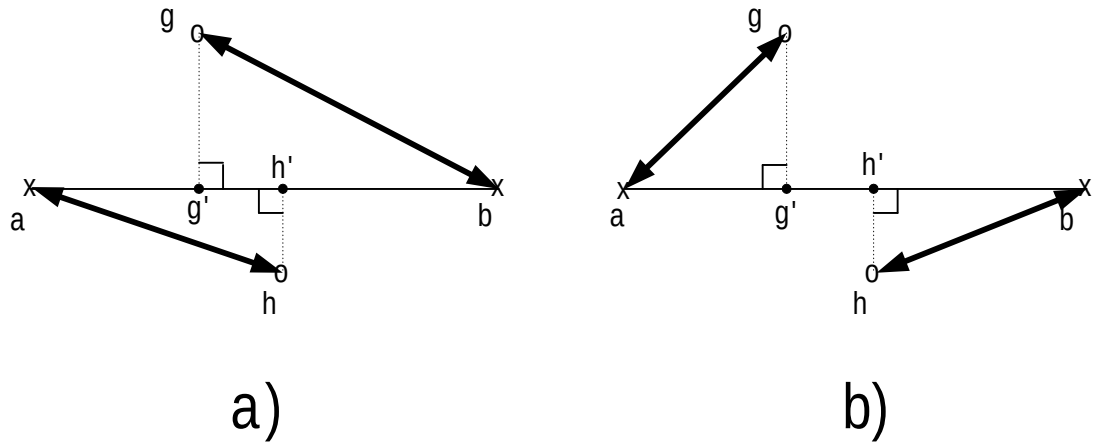


Figure 4.7: Geometric construction for comparing the costs of the two possible assignments (heavy arrows) between a pair of contacts (circles) and a pair of attractors (crosses). If squared-distance is the assignment cost metric, the swapping decision reduces to comparing the ordering of the contacts' projections (dotted perpendiculars) onto the (solid) line between the attractors. For the relative contact positions shown, the assignments of b) have lower cost than those of a) under the distance-squared metric.

Pythagorean theorem, these potentially diagonal vector lengths can be restated in terms of purely horizontal or vertical vector components:

$$||\vec{ag}||^2 = g'a_x^2 + g'g_y^2 \tag{4.22}$$

$$||\vec{bh}||^2 = h'b_x^2 + h'h_y^2 \tag{4.23}$$

$$||\vec{ah}||^2 = h'a_x^2 + h'h_y^2 \tag{4.24}$$

$$||\vec{bg}||^2 = g'b_x^2 + g'g_y^2 \tag{4.25}$$

and substituted into the inequality to obtain:

$$h'a_x^2 + h'h_y^2 + g'b_x^2 + g'g_y^2 \stackrel{?}{<} g'a_x^2 + g'g_y^2 + h'b_x^2 + h'h_y^2 \tag{4.26}$$

The vertical distances on each side cancel, leaving an expression that depends only on the horizontal separations of the contacts relative to the attractors:

$$h'a_x^2 + g'b_x^2 \stackrel{?}{<} g'a_x^2 + h'b_x^2 \tag{4.27}$$

Since $g'a_x^2$ and $h'a_x^2$ can be rewritten in terms of $h'b_x$, $g'b_x$, and ab_x:

$$g'a_x^2 = (ab_x - g'b_x)^2 \tag{4.28}$$

$$= ab_x^2 - 2ab_x g'b_x + g'b_x^2 \tag{4.29}$$

$$h'a_x^2 = (ab_x - h'b_x)^2 \tag{4.30}$$

$$= ab_x^2 - 2ab_x h'b_x + h'b_x^2 \tag{4.31}$$

substituting these expansions into Equation 4.27 and simplifying produces:

$$-2ab_x h'b_x \stackrel{?}{<} -2ab_x g'b_x \tag{4.32}$$

Assuming attractor a is to the left of attractor b so that ab_x is negative, and substituting $h'b_x = h'_x - b_x$ and $g'b_x = g'_x - b_x$, the swap condition reduces to an ordering relation of the contact coordinates as projected onto the line between the attractors:

$$g'_x \stackrel{?}{<} h'_x \tag{4.33}$$

This simple condition ensures that regardless of the contacts' orthogonal displacement from the line connecting attractors a and b, if contact g as projected onto line $\overline{ab}$ is to the left of the projection h' of contact h, contact g gets assigned to the left attractor a, and contact h gets assigned to the right attractor b. For the relative positions of h and g shown in Figure 4.7, Equation 4.33 is not true, and the swap to the assignments of Figure 4.7a is not taken because the existing assignments of Figure 4.7b already minimize the total distance.

Note that though the simplification to Equation 4.33 depends on the assumptions that the attractors lie on a horizontal line and attractor a is to the left of attractor b, the ordering relation is independent of the horizontal alignment of the attractor pair or the separation between the attractors. Rotation of the coordinate space trivially extends the result to attractor pairs lying on a non-horizontal line. This ordering relation can therefore provide translation and scale invariance to identification of multiple finger contacts.

For comparison to the case when the attractor pair competed for only one real contact, Figure 4.8 contains examples of two contact arrangements with the relevant bisectors shown. In Figure 4.8a and b, both contacts are between the attractors, but in Figure 4.8c and d one contact is to the left of the leftmost attractor. In addition to the perpendicular bisector $B_{ab\perp ab}$ of $\overline{ab}$, a bisector $B_{gh\perp ab}$ of the segment $\overline{gh}$ between the contacts is constructed *perpendicular to* $\overline{ab}$ in each case. Again, the distance-squared swapping criterion (Equation 4.17) ensures that if the contact-attractor links cross the contact bisector as in Figure 4.8a or c, the assignments will be swapped so that the contact to the right of the contact bisector $B_{gh\perp ab}$ always ends up assigned (Figure 4.8b or d) to the attractor to the right of the attractor bisector $B_{ab\perp ab}$, and the contact to the left of the contact bisector $B_{gh\perp ab}$ always ends up assigned to the attractor which is left of the attractor bisector $B_{ab\perp ab}$.

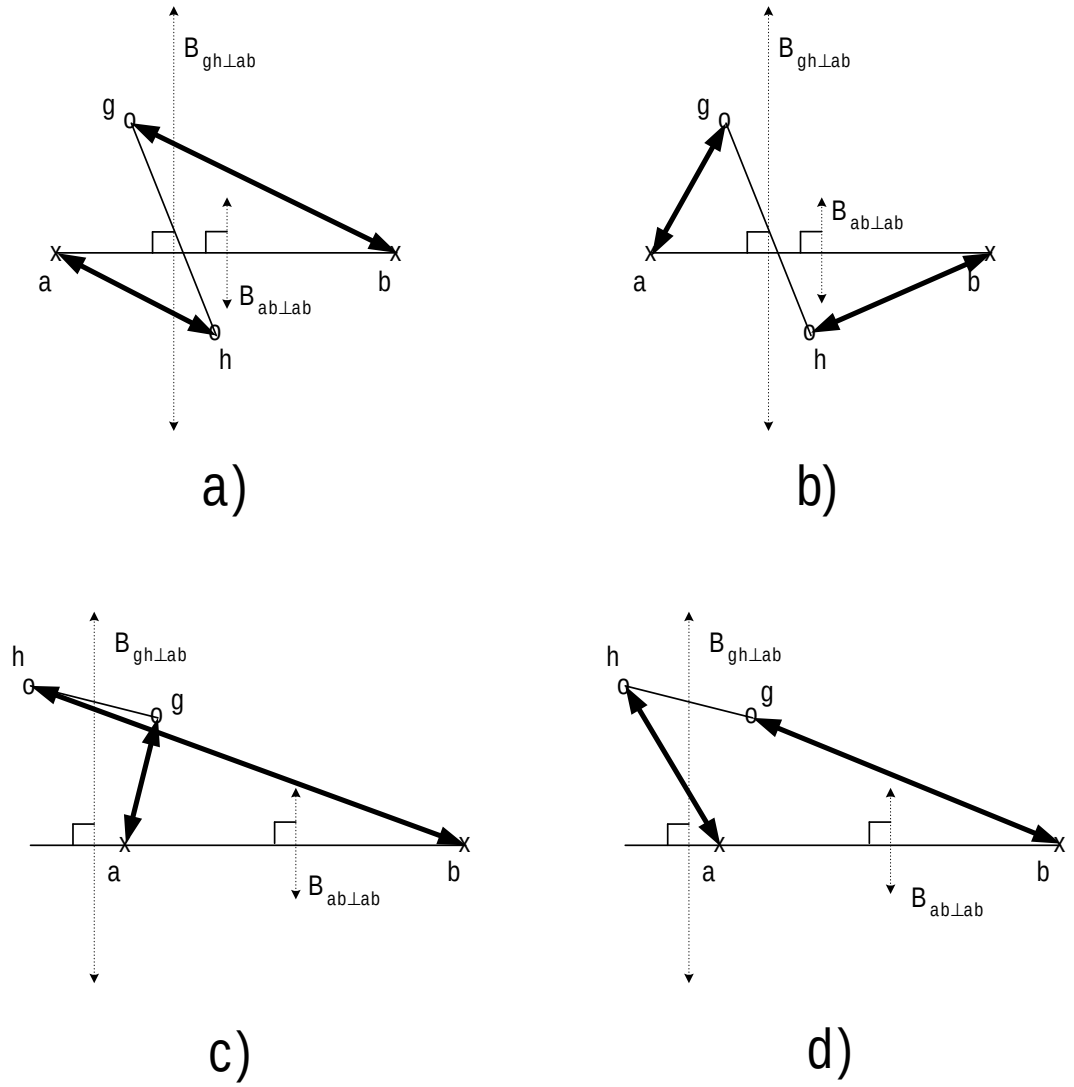


Figure 4.8: Visual comparison of distance-squared assignments via contact pair and attractor pair bisectors (dotted arrows) which are both perpendicular to the segment between the attractor (solid line). The assignments (heavy arrows) in b) and d) have lower distance-squared cost because they preserve the projected ordering. In other words, the contact to the left of the contact bisector, $B_{gh\perp ab}$, goes with the attractor to the left of the attractor bisector, $B_{ab\perp ab}$, and vice versa, regardless of the contact pair's displacement relative to the attractor pair.

4.4.5.3 Summary of Swapping Behavior using Distance-Squared Metrics

In summary, swapping behavior under distance-squared cost is quite different depending on whether the two attractors compete for one or two surface contacts.

- In the one surface contact case, the real contact simply goes to the closest attractor according to the absolute position of the bisector between attractors:

$$g_x \overset{?}{<} \frac{(a_x + b_x)}{2} \tag{4.34}$$

  A dummy contact is assigned to the farther attractor.

- With two competing surface contacts, the contacts are ordered with respect to the inter-attractor angle, but the absolute position of the attractor pair does not matter:

$$g_x \overset{?}{<} h_x \tag{4.35}$$

4.4.5.4 Contact Pair Swapping Behavior with Other Metrics

For metric spaces in which the Euclidean distance between each contact and attractor is taken to some power other than 2, the swapping condition does not simplify so nicely. For example, if the distance power is one, the inequality corresponding to Equation 4.27 includes length product terms:

$$ag_x^2 + bh_x^2 + 2||\vec{ag}||\,||\vec{bh}|| \overset{?}{<} ah_x^2 + bg_x^2 + 2||\vec{ah}||\,||\vec{bg}|| \tag{4.36}$$

which persist through the simplification:

$$||\vec{ag}||\,||\vec{bh}|| - ||\vec{ah}||\,||\vec{bg}|| \overset{?}{<} ab_x(g'b_x - h'b_x) \tag{4.37}$$

This suggests that the swap decision depends on the difference in the products of the link distances as well as the contact ordering as projected onto $\overline{ab}$.

Nevertheless, a couple special cases can illustrate the substantially different swapping behavior which arises with distance powers other than two. Consider

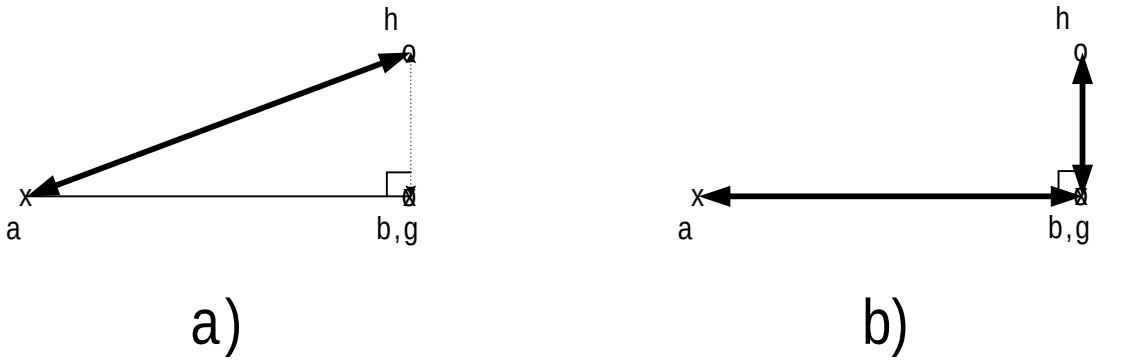


Figure 4.9: Special case when one contact (g) is perfectly aligned on an attractor (b) so that the attractor pair (crosses) and contact pair (circles) form a right triangle. Note the $\vec{bg}$ link in a) is invisible and has zero cost. Under the unsquared-Euclidean metric, a) has a lower total assignment cost than b). With the distance-squared metric, the assignment costs of a) and b) are the same, and with the distance-quadrupled metric, b) has the lower total cost.

Figure 4.9, in which one contact g is right on top of one attractor b, and the other h lies on the perpendicular passing through the same attractor b. Thus the contacts and attractors form a right triangle with both contact g and attractor b at the perpendicular corner, making $||\vec{bg}|| = 0$. The triangle inequality ensures that for the unsquared-Euclidean distance metric, the hypotenuse $||\vec{ah}||$ is shorter than the sum of the lengths of the sides $||\vec{ag}|| + ||\vec{bh}||$, so the assignments will be $g \Longleftrightarrow b$ and $a \Longleftrightarrow h$ as in Figure 4.9a.

For the distance-squared case, the Pythagorean theorem ensures that $||\vec{ah}||^2 = ||\vec{ag}||^2 + ||\vec{bh}||^2$. Therefore neither Figure 4.9a nor Figure 4.9b is preferred since they both produce the same total assignment costs. In other words, both contacts project onto $\overline{ab}$ at the same point, so their order must be chosen arbitrarily.

For the Euclidean-distance-quadrupled metric, the Pythagorean theorem can be invoked again to prove that Figure 4.9b has the lower assignment cost:

$$||\vec{ah}||^4 \;=\; (||\vec{ag}||^2 + ||\vec{bh}||^2)^2 \tag{4.38}$$

$$= \quad ||\vec{ag}||^4 + 2||\vec{ag}||^2||\vec{bh}||^2 + ||\vec{bh}||^4 \tag{4.39}$$

$$< \quad ||\vec{ag}||^4 + ||\vec{bh}||^4 \tag{4.40}$$

Note that in this case contact g gets assigned to attractor a even though its distance to attractor b is zero!

Another important case occurs when all attractors and contacts are collinear yet the contact pair is shifted away from the attractor pair as in Figure 4.10. Under

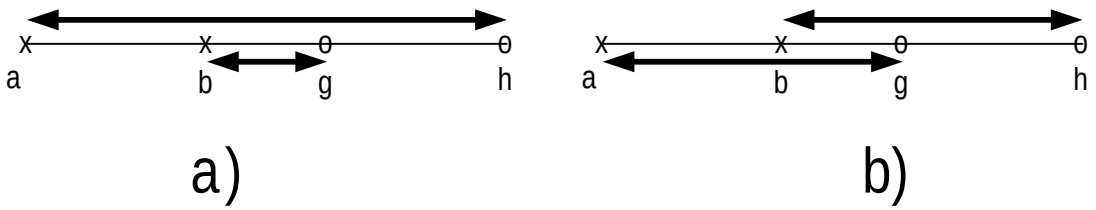


Figure 4.10: Special case when the attractor pair (crosses) and contact pair (circles) are collinear illustrates that unlike the $L2$ metric, the $L1$ metric does not preserve the contact ordering under lateral translation of the contact pair. Under the unsquared-Euclidean metric, the sum of the lengths of the assignment arrows is the same in a) and b) even though the assignments (heavy arrows) of a) reverse the horizontal ordering of the contacts with respect to their assigned attractors. The distance-squared metric produces a lower total cost for more uniform arrow lengths as in b). This causes the horizontal ordering of the contacts to be consistent with that of the attractors under arbitrary horizontal translation of the contact pair.

the unsquared-Euclidean metric, the total of the assignment arrow lengths is the same in Figure 4.10a as in Figure 4.10b even though the lengths within Figure 4.10a are disparate and reverse the horizontal ordering of the contacts with respect to the attractors. The sum of squared lengths favors the more uniform arrow lengths of Figure 4.10b because the long assignment in Figure 4.10a between contact a and attractor h produces a disproportionately large cost.

This preference for more uniform combinations of lengths or costs is one reason squared error is chosen as an error metric for a wide variety of optimization

problems. This section has shown that in the context of an assignment problem, the preference for uniform lengths has the additional consequence of picking the permutation which preserves ordering even when the cluster of contacts is translated with respect to the attractors. This is particularly beneficial for finger identification because the hand position estimates will sometimes be very wrong, causing total misalignment of the attractor ring with the cluster of hand contacts.

4.4.5.5 Distance-Squared Assignment as Sorting

In analogy to Figure 4.10, consider the case when a row of four fingertip contacts is horizontally misaligned with a row of four collinear fingertip attractors. Since the distance-squared swapping condition for any contact pair and attractor pair reduces to an ordering relation (Equation 4.35) on the contact horizontal coordinates, an appropriate exchange neighborhood sequence will sort the contacts with respect to the attractor ordering, producing the same result as a conventional sorting algorithm [79] applied only to the horizontal coordinates of the fingertips. Sorting under distance-squared assignment also tolerates scaling of fingertip spacings as easily as straight horizontal coordinate sorting.

However, assignment to the attractor ring using the distance-squared metric is more general than either horizontal coordinate sorting or assignment using the unsquared Euclidean metric. Recall that with the distance-squared contact pair swapping condition, the contacts are ordered according to their projection onto a line whose angle matches the angle between two given attractors. Thus when three or more attractors are not collinear, the ordering relation changes to fit the angle of the local attractor pair, providing sorting along arbitrary arcs or around a ring. Technically, the pairwise ordering relations will cease to be transitive when the attractors are not collinear, *i.e.* $f < g$ and $g < h$ no longer implies $f < h$. Though this creates the possibility of sorting failures as discussed in Appendix C, the sorting effect still works as long as the change in adjacent attractor angles along the arc is

fairly gradual. Though the unsquared-Euclidean metric can also sort along arcs or around the ring when the ring is perfectly aligned with the contacts, the example of Figure 4.10 shows that the unsquared-Euclidean metric has no preference for the proper fingertip ordering once the attractor ring becomes misaligned by more than one attractor spacing.

Unfortunately, these translation-invariant sorting properties of distance-squared assignment are only effective when the attractor ring is full or nearly full of contacts, *i.e.*, when most of the hand parts are touching the surface. On the edges of a cluster of only two to four finger contacts, empty attractors will compete for real contacts, potentially giving up dummy contacts. As was shown in Section 4.4.5.1, when two attractors compete for one real contact, the swapping condition is equivalent for both squared- and unsquared- distance metrics and offers no special protection against attractor ring misalignment. The absolute position comparison of Equation 4.34 causes dummy contacts to propagate toward those attractors which are farthest from the real contacts, leaving erroneous shifts in fingertip assignment, but the contact pair ordering relation still holds within groups of adjacent attractors retaining real surface contacts, ensuring that the fingertips are at least ordered correctly.

4.4.5.6 Analyzing Swaps on the Attractor Ring

When applied to an attractor ring which is full of fingertips, the geometric interpretations of the contact pair swapping conditions indicate exactly how much local deviation in finger arrangement will be tolerated before finger identities are erroneously swapped. For example, consider the attractor ring and five fingers in Figure 4.11. The fingers are arranged with the same relative angles as the attractors except the whole hand is translated about three centimeters toward the upper right with respect to the attractor ring. Since the contact pair swapping condition for the distance-squared metric is invariant to translation, and since the contact pairs

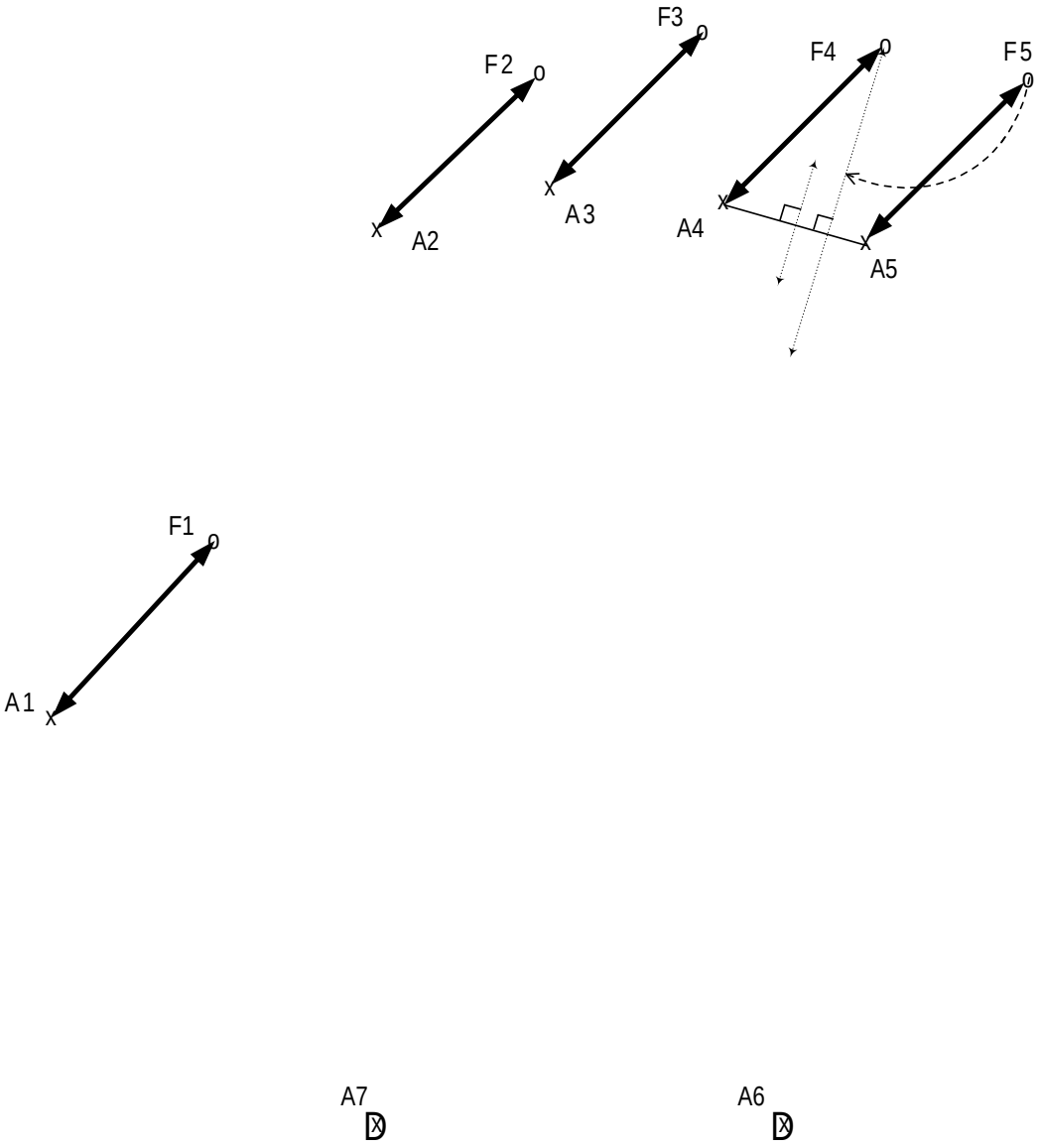


Figure 4.11: Tolerance of hand translation and finger pair deviation in assignment of five fingers to an attractor ring. Under the distance-squared metric, the correct assignments (heavy arrows) of five fingers will be produced for any translation. Furthermore, up to 90° of rotation of the pinky (F5) about the ring finger (F4) from the default angle is guaranteed to be tolerated. Erroneous identity swapping commences when the angle of the perpendicular bisector (long dotted arrow) between A4 and A5 is reached. Even though the palm heels are not touching, the open (or dummy-filled) palm attractors are too far down to receive any finger contacts.

constrain one another within the arc of finger attractors, the exchange sequence will find the correctly ordered finger assignments as shown.

Now suppose the pinky finger (F5) contact begins to rotate clockwise around the ring finger (F4). The pinky's identity will be maintained correctly for a rotation of up to 90° from the default ring-pinky angle. After a rotation of 90° both the ring (F4) and pinky (F5) contacts will be aligned on their bisector, perpendicular to the segment joining the ring (A4) and pinky (A5) attractors and parallel to the Voronoi cell wall (perpendicular bisector of (A4-A5) segment) between the ring and pinky attractors. In other words, the ring (F4) and pinky (F5) contact projections onto the ring-pinky attractor segment will be the same point, and their identities may swap unstably. Any rotation past 90° would definitely cause the ring fingertip (F4) to be assigned to the pinky attractor (A5) and the pinky fingertip (F5) to be assigned to the ring attractor (A4).

Thus *even in the face of attractor ring misalignment, the tolerance for finger pair crossover can be determined precisely.* Basically, the identifications are most stable when the inter-contact angles match the inter-attractor angles. Identities of two fingers will begin to be swapped erroneously as the angle between them approaches the angle of the perpendicular bisector between their attractors, which is usually available from the Voronoi diagram. With the unsquared-Euclidean metric, this rule would only hold when the contacts were roughly centered between their attractor pair on the attractor bisector. Given the amount of translation in Figure 4.11, the Euclidean metric probably would not produce the proper finger ordering even without finger pair rotation.

To understand the identifications which will result when only two to four fingers are touching the surface, leaving some finger attractors open, one must combine the geometric interpretation of assignment of one contact to an attractor pair (Section 4.4.5.1, Figure 4.6, Equation 4.20) with that of contact pair assignment

(Section 4.4.5.2, Figure 4.8, Equation 4.35). Since the single contact swapping condition is not as tolerant of translation as the contact pair swapping condition, the identifications will become more and more vulnerable to corrupting translations as more attractors are left unfilled. In general, finger identifications can be erroneously shifted to the right or left by as many attractors as the number of open finger attractors (5 - number of touching fingers) under sufficient horizontal translation. Finger contacts are rarely shifted into open palm attractors (Figure 4.11) because the palm attractors are so far below, meaning erroneous identification would require a vertical misalignment of more than five centimeters. Also, palm heels have fairly unique features which will be used to discriminately weight assignment to them in Section 4.4.6.

Suppose the pinky finger contact (F5) is removed from Figure 4.11 and a dummy contact (D) is assigned to the pinky attractor (A5) in its place. This situation is shown in Figure 4.12. As soon as the swapping test is applied to the ring (A4) and pinky (A5) attractor pair, the ring finger (F4) will be assigned to the pinky attractor (A5) since it is closer to the pinky attractor, *i.e.*, on the right side of the perpendicular bisector of the attractor pair. The ring attractor (A4) will receive the dummy contact (D). When swap testing proceeds to the A3-A4 attractor pair, A3 will receive the dummy contact (D) and A4 will receive F3 for similar reasons. After swap testing of the A2-A3 pair, the dummy contact (D) will arrive at A2, the index finger attractor, and stay there as shown in Figure 4.13. It will stay because for the given amount of hand translation, the thumb (F1) is still closer to the thumb attractor (A1) than the index finger attractor (A2). In other words, no real contacts lie in the index finger Voronoi cell, so the dummy contact remains there.

Notice that though the assignments of the three remaining fingertips have been erroneously shifted to the right, the distance-squared contact pair swapping condition will still maintain their proper ordering. The ordering is still governed by

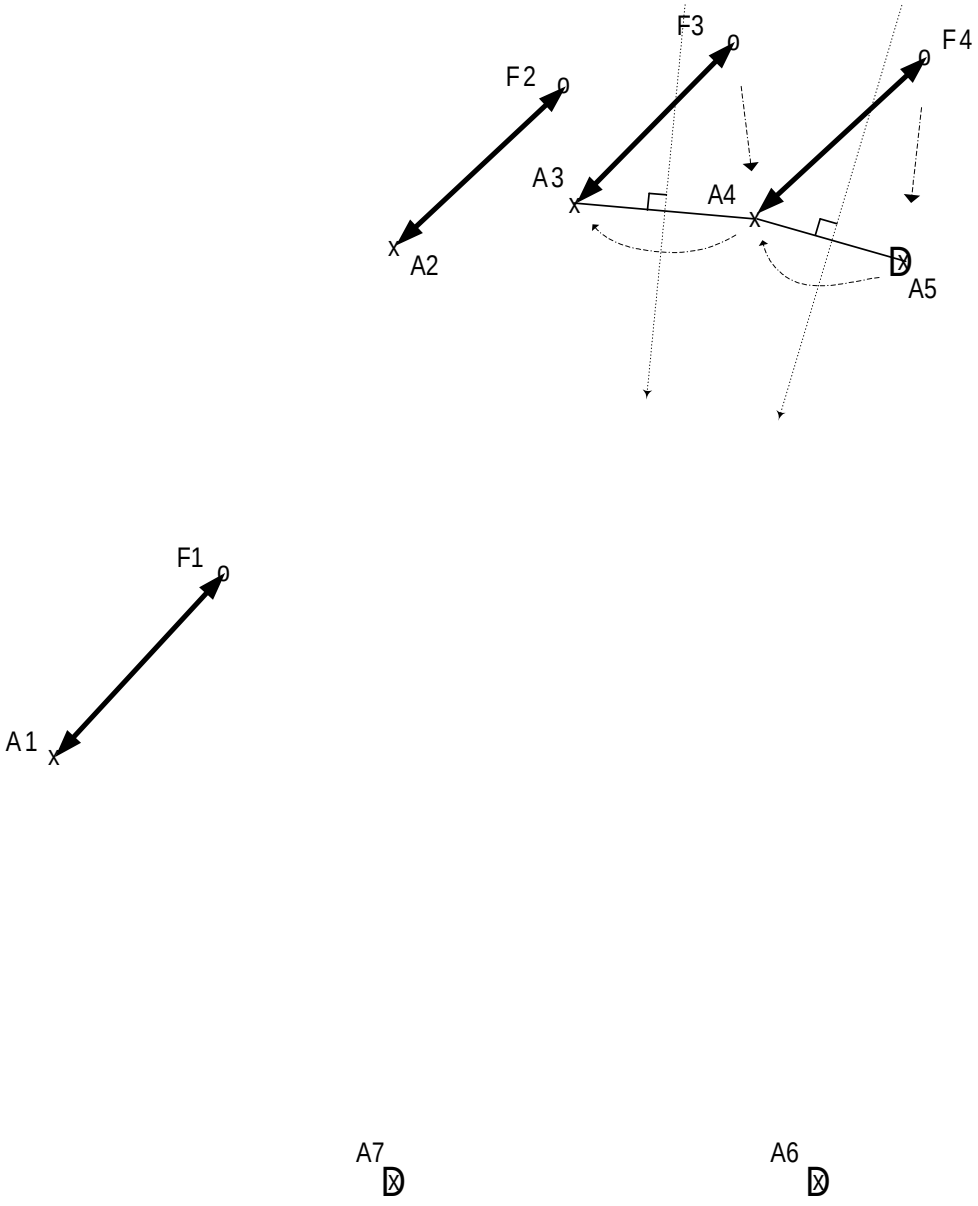


Figure 4.12: Identity swaps which occurs after the pinky finger contact is removed and replaced with a dummy contact (D). The assignments of the remaining fingertip contacts will shift right due to the hand translation as the dummy contact propagates toward the index finger attractor.

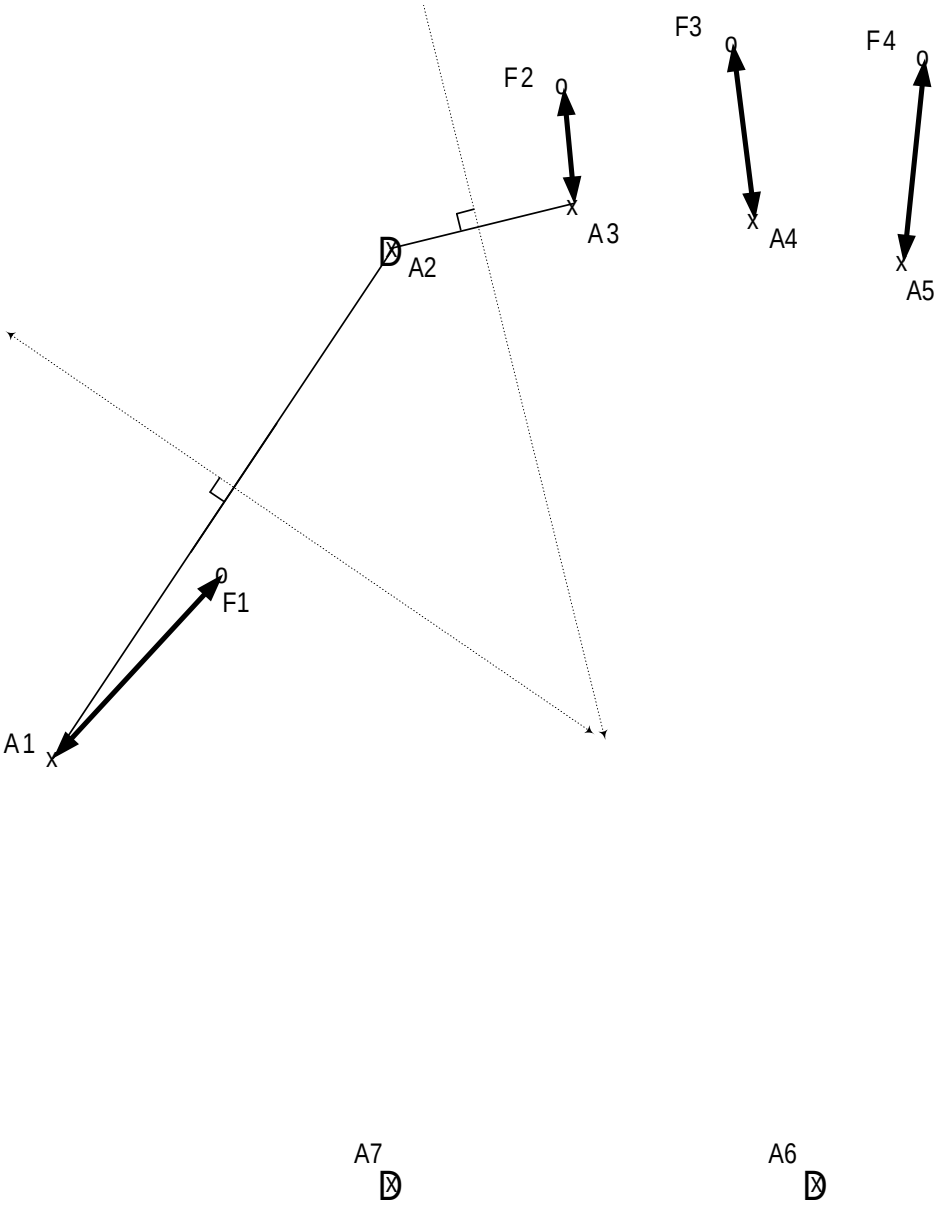


Figure 4.13: The dummy contact (D) propagates to the index finger attractor when the pinky finger is removed. It remains there because no real surface contacts lie in the index finger Voronoi cell.

inter-contact angles in the same way as in Figure 4.11, except that in this case the middle (F3) and ring (F4) finger identities will be swapped if the ring finger rotates more than 90° past the angle between the ring (A4) and pinky (A5) attractors.

4.4.6 Tuning the Attractor Ring with Weighted Voronoi Diagrams

As presented so far the attractors have been unweighted, which causes the Voronoi cells formed from perpendicular bisectors to be shaped like polygons. Multiplicatively weighting the distances to particular attractors causes the equidistant bisectors between contacts with different weightings to be circles instead of straight lines [116]. The center of these *Apollonius* circles lies along the line between the two attractors but to the outside of the attractor with the smaller weight [116]. Weighted Voronoi cells can therefore be composed of arcs as well as straight lines, and the *Apollonius* circles of attractors with large weights can contain "holes" caused by attractors with tiny weights [116]. By convention, the distances are divided by the weightings w_{ij}, *e.g.* $c_{ij} = d_{ij}^2/w_{ij}$, so that as the weighting of an attractor gets smaller, the attractor's Voronoi cell does too, and vice versa.

Static weightings of particular attractors can warp and resize the Voronoi cells to better match the range of motion of each hand part. For instance, since the thumb and pinky have a much wider range of motion than the palm heels, it will be advantageous to shrink the palm heel Voronoi cells and expand the thumb and pinky Voronoi cells into space vacated by the palm cells.

Dynamic weightings of particular attractors in proportion to the strengths of distinguishing contact features will also enlarge or shrink certain Voronoi cells in relation to those of neighboring attractors. Attractor points whose Voronoi cells appear dynamically enlarged to contacts with appropriate features are more likely to be assigned those contacts even in the face of large attractor ring alignment errors. While individual attractors can be weighted independently and the attractor ring as

a whole can be translated from the default hand position, note that the positions of individual attractors with respect to the rest of the ring never need to be changed.

4.4.6.1 Constant Additive Weighting to the Distance Matrix

Before applying any multiplicative weights to the contact-attractor distances, a constant additive weight of about 2 cm^2 is added to each distance matrix entry, d_{ij}^2. Since this offset is applied uniformly to every distance, it has no effect when assignments are unweighted except adding $2M$ to all assignment sums. Thus it does not affect the structure of the unweighted Voronoi diagram. Its purpose will be to ensure dynamic contact feature weightings are effective even when a contact is precisely on top of an attractor, that is when $d_{ij}^2 \approx 0$. This effectiveness depends on the fact that in such "compound-weighted" Voronoi diagrams, the combination of the constant offset and a tiny weighting can cause a Voronoi cell to disappear entirely [116]. Likewise, the combination of the constant offset and a huge weighting can cause the Voronoi cell of one attractor to take over the entire surface.

To illustrate this, suppose a contact's features are inconsistent with an attractor, causing its feature weighting w_{ij} for that attractor to be small, yet the contact lies right on top of the attractor. Without the constant distance offset, the weighted distance will still approach zero, *i.e.*, $d_{ij}^2/w_{ij} \approx 0$, not reflecting the feature mismatch. With the constant offset, the weighted distance $(d_{ij}^2 + 2)/w_{ij} \approx 2/w_{ij}$ will always reflect the weighting somewhat, even as the Euclidean distance goes to zero. The choice of 2 cm^2 reflects both empirical testing and the argument that since attractor ring alignment errors average a centimeter or two and fingers have ranges of motion of several centimeters from their default positions, precise alignment of a contact over an attractor is more happenstance than a strong indicator of contact identity.

What does it mean for a Voronoi cell to shrink and vanish? If the Voronoi cell is from the diagram of the entire attractor ring, it simply means that a sole

hand contact will not be assigned to the vanished cell's attractor, regardless of the contact's position on the surface. However, when multiple contacts are competing for attractors, an attractor whose cell has vanished from the Voronoi diagram can still receive a contact simply because all other nearby attractors may already be assigned to other contacts.

4.4.6.2 Static Palm Heel Weightings

In the unweighted attractor ring of Figure 4.5 on Page 141, the palm attractors had to be placed a couple centimeters lower than the measured default palm heel positions. Moving these attractors forward to their proper vertical positions of -4 cm would have enlarged the palm heel Voronoi cells too much at the cost of thumb and pinky Voronoi cell sizes. Since the thumb and pinky have much wider ranges of motion with respect to hand center than the palm heels, it would make much more sense for the palm heel Voronoi cells to be more compact than the thumb and pinky cells.

The MTS achieves compact palm heel Voronoi cells by including a small weighting on all squared distances between contacts and palm heel attractors. This weighting has been empirically chosen to be .25, *i.e.*, $w_{i6} = w_{i7} = .25$. Since the weights modify the squared distance, this effectively doubles the unsquared Euclidean distance to a palm heel attractor compared to distances to other attractors. As shown in Figure 4.14, this causes the bisectors between the inner palm heel and thumb and between the outer palm heel and pinky to contract into circles around the palm heels. Since the relative weightings between the palm heels are the same, the bisector between inner and outer palm heels continues to be a perpendicular line. Because the weightings limit the lateral and upward extent of the palm heel Voronoi cells, with the weightings it is safe to move the palm heel attractors forward to their actual measured default positions, as is done in Figure 4.14.

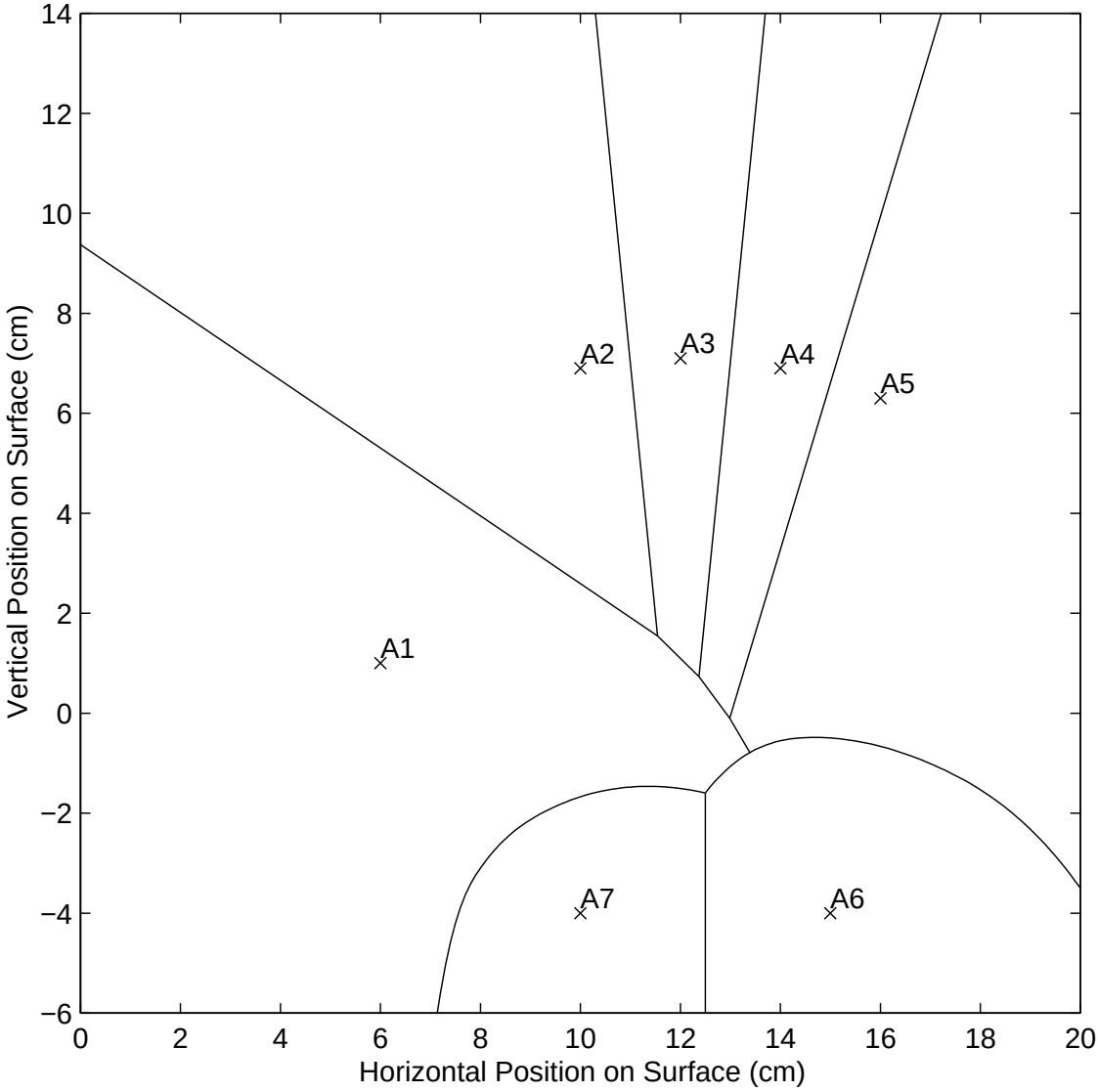


Figure 4.14: Voronoi diagram with distances from contacts anywhere in the plane to palm heels weighted to be twice as far as normal.

With these weighted palm heel attractors, the weighted Voronoi cells match the range of motion of each finger quite well. This is shown by the finger motion trajectories superimposed on the weighted Voronoi diagram in Figures 4.15 and 4.16. Figure 4.15 contains the entire range of finger flexion and extension, from outstretched hand to fist, for the author's hand. At the end of the trajectories the fingertips are actually curled under so the knuckles begin to touch the surface. The palms remain fixed on the surface throughout the flexing.

Note that each finger remains within its Voronoi cell over the entire motion range. This demonstrates that the isolated touchdown of any finger anywhere along its flexion range will result in correct identification as long as the horizontal alignment of the attractor ring is correct. The horizontal alignment as determined by the hand offset estimates will generally be perfect as shown when the hand is centered on its default position or if both palm heels are resting anywhere on the surface. If some other fingers are or were recently touching the surface and were properly identified, the horizontal offset estimate will still be within a couple centimeters of the correct alignment but probably will not be quite as good as shown. When all four fingertips are touching the surface, the global assignment optimization will find their correct identities by exclusion from the thumb and palm attractors, regardless of attractor ring alignment.

Figure 4.16 demonstrates the superb fit of the weighted Voronoi diagram when one finger at a time sweeps out its circular range of motion while all others remain resting in their default positions. Note how the circular sweeps of the thumb, index, and pinky fingers are closely circumscribed by their Voronoi cells. Though the hand started in default position, the author did not look at the Voronoi diagram while sweeping out the circles, so it is surprising that the finger sweeps so closely match a Voronoi diagram constructed only from default finger positions and empirically determined palm heel weightings.

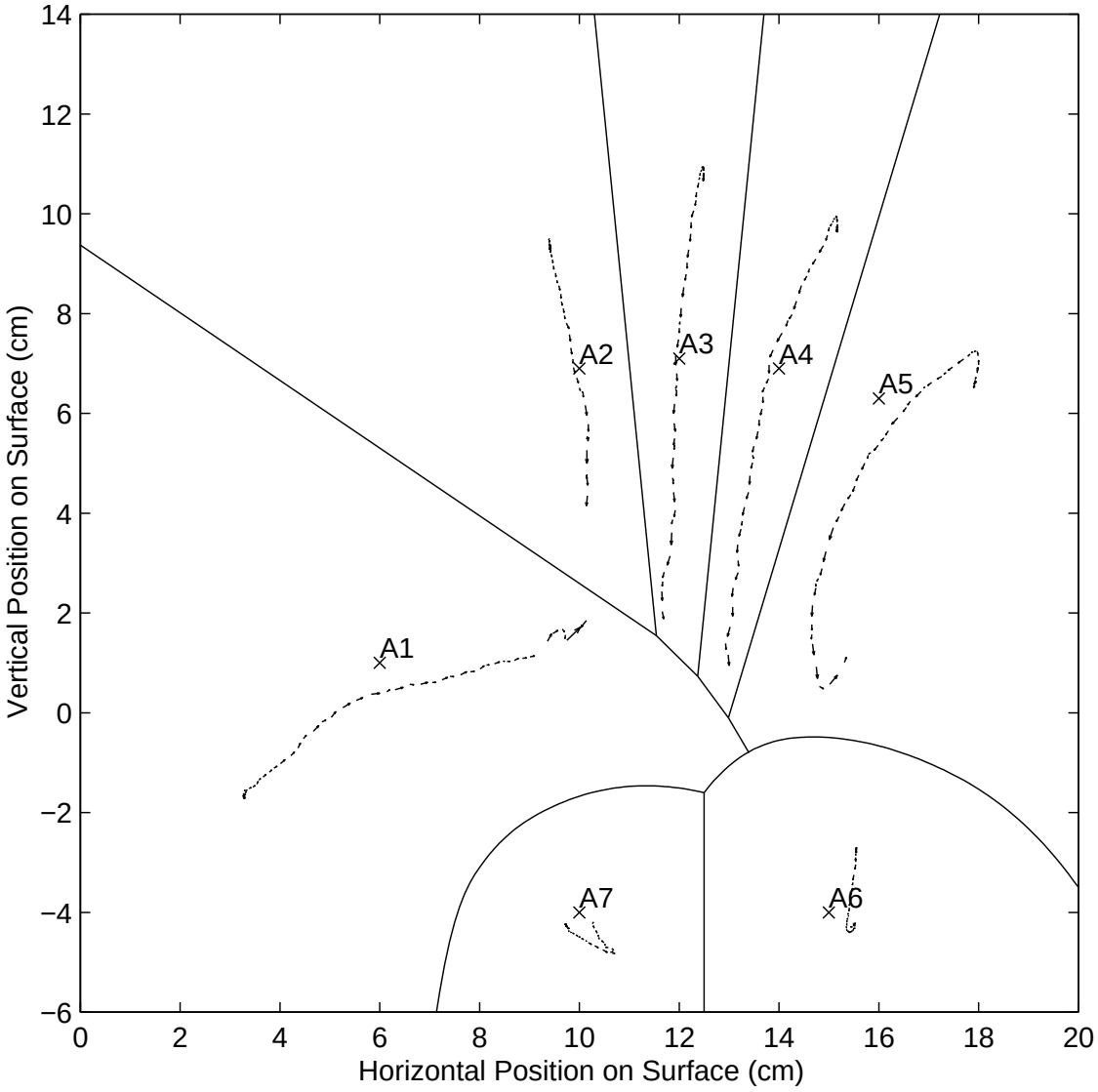


Figure 4.15: Weighted Voronoi diagram with flexing finger trajectories (tiny arrows) superimposed. The fingers of the author's hand started fully extended and outstretched and then flexed simultaneously into a fist. Note that the palm heels remain stationary over their attractors and each finger remains within its Voronoi cell over the entire motion range.

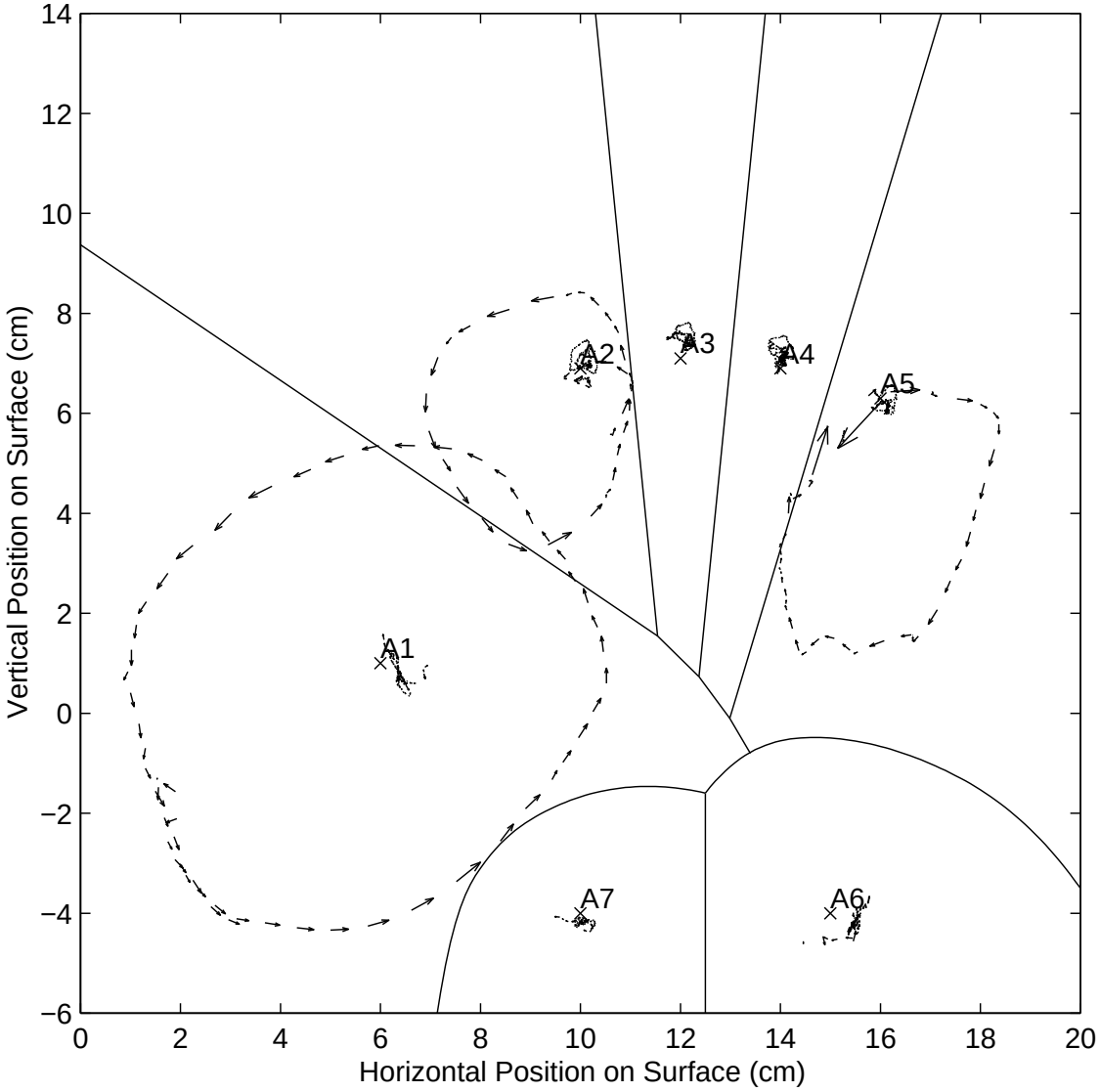


Figure 4.16: Weighted Voronoi diagram with rotating finger trajectories (tiny arrows) superimposed. All fingers of the author's hand started in their default positions right over the attractors. One at a time, the thumb, index and pinky fingers were picked up and made to sweep out their range of motion while the other fingers remained resting in their default positions. Note that the palms were allowed to lift off the surface and shift laterally to capture the full thumb range, but the palms remained on the surface at all other times. Because the middle and ring fingers have little room to move laterally while their adjacent fingers are resting, they were left out of the sweeping experiment.

4.4.6.3 Dynamic Feature Weightings

The dynamic contact-attractor distance weightings depend on whether the geometric features of the given contact match those expected from the hand part that the attractor represents. Since the thumb and palm heels exhibit the most distinguishing geometric features, weighting functions are computed for the thumb and palm heel attractors, and distances to fingertip attractors are unchanged. Each weighting function is the product of several factor versus feature relationships. Each weighting factor is designed to take on a default value of 1 when its feature measurement provides no distinguishing information, take on larger values if the measured contact feature uniquely resembles the given thumb or palm heel, and take on smaller values if the measured feature is inconsistent with the given attractor's hand part. Thus the larger a particular feature factor for a particular contact-attractor pair, the larger the attractor's Voronoi cell will appear to the contact, and the more likely the contact will fall within that cell and be assigned to its attractor.

Since each contact can have a different feature weighting for matching to the same attractor, each contact can encounter a differently warped Voronoi diagram. For example, a sufficiently large thumb weighting for a contact could make the thumb attractor so powerful that the Voronoi diagram encountered by the contact could contain only one Voronoi cell, a thumb Voronoi cell covering the whole surface. In actuality, thumb contact features are never unambiguous enough to warrant weightings this large.

The weighting functions were arrived at by trial and error. Since each weighting function can move the boundaries of one Voronoi cell, the experimenter typically decides from finger motion ranges where each boundary should be. The current boundary positions can be determined by repeatedly lifting and touching a finger over different spots until a surface position is found where finger identity becomes unstable, alternating between two Voronoi cells each time the finger touches. The

experimenter then adjusts the amplitude of the weighting function until the boundary moves to the desired location. Care must be taken to balance the thumb and palm weightings or the boundary between thumb and inner palm heel Voronoi cells may shift unintentionally.

4.4.6.4 Thumb and Inner Palm Orientation Factor

Figure 4.17 shows the right thumb and right inner palm heel orientation

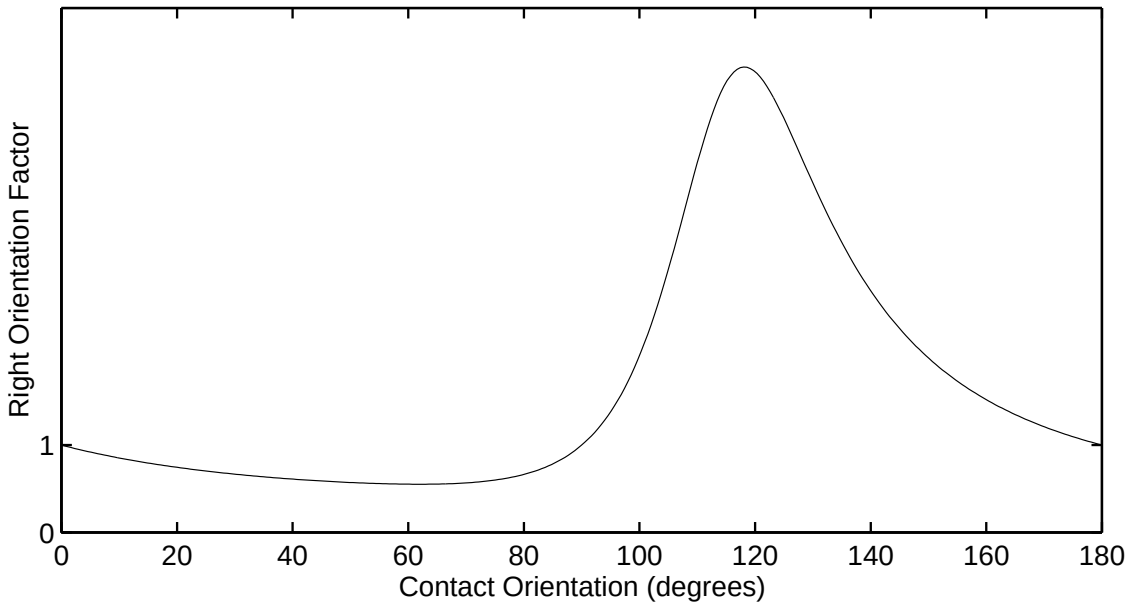


Figure 4.17: Right thumb and inner palm heel orientation factor, $Pi_{worient}[n]$, versus orientation of the contact's fitted ellipse, $Pi_{\theta}[n]$.

factor versus orientation of a contact's fitted ellipse. Orientation of these hand parts tends to be about 120°, whereas fingertip and outer palm heel contacts are usually very close to vertical (90°), and orientation of the left thumb and left inner palm heel averages 60°. The right orientation factor therefore approaches a maximum at 120°. It approaches the default value of 1 at 0°, 90°, and 180° where orientation is inconclusive of identity, and reaches a minimum at 60°, the favored orientation of

the opposite thumb or palm heel. The corresponding relationship for the left thumb and inner palm heel orientation factor is flipped about 90°.

4.4.6.5 Thumb Size Factor

Figure 4.18 approximately plots the thumb size factor. Since thumb size as

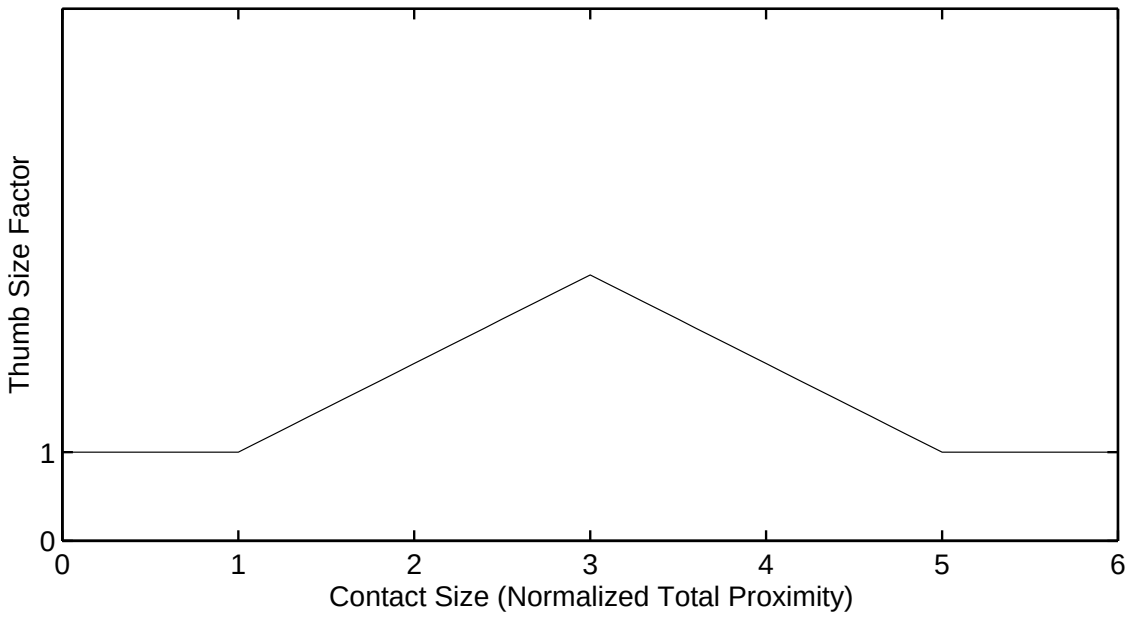


Figure 4.18: Thumb size factor, $Pi_{wthumb\_size}[n]$ versus a contact's total proximity, $Pi_z[n]$.

indicated by total proximity tends to peak at two or three times the size of the typical curled fingertip, the thumb size factor peaks at these sizes. Unlike palm heels, thumb contacts cannot be much larger than two or three times the default fingertip size, so the thumb factor drops back down for larger sizes. Since any hand part can appear small when touching the surface very lightly or just starting to touchdown, small size is not distinguishing, so the size factor defaults to 1 for very small contacts.

4.4.6.6 Palm Heel Size Factor

Figure 4.19 approximately plots the palm heel size factor. As more pressure

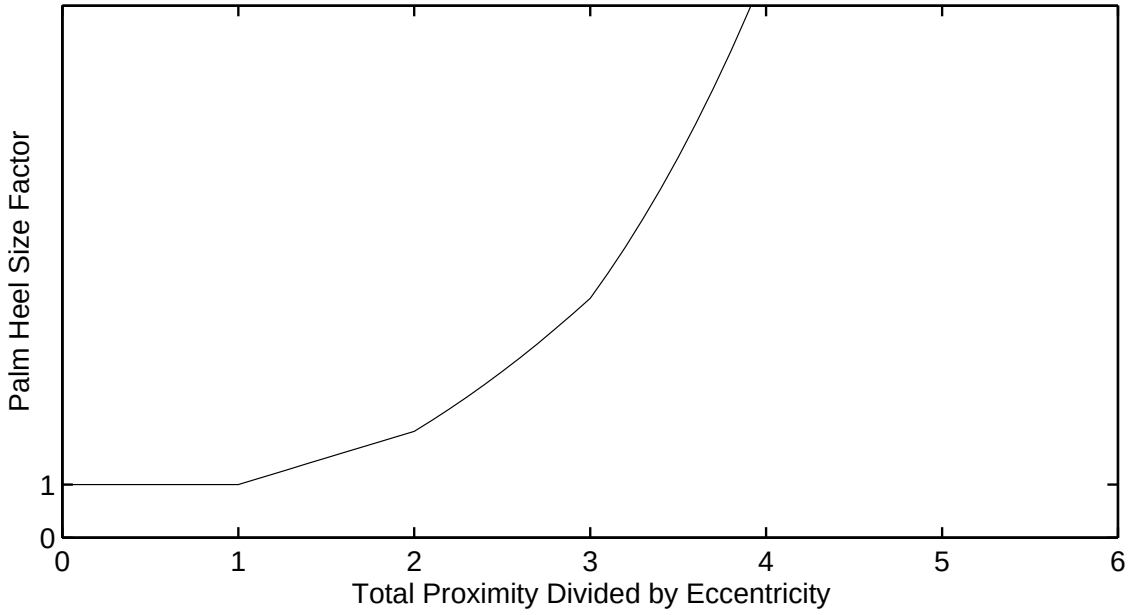


Figure 4.19: Palm heel size factor, Pi_{wpalm_size} versus the ratio of a contact's total proximity to its eccentricity, $P_z[n]/P_\epsilon[n]$.

is applied to the palms, the palm heel contacts can grow quite large, remaining fairly round as they do so. Thus the palm heel size factor is much like the thumb size factor except the palm factor is free to increase indefinitely. For palm heels larger than the maximum expected thumb size, the palm heel size factor becomes so large that the palm heel Voronoi cells engulf the entire surface. Thus if the full weight of the hands rests on the palm heels and fully flattens them, they will be correctly identified anywhere on the surface, regardless of attractor ring alignment.

However, fingertip contacts can grow by becoming taller as the fingers are flattened. But since finger width is constant, the eccentricity of an ellipse fitted to a growing fingertip contact increases in proportion to the height. To prevent flattened fingers from having a large palm factor, the size measure is modified to be the ratio of total contact proximity to contact eccentricity. This has little effect for palms,

whose eccentricity remains near 1, but cancels the high proximities of flattened fingertips. Though directly using the width from the contact's fitted ellipse would be less accurate for low resolution electrode arrays, the proximity to eccentricity ratio basically indicates contact width.

4.4.6.7 Palm Heel Separation Factor

Another important distinguishing feature of the palm heels is that wrist anatomy keeps the centroids of their contacts separated from one other and from the fingers by several centimeters. This is not true of the thumb and fingertips, which can be moved within a centimeter of one another via flexible joints. Minimum contact separation can be measured without knowing contact identities by searching all contacts for the nearest neighbor contact of a given contact and measuring the distance to that neighbor. As plotted approximately in Figure 4.20, the

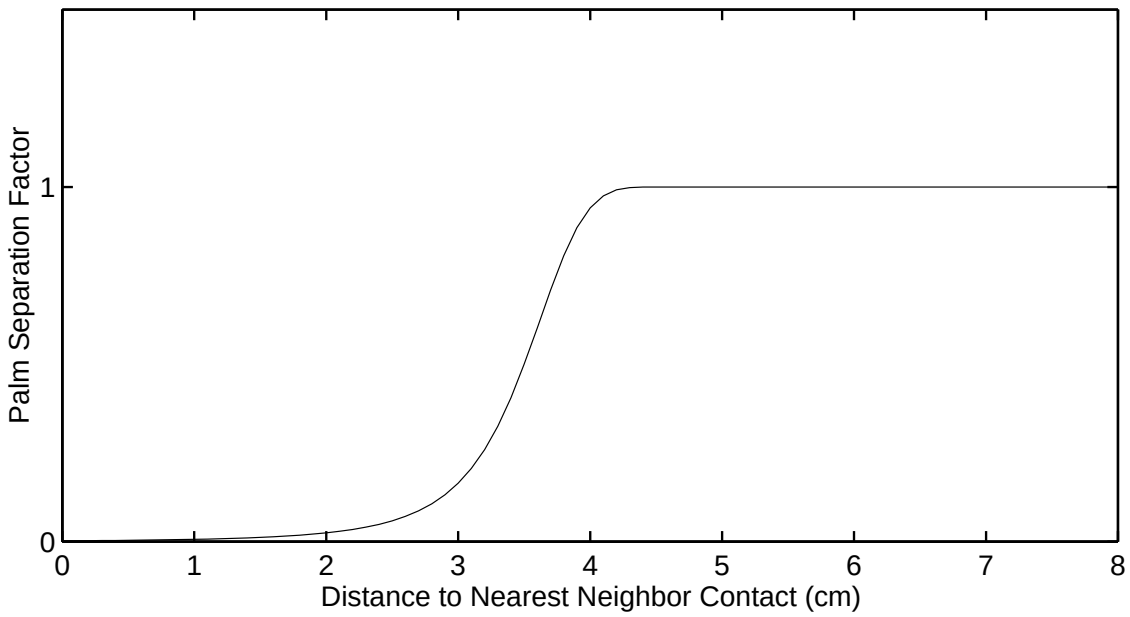


Figure 4.20: Palm heel separation factor, $Pi_{wpalm\_sep}[n]$ versus the Euclidean distance between contact Pi and its nearest neighbor contact.

palm separation factor quickly decreases as the separation between the contact and its nearest neighbor falls below a few centimeters, indicating that the given contact (and its nearest neighbor) are *not* palm heels.

Unlike the size and orientation factors, which only become reliable as the weight of the hands fully compresses the palms, the palm separation factor is especially helpful in distinguishing pairs of adjacent fingertips from palm heels because it applies equally well to light, small contacts. For small separations, this weighting factor becomes so small that the palm heel Voronoi cells vanish. Therefore, a pair of contacts which are within about 3 cm of one another will not be identified as palm heels, regardless of their position on the surface or the alignment of the attractor ring. However, the palm separation factor should only be made this influential if the segmentation system always either merges the two palm heels into one huge palm contact or divides the palm across its central vertical crease into exactly two heel contacts. If the segmentation system erroneously splits one of the palm heels into two contacts which are less than 3 cm apart, the small separation factor which results can cause both contacts to be identified erroneously as fingers, again regardless of the alignment of the attractor ring.

4.4.6.8 Forepalm Attractors and Weightings

The MTS includes four additional attractors near the center of the attractor ring to handle forepalms contacts and other extra groups from segmentation of flattened palms. This increases the size of the contact attractor distance matrix to 11×11. Since the forepalms typically do not touch the surface unless the rest of the hand is flattened onto the surface as well, the forepalm attractors must be weighted such that contacts near hand center are assigned to them only when the hand is flattened. When the hand is not flattened, contacts near hand center might be fingers accessing keys on the bottom row of the key layout and should not be identified as forepalms.

The easiest way to determine if the hand or palm is flattening is to measure the total proximity of all contacts assigned to the hand, $H_{totalz}[n]$. A forepalm weighting function is devised to be so small when the hand is not flattened that the forepalm Voronoi cells totally vanish:

$$w_{i,8-11} = \frac{1}{\max\left(2, (8 - H_{totalz}[n])\right)} \tag{4.41}$$

Only when the total hand proximity $H_{totalz}[n]$ becomes large due to finger or palm flattening will the forepalm weightings be large enough that their Voronoi cells appear near the center of the hand. To discourage fingers or palms away from hand center from being assigned to forepalm attractors, forepalm entries in the contact-attractor distance matrix are squared again, making forepalm assignment costs vary with the fourth power of Euclidean distance. The final forepalm assignment costs $[c_{i,8-11}]$ can then be written:

$$c_{i,8-11} = \max\left(2, (8 - H_{totalz}[n])\right) \times (d^2_{i,8-11} + 2)^2 \tag{4.42}$$

4.4.6.9 The Fully Weighted Assignment Cost Matrix

All of the static and dynamic weightings are combined to form a fully weighted assignment cost matrix $[c_{ij}]$, where:

$$c_{ij} = \begin{cases} (d^2_{ij} + 2)/(Pi_{wthumb\_size}[n] Pi_{worient}[n]) & \text{if } j == 1 \\ (d^2_{ij} + 2) & \text{if } 2 <= j <= 5 \\ 4(d^2_{ij} + 2)/(Pi_{wpalm\_size}[n] Pi_{wpalm\_sep}[n]) & \text{if } j == 6 \\ 4(d^2_{ij} + 2)/(Pi_{wpalm\_size}[n] Pi_{worient}[n] Pi_{wpalm\_sep}[n]) & \text{if } j == 7 \\ \max\left(2, (8 - H_{totalz}[n])\right) \times (d^2_{ij} + 2)^2 & \text{if } 8 <= j <= 11 \end{cases} \tag{4.43}$$

The basic assignment optimization of Equation 4.12 is then restated as finding the permutation $\{\pi_1, \ldots, \pi_{11}\}$ of integer hand part identities $\{1, \ldots, 11\}$ which minimizes:

$$\sum_{i=1}^{11} c_{i\pi_i} \tag{4.44}$$

where c_{ij} is the weighted distance from contact i to attractor j, and contact i and attractor j are considered assigned to one another when $\pi_i \equiv j$. Though the various weightings can warp the Voronoi cells in complex ways which were not anticipated in the design of the combinatorial search exchange sequence (Section 4.4.4.4), no convergence failures have been noticed as long as palm attractors are allowed to swap with any other attractor on the ring. Making the exchange neighborhoods of palm attractors include all other attractors ensures that when a palm Voronoi cell vanishes it can give up its contact to any finger attractor, or when a palm Voronoi cell engulfs the surface it can accept a contact from any finger attractor.

4.4.6.10 Tolerance of Different Hand Sizes

Though the default finger positions and thus the attractor positions were determined from the author's medium-sized male hand, anthropomorphic data suggests that the given attractor ring will perform well for most adult hands. Wagner's study [150] of anthopometry and biomechanics in the pianist's hand found less than a 10% standard deviation in all hand shape parameters except the prominences (relative lengths with hand flattened) of the thumb and pinky. While male hands are on average nearly 10% larger than female hands, the range of adult hand sizes for a given hand posture is much narrower than the range of finger flexion and extension tolerated in Figure 4.15. Thus the only people for which the attractor ring might need to be resized or reshaped would be very small children.

The feature weighting functions are more sensitive to hand size variation than the attractor ring itself. For example, the palm heel separation factor cuts off sharply at 4 cm and might become erroneously small for smaller hands whose palm heels are as close as 3 cm to one another. Luckily, palm heel separation would actually be much easier to measure and adapt to for individual operators than finger lengths since the palm heels are immobile relative to one another. The system could power up assuming a small palm heel separation to accomodate operators with small hands.

Then, the actual palm heel separation could be measured upon the first confirmed touchdown of both palm heels, and this measured palm heel separation could then be used to calibrate the inflection points of the palm heel separation function of Figure 4.20.

Fatter fingers cause larger total proximities for all contacts and thus cause the proximity inflection points of the thumb and palm heel size factors to become misaligned. Modifications to the thickness of the surface dielectric can also scale all proximities and thus disrupt the calibration of the proximity inflection points. Such variations in proximity scaling can potentially be dealt with by adapting the average fingertip proximity, $Z_{averageFingertip}$ in Equation 3.20, to peak fingertip proximities when the fingertips are normal to the surface.

4.4.7 Thumb Verification

The identifications produced by this attractor assignment method are highly reliable when all five fingers are touching the surface or when thumb and palm features are unambiguous. Checking that the horizontal coordinates for identified fingertip contacts are in increasing order easily verifies that fingertip identities are not erroneously swapped. However, when only two to four fingers are touching, yet no finger strongly exhibits thumb size or orientation features, the assignment of the innermost finger contact may wrongly indicate whether this contact is the thumb because distance-squared assignment is sometimes too lenient about thumb-fingertip angles and separations. In this case, the MTS employs a thumb verification process to take further measurements between the innermost finger contact and the other fingers. If these further measurements strongly suggest the innermost finger contact identity is wrong, the thumb verification process changes the assignment of the innermost finger contact. Once the finger assignments are verified, statistics about the assignments within each hand such as the number of touching fingertips

are compiled. These statistics provide convenient summaries of identification results for other modules.

Figure 4.21 shows the steps within the thumb verification process. The first is to compute several velocity, separation, and angle factors for the innermost contact identified as a finger relative to the other contacts identified as fingers. Since these inter-path measurements presuppose a contact identity ordering, they could not have easily been included as attractor distance weightings because contact identities are not known until the attractor distance minimization is complete. For the descriptions below, let FI be the innermost contact tentatively identified as a finger, FN be the next innermost finger contact, and FO be the outermost finger contact.

4.4.7.1 Inner Finger Separation Factor

The separation between thumb and index finger is often larger than the separations between fingertips, but all separations tend to grow as the fingers are outstretched. Therefore an inner separation factor $inner\_separation\_fact$ is defined as the ratio of the distance between the innermost and next innermost finger contacts to the average of the distances between other adjacent fingertip contacts, $avg\_separation$:

$$inner\_separation\_fact = \min\left(1, \frac{\sqrt{(FI_x - FN_x)^2 + (FI_y - FN_y)^2}}{avg\_separation}\right) \qquad (4.45)$$

The factor is clipped to be greater than one since an innermost separation less than the average can occur regardless of whether thumb or index finger is the innermost finger touching the surface. In case there are only two finger contacts, a default average separation of 2-3 cm is used. This factor tends to become larger than one if the innermost contact is actually the thumb but remains near one if the innermost contact is a fingertip.

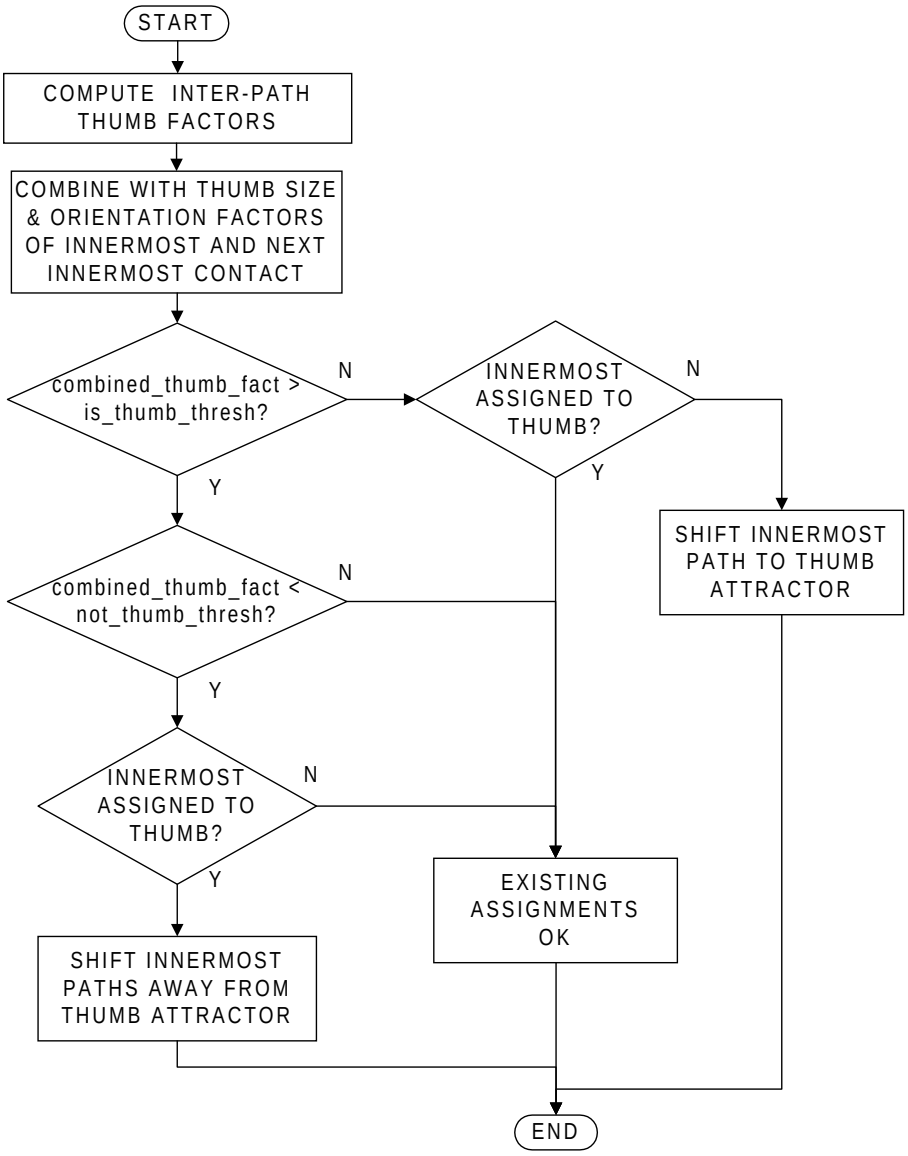


Figure 4.21: Flow chart of the thumb presence verification algorithm.

4.4.7.2 Inner Finger Angle Factor

Since the thumb rarely moves further forward than the fingertips except when the fingers are curled into a fist, the angle between the innermost and next innermost finger contacts can help indicate whether the innermost finger contact is the thumb. For the right hand the angle of the vector from the thumb to the index finger is most often 60°, though it ranges to 0° as the thumb moves forward and to 120° as the thumb adducts under the palm. This is reflected in the approximate plot of the inner angle factor in Figure 4.22, which peaks at 60° and approaches 0 toward 0°

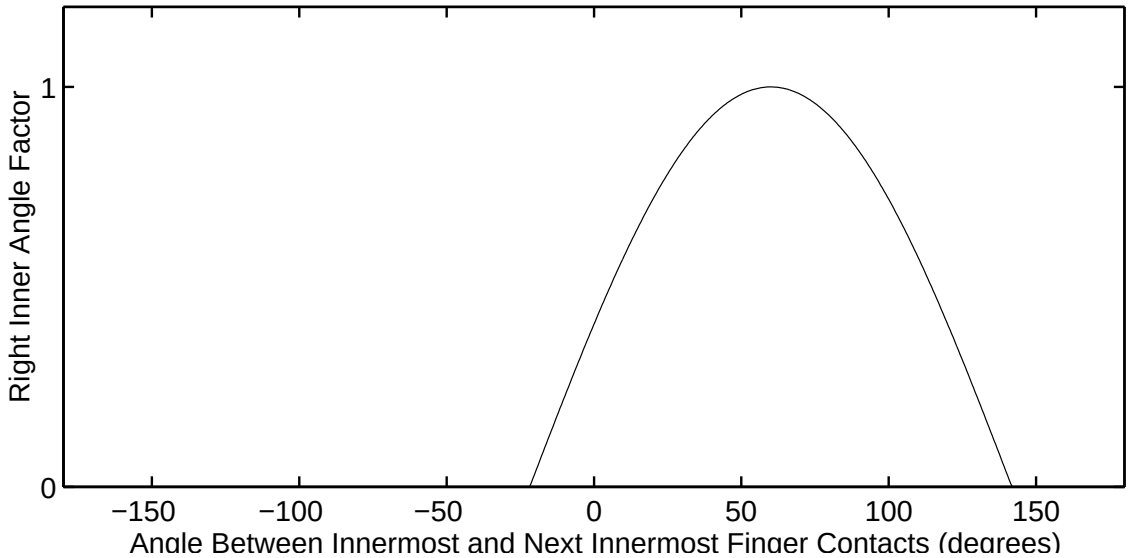


Figure 4.22: Right inner angle factor, $angle\_fact$, versus the vector angle between the two innermost contacts identified as fingers.

and 120°. If the innermost finger contact is actually from the index fingertip, the measured angle between innermost and next innermost contact would probably be between 30° and minus 60°, producing a very small angle factor.

The inner separation and angle factors are highly discriminating of neutral thumb postures, but users often exceed the above cited separation and angle ranges when performing hand scaling or rotation gestures. For instance, during an anti-pinch gesture, the thumb may start pinched against the index or middle fingertip,

but then the thumb and fingertip slide away from one another. This causes the inner separation factor to be relatively small at the start of the gesture. Similarly, the thumb-index angle can also exceed the range expected by the inner angle factor at the beginning or end of hand rotation gestures, wherein the fingers rotate as if turning a screw. To compensate, the inner separation and angle factors are fuzzy OR'ed with expansion and rotation factors which are selective for symmetric finger scalings or rotations centered on a point between the thumb and fingertips.

4.4.7.3 Thumb-Fingertip Expansion Factor

When defined by the following approximate equation, the expansion factor peaks as the innermost and outermost finger contacts slide at approximately the same speed and in opposite directions, parallel to the vector between them:

$$\begin{aligned} expansion\_factor[n] &\approx -\sqrt{FI_{speed}[n] \times FO_{speed}[n]} \\ &\quad \times \cos\left(FI_{dir}[n] - \angle(FI[n], FO[n])\right) \\ &\quad \times \cos\left(FO_{dir}[n] - \angle(FI[n], FO[n])\right) \quad (4.46) \\ clipped\_expansion\_fact[n] &= \max\left(0, expansion\_factor[n]\right) \quad (4.47) \end{aligned}$$

where $\angle(FI[n], FO[n])$ is the angle between the fingers:

$$\angle(FI[n], FO[n]) = \arctan\left(\frac{FI_y[n] - FO_y[n]}{FI_x[n] - FO_x[n]}\right) \quad (4.48)$$

Translational motions of both fingers in the same direction produce negative factor values which are clipped to zero by the max operation. Computing the geometric rather than arithmetic mean of the innermost and outermost speeds aids selectivity by producing a large expansion factor only when speeds of both contacts are high.

4.4.7.4 Thumb-Fingertip Rotation Factor

The rotation factor must also be very selective. If the rotation factor was simply proportional to changes in the angle between innermost and outermost finger,

it would erroneously grow in response to asymmetries in finger motion such as when the innermost finger starts translating downward while the outermost contact is stationary. To be more selective, the rotation factor must favor symmetric rotation about an imaginary pivot between the thumb and fingertips. The approximate rotation factor equation below peaks as the innermost and outermost finger move in opposite directions, but in this case the contacts should move perpendicularly to the vector between them:

$$\begin{aligned} rotation\_factor[n] &\approx -\sqrt{FI_{speed}[n] \times FO_{speed}[n]} \\ &\quad \times \sin\left(FI_{dir}[n] - \angle(FI[n], FO[n])\right) \\ &\quad \times \sin\left(FO_{dir}[n] - \angle(FI[n], FO[n])\right) \qquad (4.49) \\ clipped\_rotation\_fact[n] &= \max\left(0, rotation\_factor[n]\right) \qquad (4.50) \end{aligned}$$

Since motions which maximize this rotation factor are easy to perform between the opposable thumb and another finger but difficult to perform between two fingertips, the rotation factor is a robust indicator of thumb presence.

4.4.7.5 Combining and Testing the Thumb Factors

The following expression essentially ORs these inter-contact factors with the innermost and next innermost contacts' thumb features:

$$\begin{aligned} combined\_thumb\_factor[n] \approx\ & clipped\_expansion\_fact[n] \\ & + clipped\_rotation\_fact[n] \\ & + inner\_separation\_factor[n] \times angle\_factor[n] \\ & \times (FI_{worient}/FN_{worient}) \\ & \times (FI_{wthumb\_size}/FN_{wthumb\_size}) \qquad (4.51) \end{aligned}$$

The feature weighting ratios of this expression attempt to compare the features of the innermost contact to current features of the next innermost contact, which is

already known to be a fingertip. If the innermost contact is also a fingertip its features should be similar to the next innermost, causing the ratios to remain near one. However, thumb-like features on the innermost contact will cause the ratios to be large.

The action taken by the thumb verification module (Figure 4.21) depends on tests of $combined\_thumb\_factor[n]$ against two thresholds, producing three cases:

1. If $combined\_thumb\_factor[n]$ exceeds the high threshold, the innermost contact is definitely a thumb. If the assignment algorithm has not already put the innermost contact with the thumb attractor, the assignment algorithm must be overridden. Thumb verification shifts the innermost contact's assignment inward on the attractor ring to the thumb attractor.

2. If $combined\_thumb\_factor[n]$ is between the low and high thresholds, the thumb verification test is ambiguous. The identification of the innermost contact made by the assignment algorithm is left unchanged since the assignment algorithm takes into account hand position estimate clues, but thumb verification does not.

3. If $combined\_thumb\_factor[n]$ is less than the low threshold, the innermost contact is definitely *not* the thumb. For this conclusion to be reached, the expansion and rotation velocity factors must essentially be zero, the innermost and next innermost contact sizes and orientations must match, and either the inner angle must be near horizontal or the inner separation must be less than 2.5 cm. If the assignment algorithm has put the innermost contact with the thumb, thumb verification overrides it by shifting the innermost contact outward to the index finger attractor. Non-innermost contacts may need to be shifted outward as well to make the index finger attractor available.

Like the Voronoi cells of the assignment algorithm, the thumb verification expression (Equation 4.51) and thresholds establish clear cutoffs for thumb identity. Instead of utilizing hand position estimates, thumb verification imposes stricter tests on inter-contact velocity, angles, and separations. Since Equation 4.51 adds or essentially ORs the various feature measurements, thumb verification is most difficult when only one of the features is discriminating. Figure 4.23 plots the inner separation and angle cutoffs when the velocity factors are zero and the size and orientation ratios are one, providing no discriminating information. Likewise, for the expansion or rotation factors acting alone to surpass the high threshold and trigger identification of the innermost contact as the thumb, thumb-finger motions must be properly symmetric and exceed speeds of 1.5 cm/sec. For size ratios alone to trigger identification of the innermost as the thumb, the innermost contact must be at least twice as large as the next innermost. When several features act in combination, they need not be as strong as cited here to force identification of the innermost as the thumb.

4.4.8 Ratcheting Identification Accuracy

The quality of the constraints available for finger identification fluctuates as palm pressures change, fingers lift off, or more hand parts touch down on the surface. Running the assignment algorithm from scratch after segmenting each image could discard accurate assignments made when the hands started in a neutral posture or when more hand parts were touching the surface. Therefore the assignment and thumb verification algorithms for a hand are only executed for images in which the total hand proximity is increasing or when a touchdown has recently been attributed to the hand. Reassignment is *not* triggered by finger liftoff. This prevents degradation of identifications upon finger liftoff such as the dummy contact propagation upon pinky liftoff in Figure sequence 4.11–4.13. This also prevents erroneous

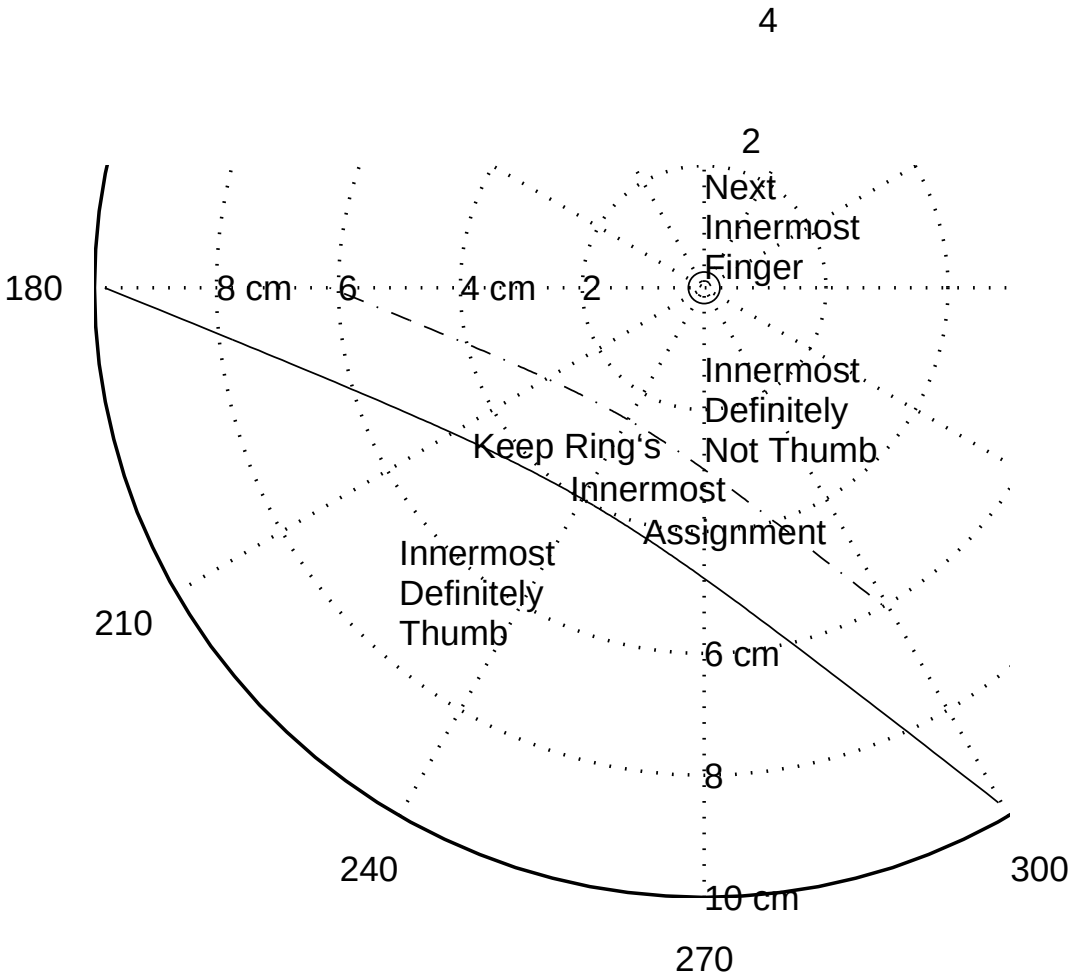


Figure 4.23: Thumb verification cutoffs for right hand versus inner separation and angle when other inter-contact features are not discriminating. This is a polar plot of the decision regions relative to the position of the next innermost finger contact (circle at center). If the innermost contact lies to the lower left of the solid diagonal line, it is definitely a thumb. If the innermost contact lies to the upper right of the dashed line, it is most likely *not* a thumb. If the innermost contact lies in the band between the solid and dashed lines, the thumb verification test is inconclusive. In this case, the innermost identification based upon assignment to the attractor ring is left unchanged to retain weak clues from the hand position estimate.

shifting of assignments for hands which touchdown in a neutral orientation but then undergo extreme rotation, past the hand rotation tolerances of the attractor ring.

Persistent path tracking extends previous identifications of existing contacts to images for which the assignment algorithm will not be executed. The hand position estimates then only need be relied upon to ensure accurate reidentification of fingers which lift off the surface temporarily. Such a system which only reassigns fingers when image information increases substantially, such as when another finger touches down, produces much more stable and reliable identifications.

4.4.9 Finger Identification Results

Since the finger identification system is deterministic in the sense that it usually provides repeatable results for motion patterns having roughly the same geometry, identification convergence will be demonstrated upon a variety of extreme conditions, each of which reveals the importance of a particular identification mechanism. The trivial cases of isolated finger touchdown and touchdown of the whole hand in default position are note shown since they can be predicted entirely from the weighted, static Voronoi diagram of Figure 4.14.

Figures 4.24–4.34 plot finger trajectories (blue arrowheads) which are selectively labeled with the finger and palm identities (F#) which the identification system has assigned to the paths at each time step (in each proximity image). The identity labels, shown as an 'F' followed by the hand part number (see Table 4.1 on Page 4.1 for hand part number list), are printed upon hand part touchdown, hand part liftoff, and any time step in between that the identification system changes the hand part identity. Thus arrows without identity labels retain the identity last shown along the given trajectory. Usually when a trajectory has the same label at its beginning and end and no labels in between, it was identified correctly in the image in which it first touched down, possibly while the hand position estimate was still at default, so its identity never had to be changed.

Except for rotations and scalings in thumb verification, finger velocity has little or no effect on the results, but the hand will be moved in some cases to spread out the labels and make changes in identity easily visible. To indicate relative timing of the trajectories, blue dotted lines connect all finger contacts at selected time steps, especially when any hand part touches down or lifts off. Ellipses (cyan) are also centered on each hand part at selected time steps to indicate orientation, eccentricity, and relative size of the contacts.

Figures 4.24, 4.26, 4.31, 4.32 also include the motions of selected hand part attractors as caused by changes in the estimated hand position. Since all attractors in the ring translate together, only those attractors relevant to the hand parts being identified or the involved region of the surface are shown. Attractors are labeled with an 'A' followed by their associated hand part number at the first time step only. Attractor arrows are plotted in red and have a diamond at their head.

The right hand is used on the right half of the multi-touch surface in every experiment, avoiding hand identification issues. For every plot, the right hand position estimate is allowed to drift back to the default hand position before starting the experiment. Then the hand touches down in the indicated region of the surface and identification attempts begin immediately. Trajectory capture is halted before or immediately after liftoff, so the drift of the hand position estimates back to the default is never shown, but this typically takes just a second or two.

The results of Figures 4.24–4.34 can be summarized as follows. Single, isolated fingers are only identified correctly if hand position estimates are consistent with actual hand position, *i.e.*, the attractor ring is aligned horizontally within a centimeter and vertically within about 5 cm. Isolated palm heels, on the other hand, will be properly identified regardless of their position or that of the attractor ring if their contacts reach a uniquely large size. Multiple fingers are always ordered properly around the ring and never misidentified as palms unless they are separated

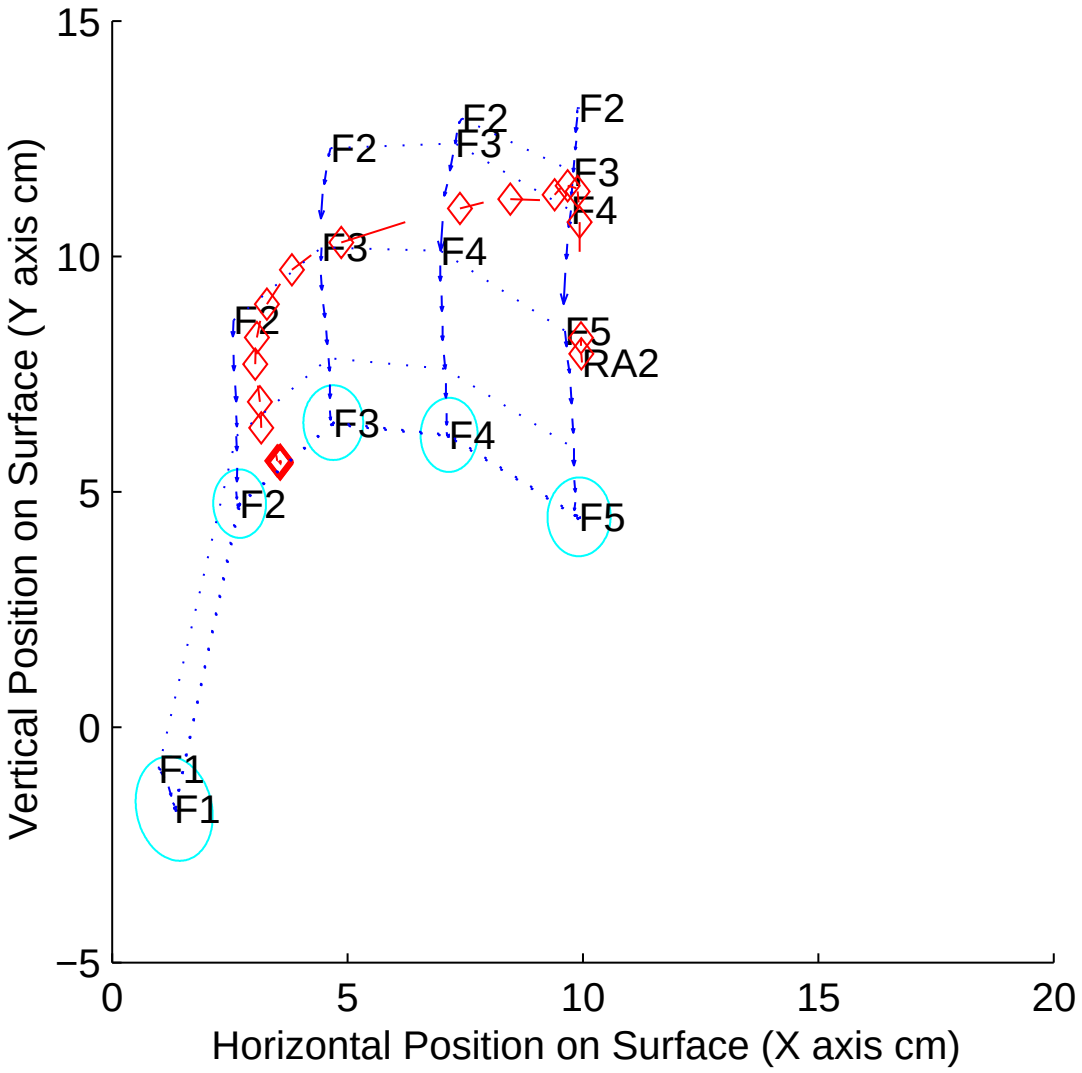


Figure 4.24: The hand starts at the top left with pinky touching down first and the other fingers rolling onto the surface at regular time intervals as the hand as a whole slides down. Since the original pinky touchdown occurs in the index finger Voronoi cell while there are no feature or inter-contact constraints, the pinky is initially misidentified as the index finger (F2), and the index finger attractor (A2) starts moving up to meet it. The sorting behavior of distance-squared assignment forces identifications of the pinky to shift as fingers touch down to the left of it, with all identifications corrected once all four fingertips touch. As identities are corrected, the hand position estimate corrects left and follows the hand down so the A2 attractor ends up near the actual index finger.

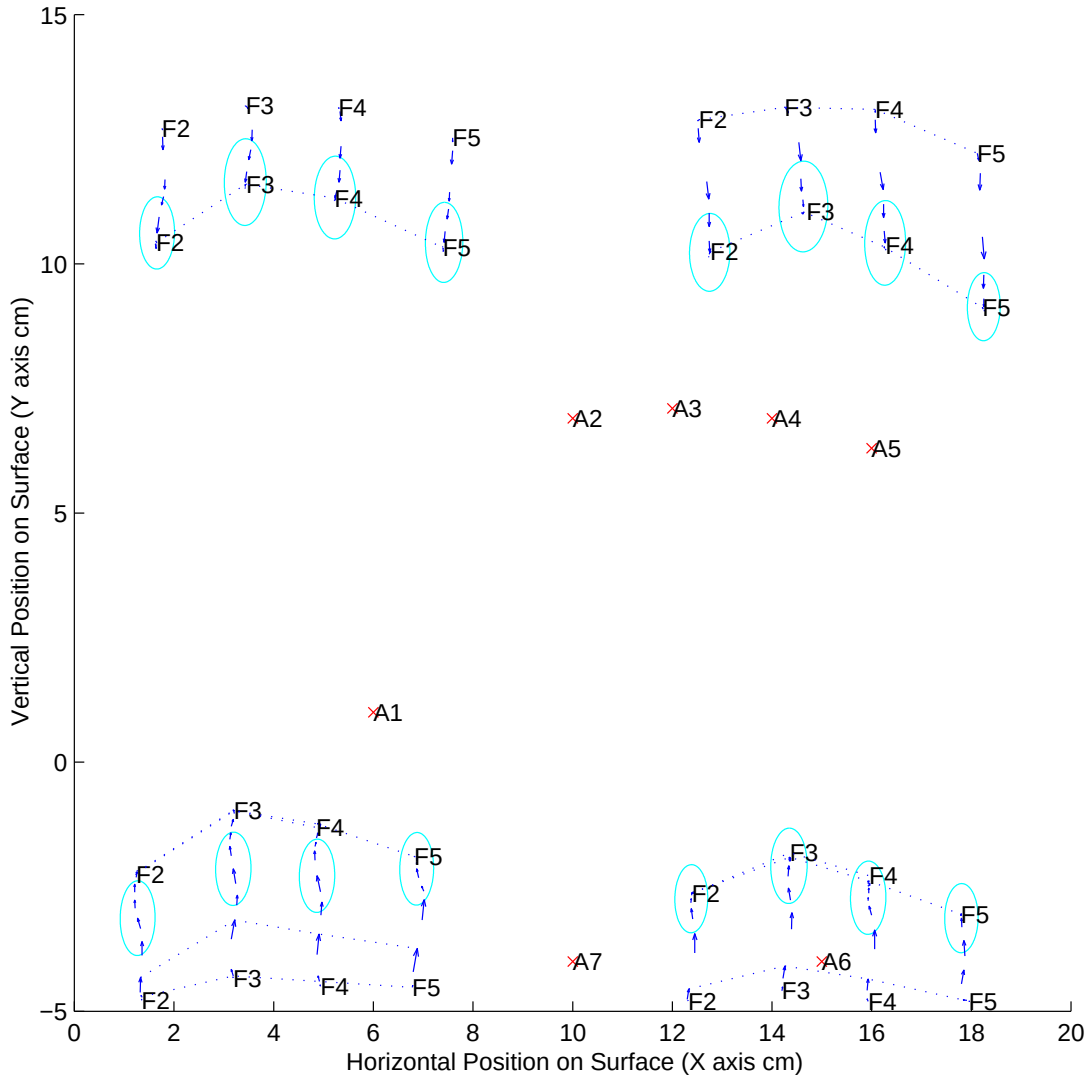


Figure 4.25: Row of four fingertips is placed successively at the four corners with attractors always starting at their defaults (red x's). Correct identification throughout every placement shows that four fingertips in a roughly horizontal row are sufficient for perfect, instantaneous identification anywhere; from this it follows that five fingers or four fingertips plus palm heels will be identified perfectly anywhere. All cases rely on the translation-invariant sorting behavior of distance-squared assignment. At the bottom corners the palm heel separation factor comes into play. At the left corners, the assignment algorithm may attribute the leftmost contact to the thumb, but thumb verification finds the inner angle and separation *not* indicative of the thumb (see Figure 4.23) and shifts all identities to the right, correctly attributing F2 to the leftmost contact.

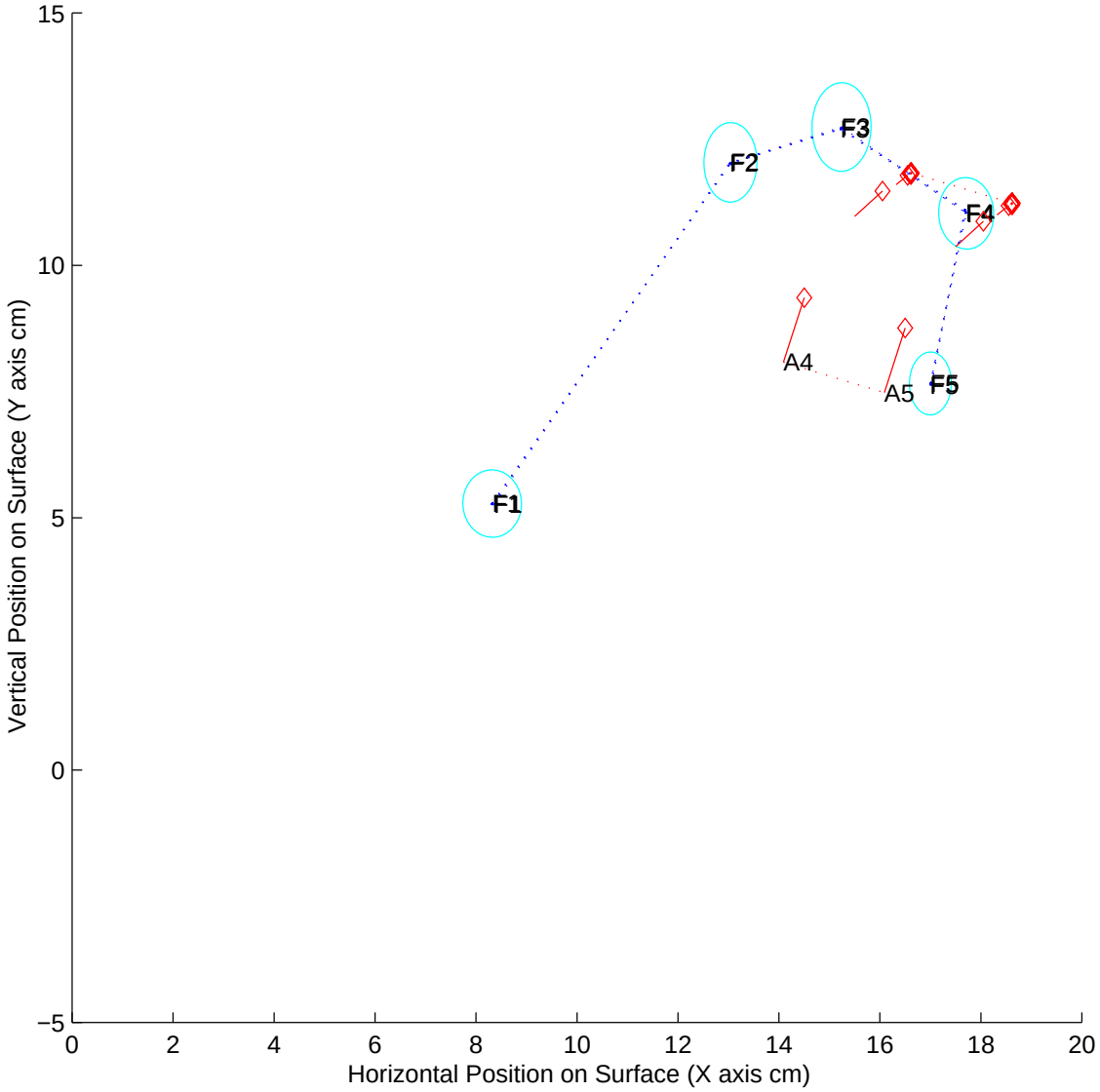


Figure 4.26: A claw hand with the pinky crossed under the ring finger verifies the finger rotation tolerances postulated for the attractor ring in Figure 4.11 on Page 160. The pinky contact (bottom right) is always identified correctly, regardless of attractor translation, since the angle between ring and pinky does not quite become perpendicular to the A4-A5 attractor angle (red dotted-segments). Simple sorting of the horizontal contact coordinates would fail in this case, swapping the F4 and F5 identities. Further cross-under of the pinky would pass the perpendicular and also cause swapping, but the hand twisted as shown is already at the maximum range of ulnar deviation (wrist rotation).

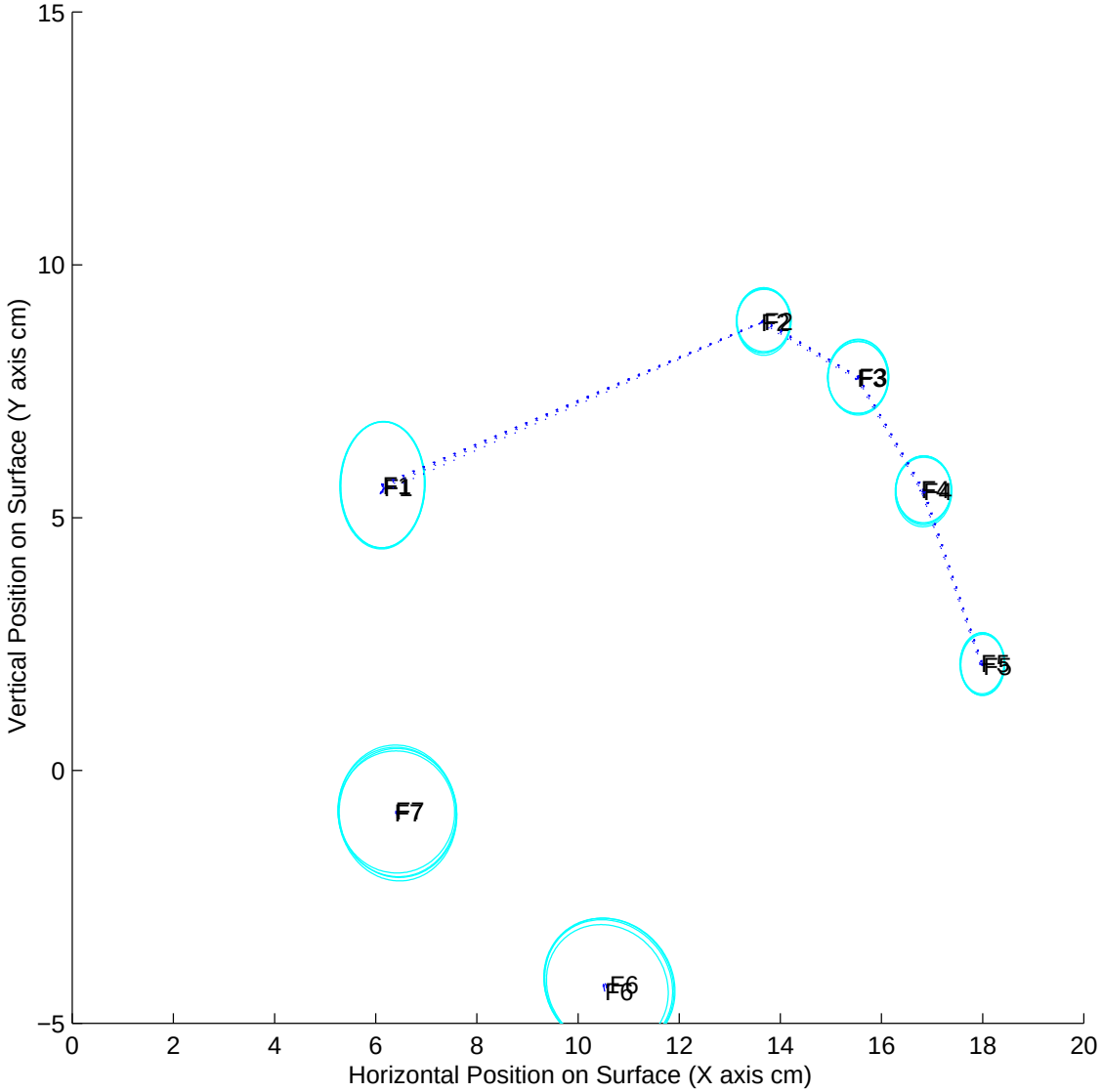


Figure 4.27: Fingers in a hand rotated fully clockwise to the limits of ulnar deviation at the wrist are always identified perfectly. Presence of the palm heels is not needed for this correct result. However, the fingertips are kept well spread to avoid parallelogram-electrode-induced segmentation merging as occurred for the sideways hand in Figure 3.13 on Page 96 of Chapter 3.

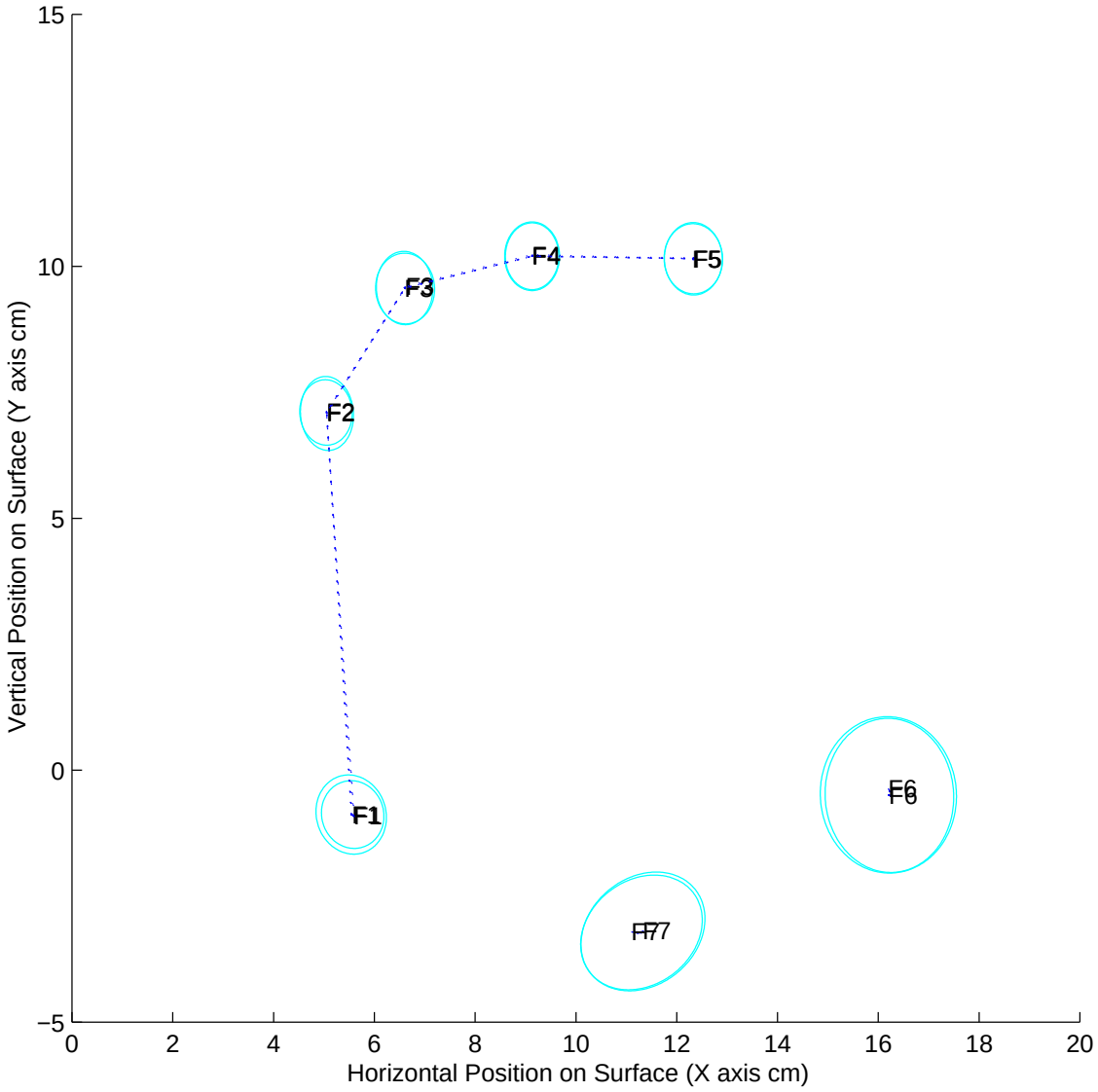


Figure 4.28: Fingers in a hand rotated fully counter-clockwise to the limits of radial deviation are identified correctly in this experiment, but not always. As will be seen in the next figure, absence or merging of both palm heels will open up the inner palm heel attractor (A7) which can then grab the thumb contact which is at the lower left.

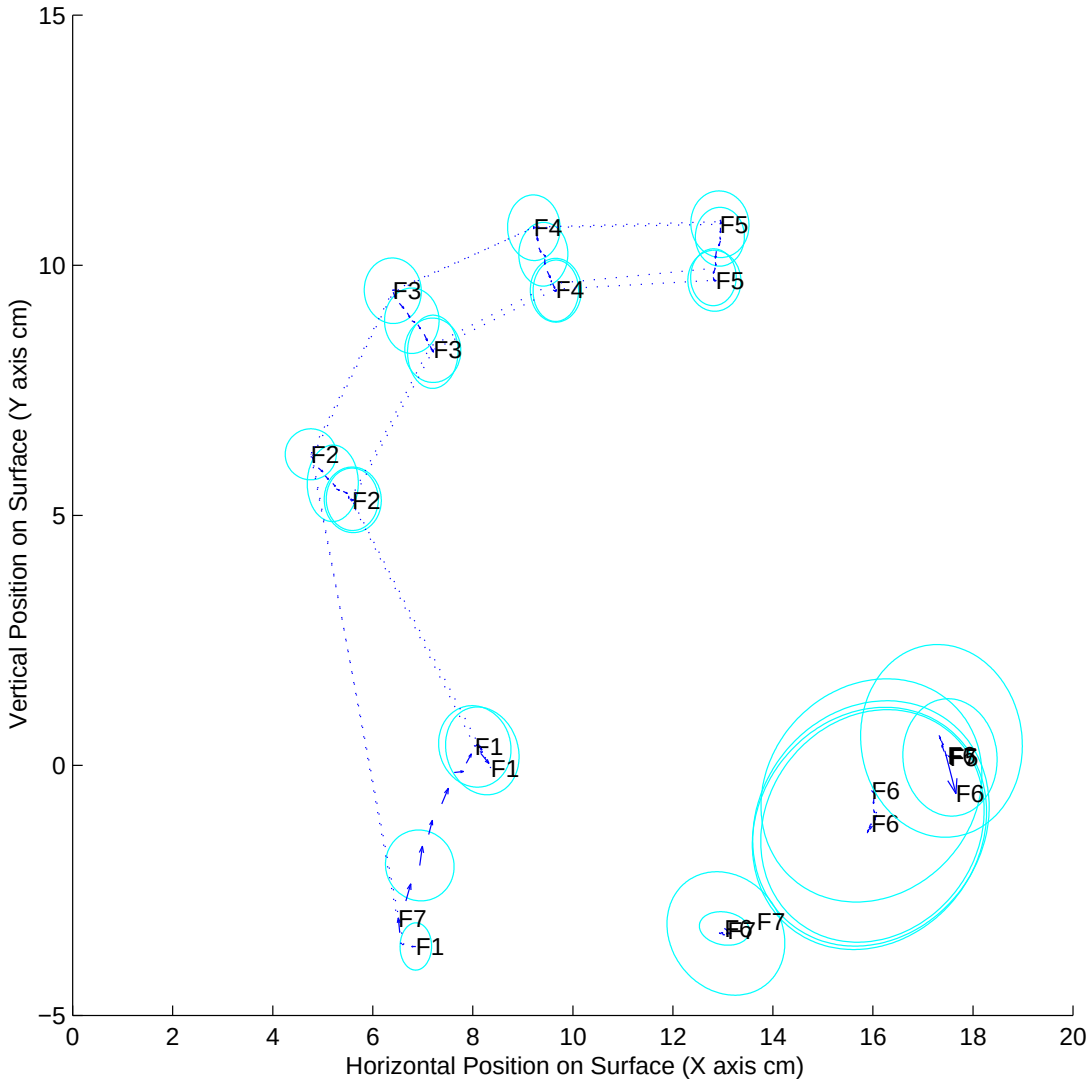


Figure 4.29: Absence or merging of both palm heels can cause thumb misidentification when right hand is rotated fully counter-clockwise. In this case, all identifications are initially correct, but as the palm heels press onto the surface they merge into one huge contact, leaving the inner palm heel attractor (A7) unfilled. This causes the identity attributed to the thumb contact at lower left to change temporarily to F7 until pressure is released from the palms and they split back into two heels. Identities of the fingertips are unaffected. This sort of failure can also occur if the palm heels never touch down because, under this much rotation, the position of the thumb relative to the index finger is the position normally expected for the inner palm heel when the hand is not rotated.

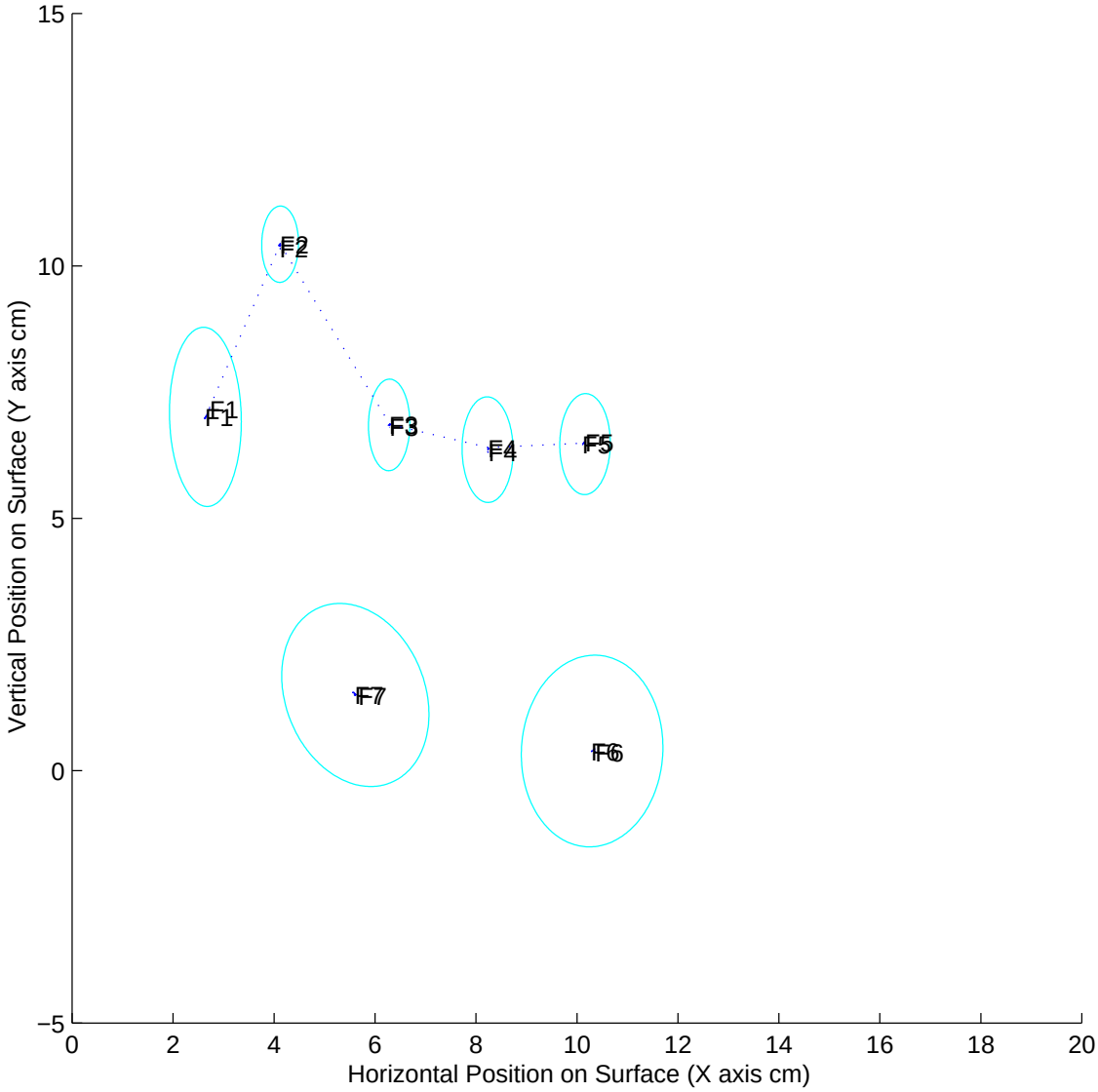


Figure 4.30: Fingers in a pen grip configuration are always identified correctly anywhere on the surface as long as they are segmented properly. In this experiment the fingers were kept a little looser than they would be in practice to prevent the thumb contact from merging with the index fingertip or the other fingers from merging with the palms. Note that the outer finger contacts labeled F3-F5 are actually caused by the knuckles of fingers curled under the palm, not the tips of the fingers.

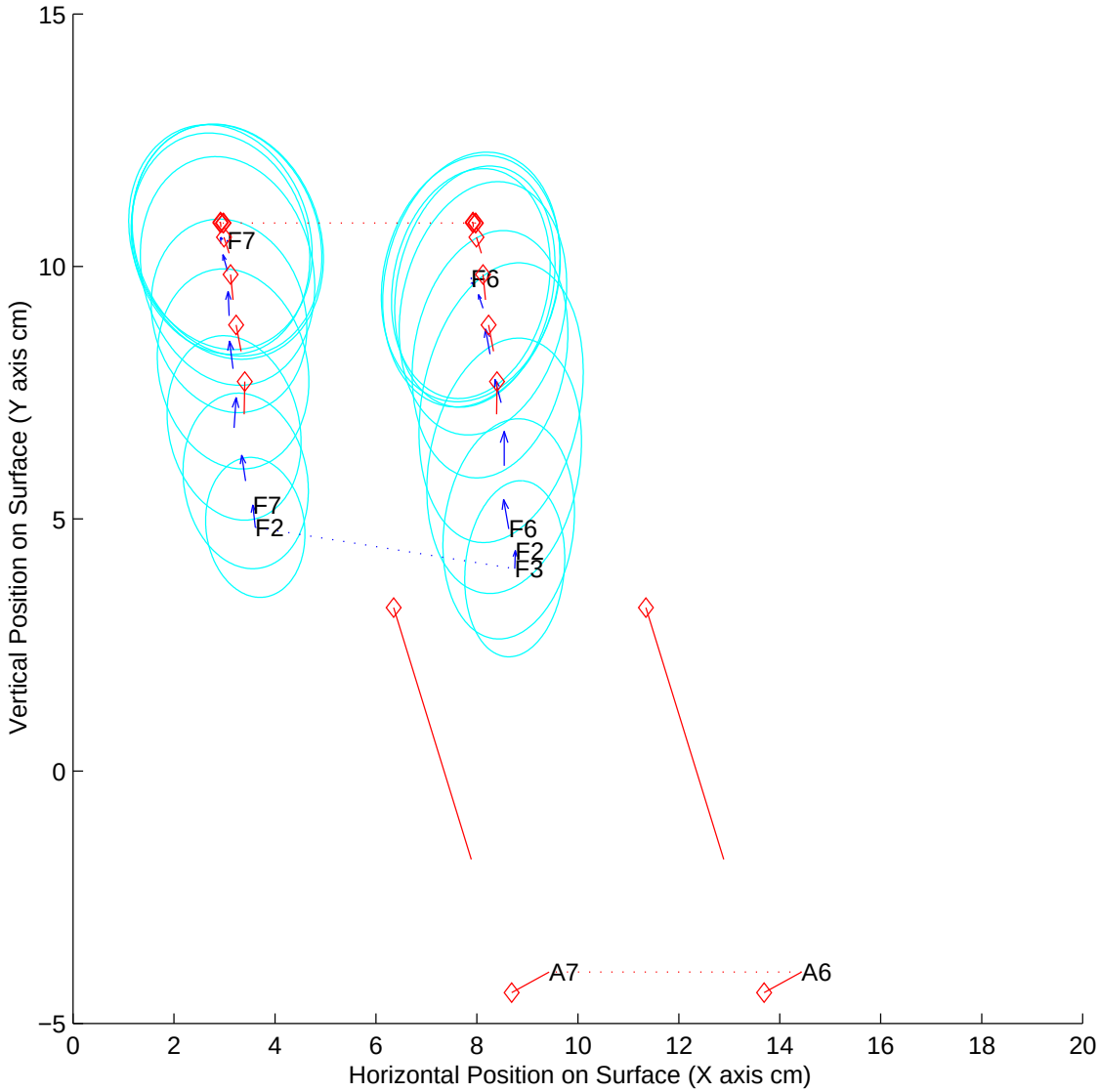


Figure 4.31: Palm heels alone are identified correctly anywhere on surface when they bottom out and reach full size. In this experiment the palms touch down gradually on the middle left and slide toward the top of the surface. Initially they are misidentified as fingertips and the palm attractors (A6,A7) begin moving left. But by the third image they grow enough that the palm heel size factor kicks in to expand their Voronoi cells over the whole surface. The identifications are corrected to F6 and F7 and the hand position estimate shoots upward, bringing the sloppy segmentation regions (Section 3.2.6.5, Page 80 and Figure 3.3 Page 70) with it to ensure the two enlarging palm heel contacts do not get split into three or four contacts.

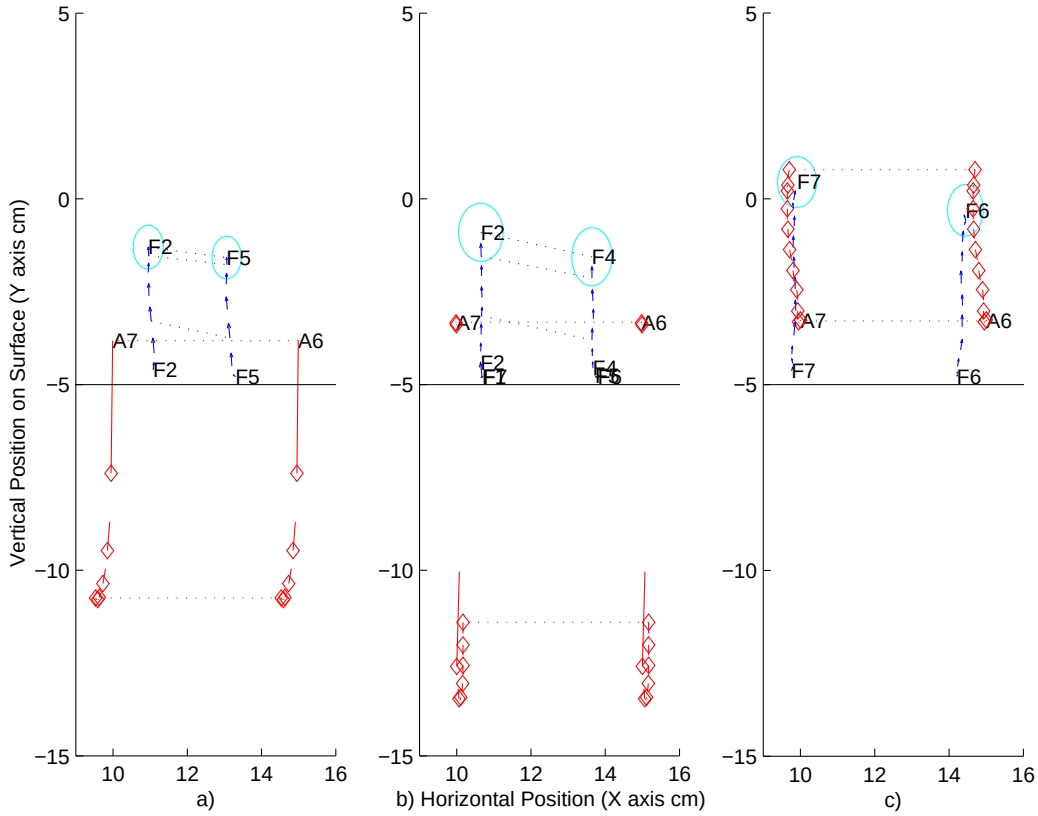


Figure 4.32: Dependency of fingertip pair identification in palm regions on fingertip separation. In each case, middle and ring fingers touch down and slide upward between the palm heel attractors near the bottom edge (horizontal black line) of the sensing area. In a) the adjacent fingertips are about 2 cm apart like normal. This causes a low palm heel separation factor which causes the palm heel Voronoi cells to vanish and the contacts to be identified as fingertips (but not adjacent ones) upon touchdown. The hand position estimate, palm heel attractors (A6,A7), and sloppy segmentation region all shoot downward to stabilize these identifications. In b) the fingertips remain separated by 3 cm but do not touch down synchronously. The first down is initially misidentified as a palm heel (F7) because the separation factor cannot be computed until both are touching. However, the identifications are soon corrected, and the hand position estimate again shoots downward only to follow the fingertips back up somewhat. In c) the fingertips are separated by 4 cm, the nominal palm heel separation, the separation factor is ineffective, the fingertips remain misidentified as palm heels, and the palm heel attractors (A6,A7) stay with them.

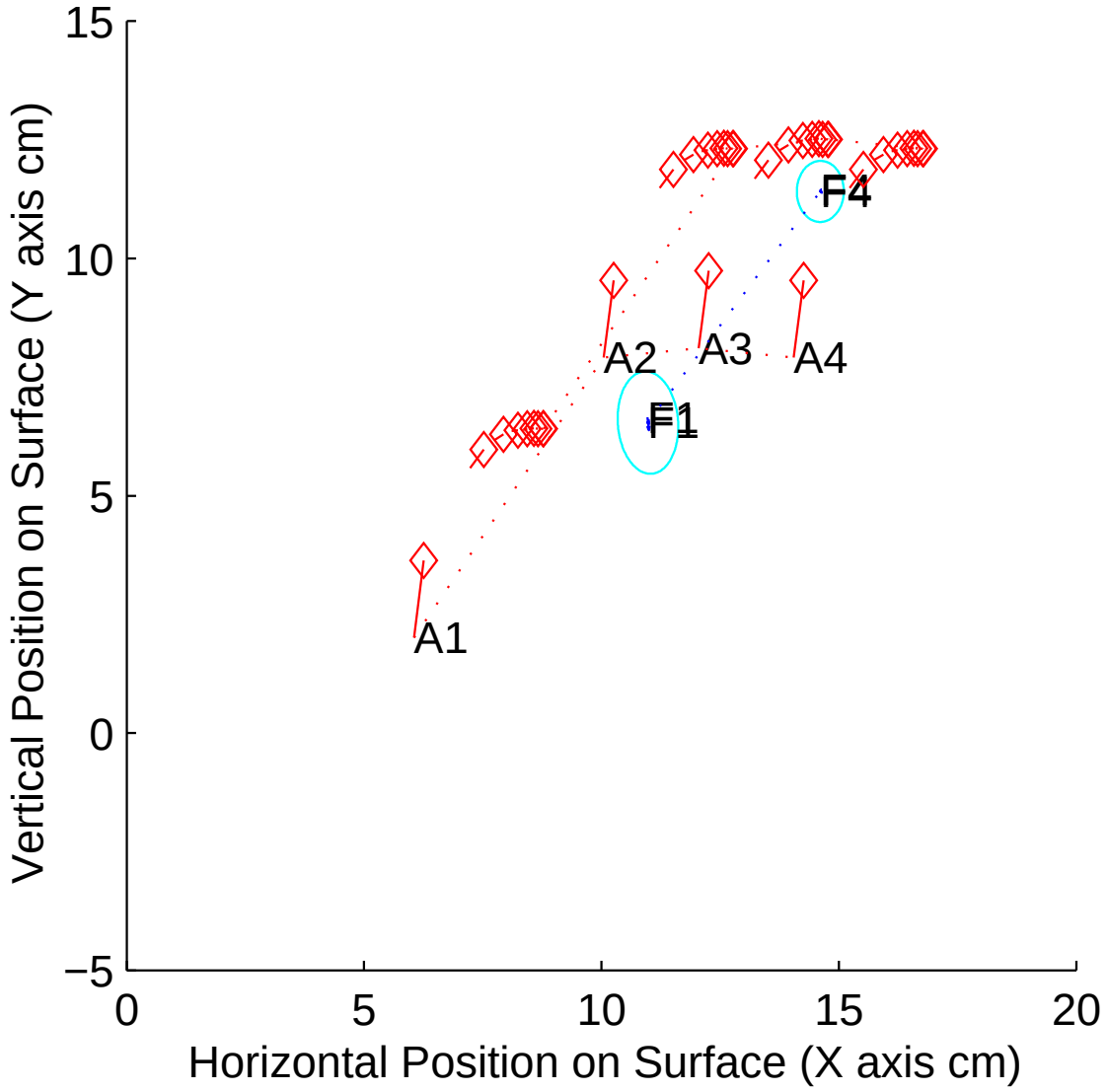


Figure 4.33: Correct thumb identification for a thumb-middle fingertip chord in the fingertip regions. Note how the thumb and fingertip attractors jump to the upper right to align with the finger contacts. The thumb verification module is able to distinguish a thumb and a fingertip from two fingertips anywhere on the surface as long as the inter-contact separation is more than about 4 cm and the inter-contact angle is not near horizontal. Without it, the assignment algorithm probably would have left these contacts identified as two fingertips. Note that because of identification ratcheting (Section 4.4.8), the fingertip identity never gets corrected from F4 to F3 even though the A3 attractor ends up nearest the fingertip.

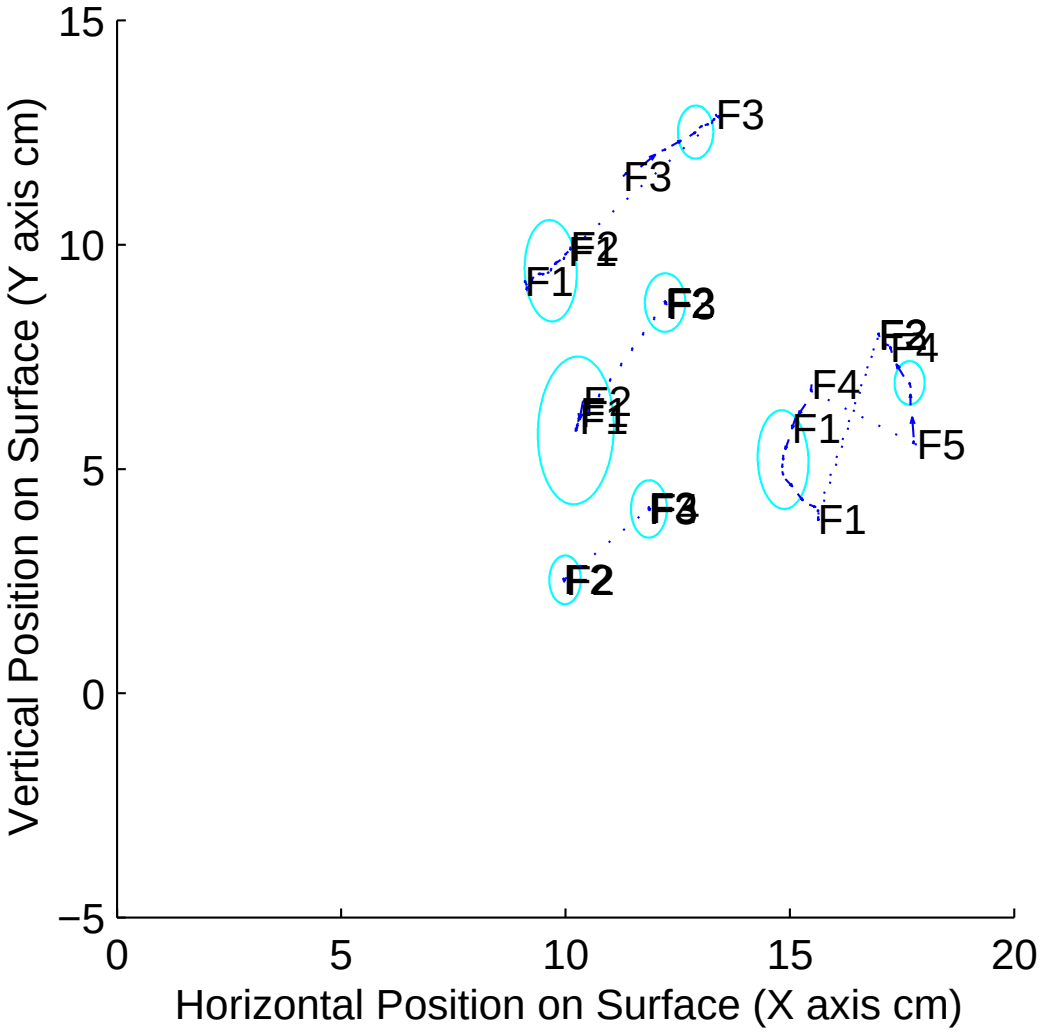


Figure 4.34: Identifications of a thumb and middle finger which do not start uniquely separated but perform unique motions. In a) at the top, the thumb is initially misidentified as F2, but the expansion factor (Section 4.4.7.3) quickly responds to the anti-pinch scaling motion between the thumb and fingertip, correcting the thumb identity to F1. In b) the thumb identification is corrected as the thumb flattens out and its contact becomes much taller than that of the fingertip. In c) only the tip of the thumb touches lightly, giving it a small contact, and it does not move, so it remains misidentified as F2. In d) the contacts are initially identified as F4 and F5, but when the thumb and pinky rotate counter-clockwise as if loosening a screw, the rotation factor (Section 4.4.7.4) detects this and corrects the thumb contact identity to F1.

by more than 4 cm, but if there are less than four fingertips touching, the fingertip identifications may be shifted improperly. Even with less than four fingertips touching, the thumb will always be identified properly as long as it is well-separated from the fingertips, flattens onto the surface when they do not, or is involved in expansive hand scaling or rotational motions about a point centered between it and the fingertips. With four fingertips plus any combination of thumb and palm heels touching, all identifications are perfect except possibly under extreme hand rotations.

4.5 Hand Identification

Hand identification is needed for multi-touch surfaces which are large enough to accomodate both hands simultaneously and which have the left and right halves of the surface joined such that a hand can roam freely across the middle to either half of the surface. The simplest method of hand identification would be to assign hand identity to each contact according to whether the contact initially touched down in the left or right half of the surface. However, if a hand touched down in the middle, straddling the left and right halves, some of the hand's contacts would end up assigned to the left hand and others to the right hand. Therefore, more sophisticated methods which take into account the clustering properties of hand contacts must be applied to ensure all contacts from the same hand get the same identity. Once all surface contacts are initially identified, the path tracking module can reliably retain existing identifications as a hand slides from one side of the surface to the other.

The thumb and inner palm contact orientations and the relative thumb placement are the only contact features independent of cluster position which distinguish a lone cluster of right hand contacts from a cluster of left hand contacts. If the thumb is lifted off the surface, a right hand contact cluster appears indistinguishable from a left hand cluster. In this case cluster identification must still depend heavily on which side of the board the cluster starts on, but the identity of contacts which

recently lifted off nearby also proves helpful. For example, if the right hand moves from the right side to the middle of the surface and lifts off, the next contacts which appear in the middle will most likely be from the right hand touching back down, not from the left hand moving to the middle and displacing the right hand. The division between left and right halves of the surface should therefore be dynamic, shifting toward the right or left according to which hand was most recently near the middle. Since the hand offset estimates temporarily retain the last known hand positions after liftoff, such a dynamic division is implemented by tying the positions of left hand and right hand attractor templates to the estimated hand positions.

Though cases remain in which the operator can fool the hand identification system with sudden placements of a hand in unexpected locations, the operator may actually wish to fool the system in these cases. For example, operators with only one hand free to use the surface may intentionally place that hand far onto the opposite half of the surface to access the chord input operations of the opposite hand. Therefore, when a hand cluster suddenly touches down well into the opposite half of the surface, it can safely be given the opposite half's identity, regardless of its true identity. Arching the surface across the middle can also discourage users from sliding a hand to the opposite side by causing awkward forearm pronation should users do so.

4.5.1 Checking for Contact Stabilization

Figure 4.35 shows process details within the hand identification module. The MTS first determines whether the hand identification algorithm actually needs to be executed for the current sensor array scan cycle by checking whether all path proximities have stabilized. To maximize stability of the identifications, hand and finger identities need only be reevaluated when a new hand part touches down or disambiguating features of existing contacts become stronger. The contact size and orientation features are unreliable until the flesh fully compresses against the

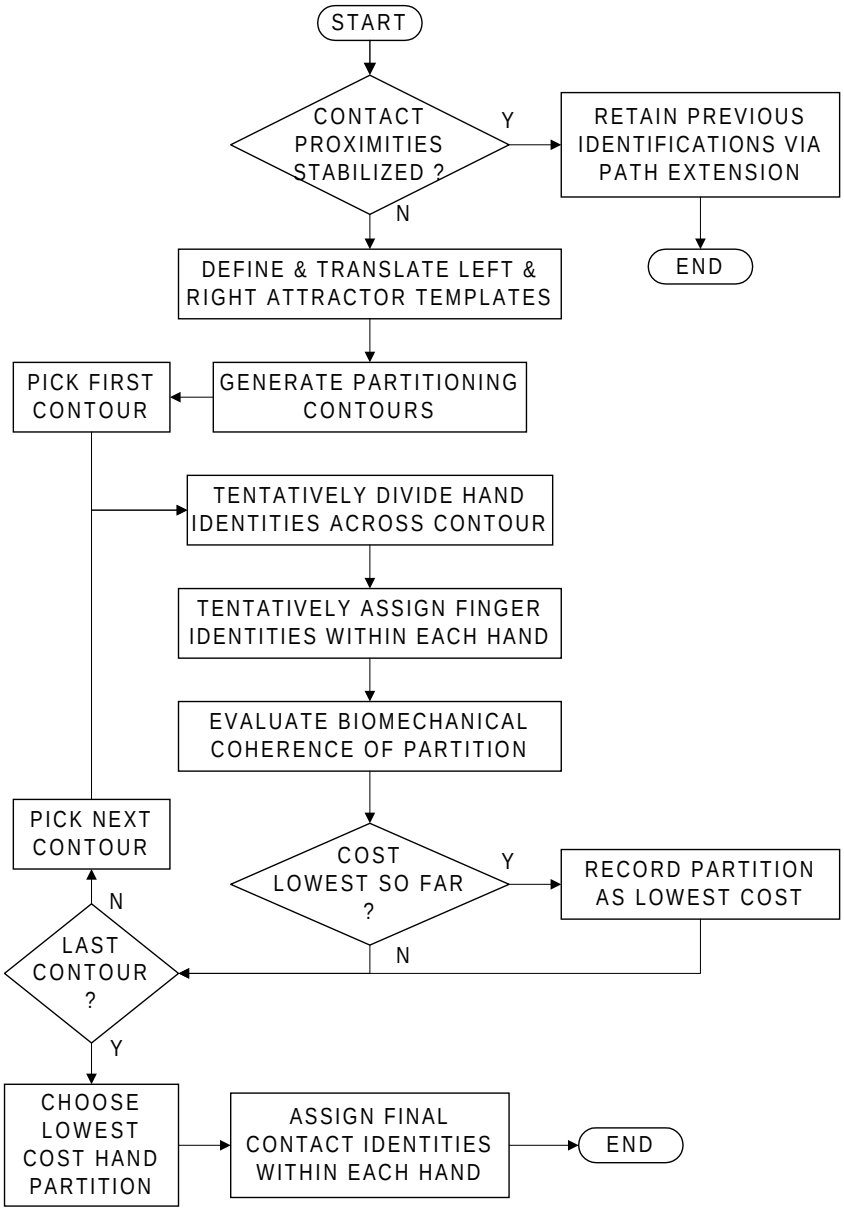


Figure 4.35: Flow chart of the hand identification algorithm.

surface a few dozen milliseconds after initial surface contact. Therefore, the hand identification algorithm executes for each proximity image in which a new contact appears and for subsequent proximity images in which the total proximity of any new contacts continues to increase. For images in which proximities of existing contacts have stabilized and no new contacts appear, path continuation as performed by the path tracking process (Section 3.3) is sufficient to retain and extend the contact identifications computed from previous images.

4.5.2 Placing Left and Right Attractor Rings

Should the hand identification algorithm be invoked for the current image, the first step is to define and position left and right hand attractor templates. These should be basically the same as the attractor templates (Figure 4.5) used for finger identification, except that both left and right rings must now be utilized at once. The default placement of the rings relative to one another should correspond to the default left and right hand contact positions shown in Figure 3.3a. Each ring translates to follow the estimated position of its hand, just like the sloppy segmentation regions follow the hands in Figure 3.3b. Individual attractor points can safely be translated by their corresponding estimated finger offsets. Therefore the final attractor positions $(Aj_x[n], Aj_y[n])$ for the left hand L and right hand R attractor rings are:

$$LAj_x[n] = LH_{eox}[n] + LFj_{eox}[n] + LFj_{defx} \quad (4.52)$$

$$LAj_y[n] = LH_{eoy}[n] + LFj_{eoy}[n] + LFj_{defy} \quad (4.53)$$

$$RAj_x[n] = RH_{eox}[n] + RFj_{eox}[n] + RFj_{defx} \quad (4.54)$$

$$RAj_y[n] = RH_{eoy}[n] + RFj_{eoy}[n] + RFj_{defy} \quad (4.55)$$

Basically the hand identification algorithm will compare the cost of assigning contacts to attractors in one ring versus the other, the cost depending on the sum of weighted distances between each contact and its assigned attractor. Adjusting the

attractor ring with the estimated hand and finger offsets lowers the relative costs for assignment hypotheses which resemble recent hand assignments, helping to stabilize identifications across successive proximity images even when hands temporarily lift off.

4.5.3 Generating Plausible Partition Hypotheses

Next a set of assignment hypotheses must be generated and compared. The most efficient way to generate sensible hypotheses is to define a set of roughly vertical contour lines, one between each horizontally adjacent contact. This is done by ordering all surface contacts by their horizontal coordinates and establishing a vertical contour halfway between each pair of adjacent horizontal coordinates. Figures 4.36a–c show examples of three different contours and their associated assignment hypotheses for a fixed set of contacts. Each contour corresponds to a separate hypothesis, known also as a partition, in which all contacts to the left of the contour are from the left hand, and all contacts to the right of the contour are from the right hand. Contours are also necessary at the left and right ends of the surface to handle the hypotheses that all contacts on the surface are from the same hand. Contours which hypothesize more contacts on a given hand than can be caused by a single hand are immediately eliminated.

Generating partitions via vertical contours avoids all hypotheses in which contacts of one hand horizontally overlap or cross over contacts of the opposite hand. Considering that each hand can cause seven or more distinct contacts, this reduces the number of hand identity permutations to examine from thousands to at most a dozen. With fewer hypotheses to examine, the evaluation of each partition can be much more sophisticated, and if necessary, computationally costly.

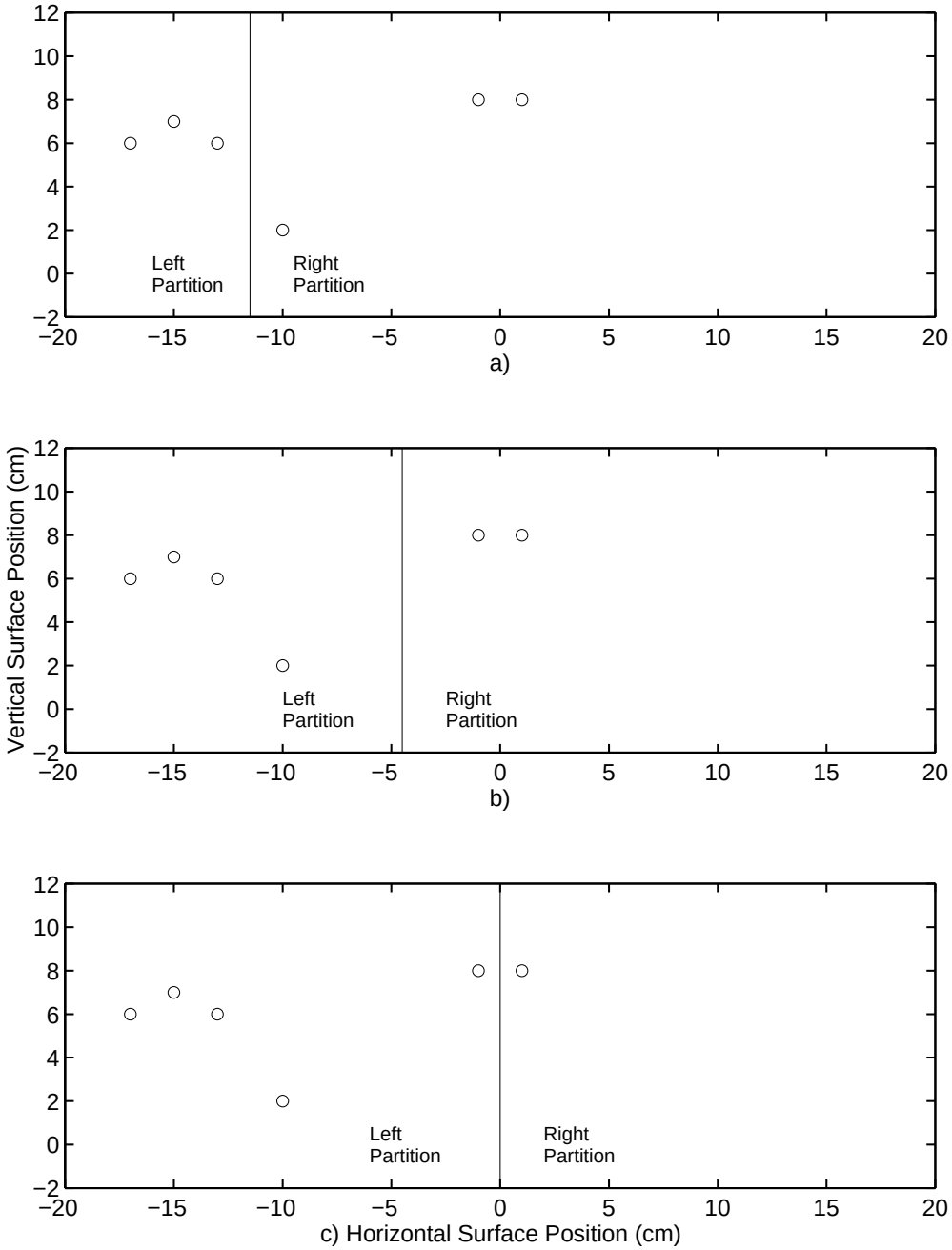


Figure 4.36: Vertical contours (dividing lines) creating three different partitioning hypotheses. Contours are always placed halfway between horizontally adjacent contacts (circles). The partitioning of b) is probably the correct partitioning for this arrangement of contacts, meaning that the two contacts in the middle of the surface are from the right hand, and the rest are from the left hand.

4.5.4 The Optimization Search Loop

The goal of the optimization search is to determine which of the contours partitions the contacts into left hand and right hand clusters such that the cluster positions and contact arrangements within clusters best satisfy known anatomical and biomechanical constraints. The optimization begins by picking a first contour divider such as the leftmost and tentatively assigning any contacts to the left of the contour to the left hand and the rest to the right hand. The finger identification algorithm (Figure 4.4) then attempts to assign finger and palm identities to contacts within each hand.

Returning to Figure 4.35, the next step is to compute a cost for the partition. This cost is meant to evaluate how well the tentatively identified contacts fit their assigned attractor ring and how well the partition meets between-hand separation constraints. This is done by computing for each hand the sum of weighted distances from each tentatively identified contact to its assigned attractor point as in Equation 4.44 of finger identification, including size and orientation feature factors for thumb and palm attractors. This sum represents the basic template fitting cost for a hand. Each hand cost is then weighted as a whole with the reciprocals of its clutching velocity, handedness, and palm cohesion factors. These factors, to be described below, represent additional constraints which are underemphasized by the weighted attractor distances. Finally, the weighted left and right hand costs are added together and scaled by the reciprocal of a hand separation factor to obtain a total cost for the partition.

This process is repeated for each partitioning contour until the costs of all hypothesized partitions have been evaluated. The partition which has the lowest cost overall is chosen as the actual hand partitioning, and the hand identities of all contact paths are updated accordingly. The within-hand or finger identification algorithm is invoked once more so that the thumb verification and statistics gathering

processes can execute using the actual hand assignments.

4.5.5 Partition Cost Modifiers

4.5.5.1 Clutching Direction Factor

Users often perform clutching motions in which the right hand, for example, lifts off from a slide at the right side of the surface, touches back down in the middle of the surface, and resumes sliding toward the right. Therefore when a hand is detected touching down in the middle of the surface and sliding toward one side, it probably came from that side. A hand clutching direction factor, plotted approximately in Figure 4.37, captures this phenomenon by slightly increasing in

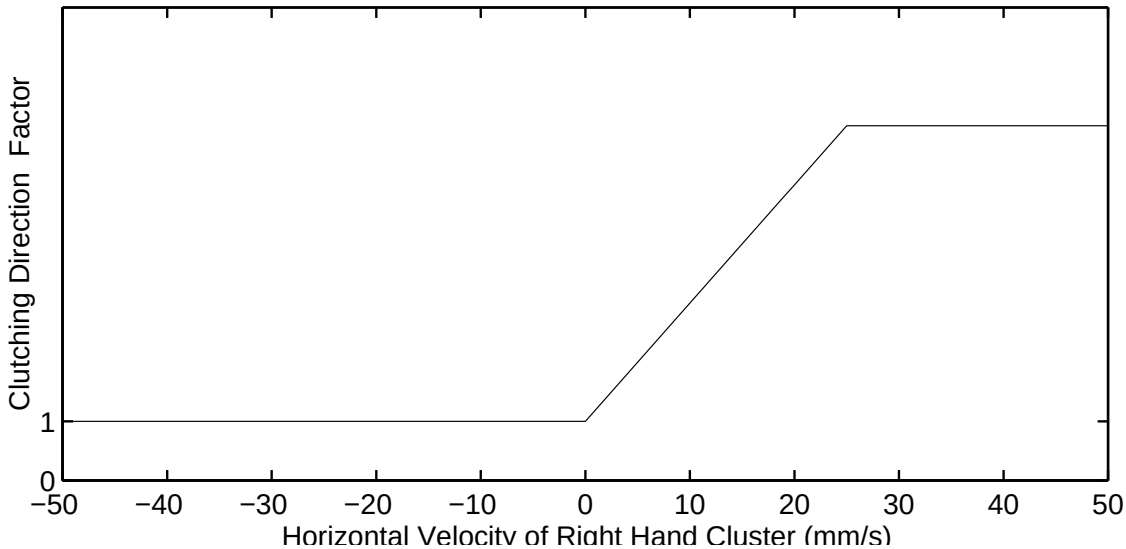


Figure 4.37: Hand clutching direction factor versus the average of the right hand's horizontal contact velocities.

value when a hand cluster's contacts are moving toward the cluster's assigned side of the board, thus decreasing the basic cost of the hand. The factor is a function of the average of the contacts' horizontal velocities and the side of the surface the given cluster is assigned. Since high speeds do not necessarily give a stronger indication of user intent, the factor saturates at moderate speeds.

4.5.5.2 Handedness Factor

Though the thumb orientation factors help identify which hand a thumb is from when the thumb lies in the ambiguous middle region of the surface, the vertical position of the thumb relative to other fingers in the same hand also gives a strong indication of handedness. The thumb tends to be positioned much lower than the fingertips, but the pinky tends to be only slightly lower than the other fingertips. The handedness factor, plotted approximately in Figure 4.38, takes advantage of this

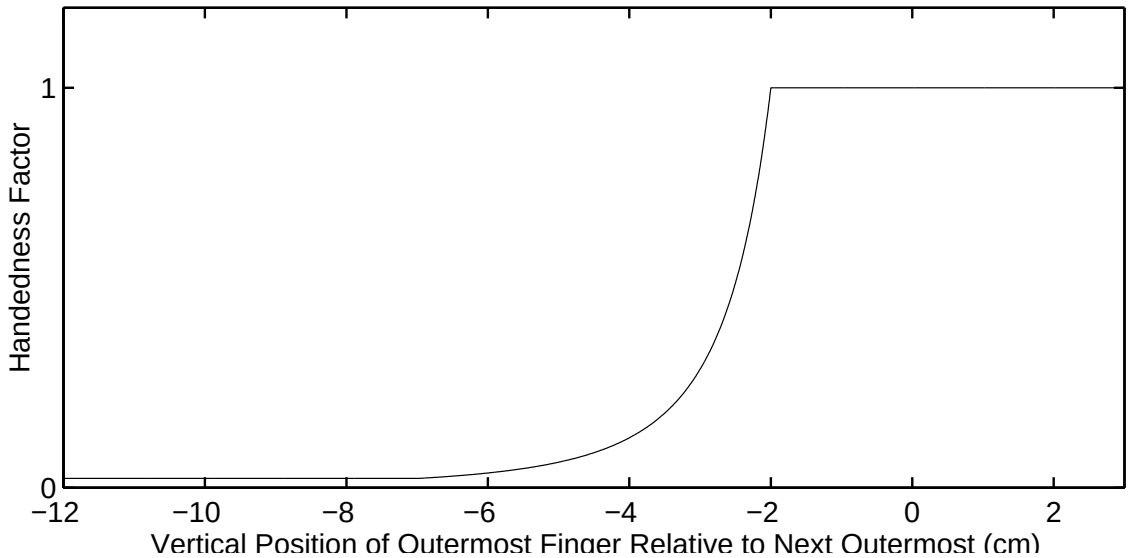


Figure 4.38: Handedness factor versus the vertical separation between outermost and next outermost finger contacts.

constraint by boosting the hand cost when the contact identified as the outermost fingertip is more than a couple centimeters lower than the next outermost fingertip contact. In such cases the tentative hand assignment for all contacts in the cluster is probably wrong. Since this causes the within-hand identification algorithm to fit the contacts to the wrong attractor ring, finger identities become reversed such that the supposedly lowered pinky is truly a lowered thumb of the opposite hand. Unfortunately, limited confidence can be placed in the handedness factor. Though the pinky should not appear lowered as much as the thumb, the outer palm heel

can, creating an ambiguity in which the thumb and fingertips of one hand have the same contact arrangement as the fingertips and outer palm heel of the opposite hand (Figure 4.45). This ambiguity can cause the handedness factor to be erroneously low for an accurately identified hand cluster, so the handedness factor is only used on clusters in the middle of the surface where hand position is ambiguous.

4.5.5.3 Palm Cohesion Factor

Distinguishing contact clusters is challenging because a cluster can become quite sparse and large when the fingers are outstretched, with the pinky and thumb of the same hand spanning up to 20cm. However, the palm can stretch very little in comparison, placing useful constraints on how far apart palm heel contacts and forepalms from the same hand can be. The entire palm region of an outstretched adult hand is about 10 cm square, so palm contact centroids should not be scattered over a region larger than about 8 cm. When a partition wrongly includes fingers from the opposite hand in a cluster, the within-cluster identification algorithm tends to assign the extra fingers from the opposite hand to palm heel and forepalm attractors. This usually causes the contacts assigned to the cluster's palm attractors to be scattered across the surface wider than is plausible for true palm contacts from a single hand. To punish such partitions, the palm cohesion factor quickly drops below one for a tentative hand cluster in which the supposed palm contacts are scattered over a region larger than 8 cm. Therefore its reciprocal will greatly increase the hand's basic cost. Figure 4.39 shows the value of the palm cohesion factor versus horizontal separation between palm contacts. The horizontal spread can be efficiently measured by finding the maximum and minimum horizontal coordinates of all contacts identified as palm heels or forepalms and taking the difference between the maximum and minimum. The measurement and factor value lookup are repeated for the vertical separation, and the horizontal and vertical factors are multiplicatively combined to obtain the final palm cohesion factor.

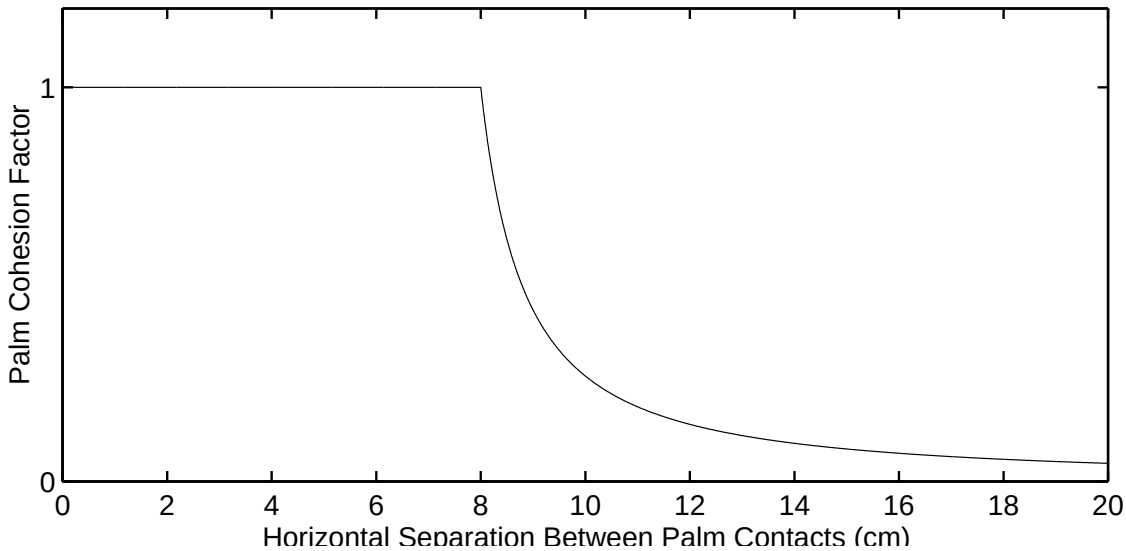


Figure 4.39: Palm cohesion factor versus the horizontal separation between the innermost and outermost contacts identified as palms.

4.5.5.4 Inter-Hand Separation Factor

Figure 4.40 is an approximate plot of the inter-hand separation factor. This factor increases the total costs of partitions in which the estimated or actual horizontal positions of the thumbs from each hand approach or overlap. It is measured by finding the minimum of the horizontal offsets of right hand contacts with respect to their corresponding default finger positions. Similarly the maximum of the horizontal offsets of the left hand contacts with respect to their corresponding default finger positions is found. If the difference between these hand offset extrema is small enough to suggest the thumbs are overlapping the same columnar region of the surface while either touching the surface or floating above it, the separation factor becomes very small. Such overlap corresponds to a negative thumb separation in the plot. To encourage assignment of contacts which are within a couple centimeters of one another to the same cluster, the separation factor gradually begins to drop starting with positive separations of a few centimeters or less. The inter-hand separation factor is not applicable to partitions in which all surface contacts are

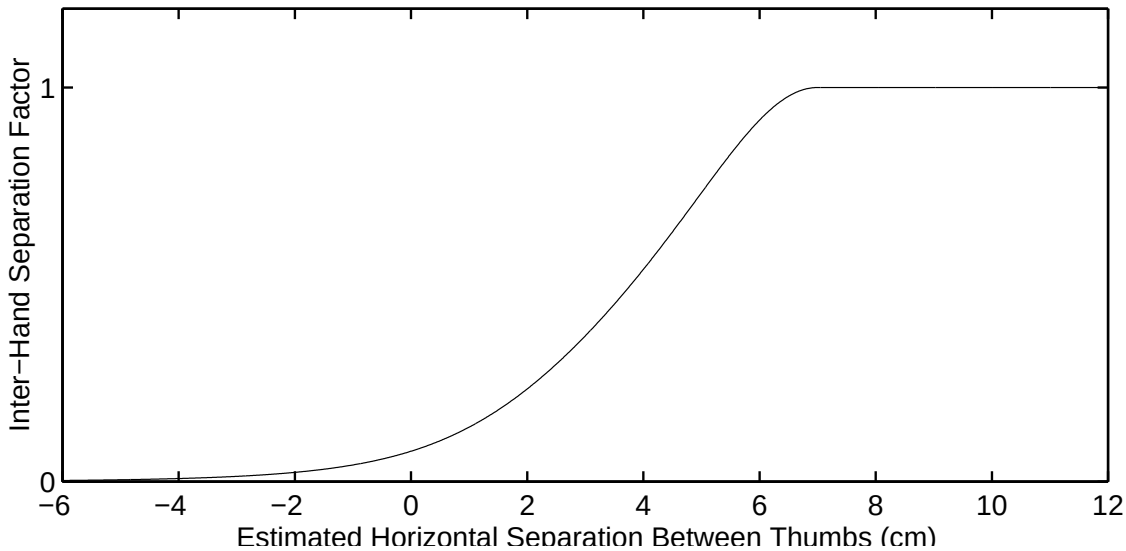


Figure 4.40: Inter-hand separation factor versus the estimated distance between the left and right thumbs.

assigned to the same hand, and takes on the default value of one in this case.

4.5.6 Hand Identification Results

Hand identification results (Figures 4.41–4.48) are presented the same way as finger identification results (Section 4.4.9) except both halves of the surface and both hands are shown. To distinguish fingers and attractors from left and right hands, finger and attractor labels are preceded with an 'L' or an 'R.' As in the finger identification experiments, each experiment starts with the estimated hand positions in their default positions on opposite sides of the board.

The hand identification results can be summarized as follows. Parts from a hand which slides to the opposite side of the board and lifts off will be identified correctly if the hand touches back down within a couple seconds (Figure 4.41). Parts of a hand which touch down in the middle of the surface will always be clustered together properly and will be attributed to the correct hand if the thumb is present or if the cluster slides quickly to the correct hand's side of the board

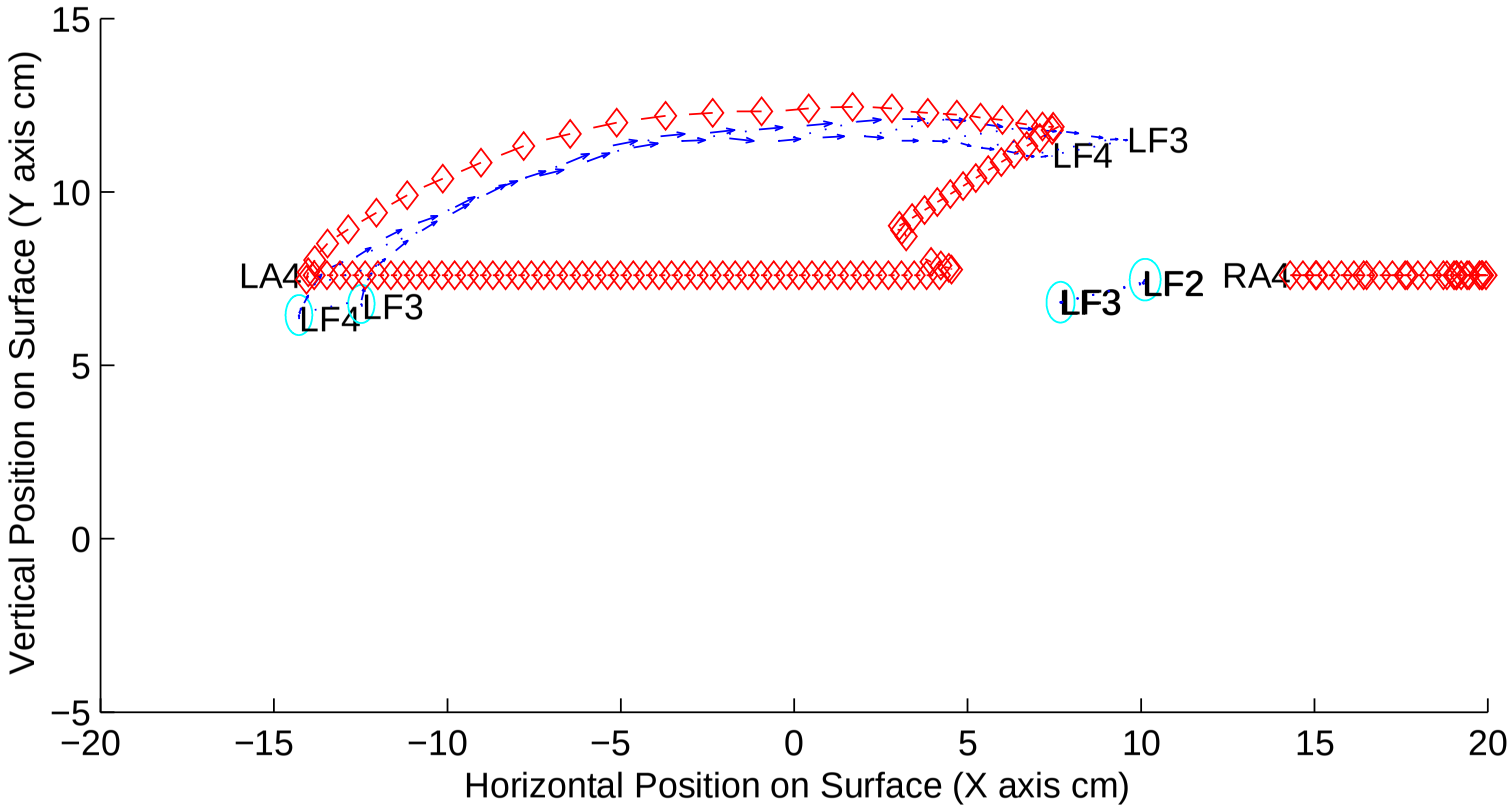


Figure 4.41: Short-term memory of hand identity as maintained by the hand position estimate. In this experiment, a left hand finger pair starts in left default position and slides well into the right side of the surface, then lifts off for a second and taps the surface a few cm lower. Because the left hand position estimate follows the fingers, brings left hand attractors (*e.g.* LA4) along, and eventually pushes right hand attractors (*e.g.* RA4) off the right edge, the tapping finger pair is correctly attributed to the left hand even though its temporary liftoff and downward shift caused a break in path tracking. Note that during the temporary liftoff, LA4 began drifting back to the left side, then swerved back right during the tap before drifting all the way back to default after final pair liftoff.

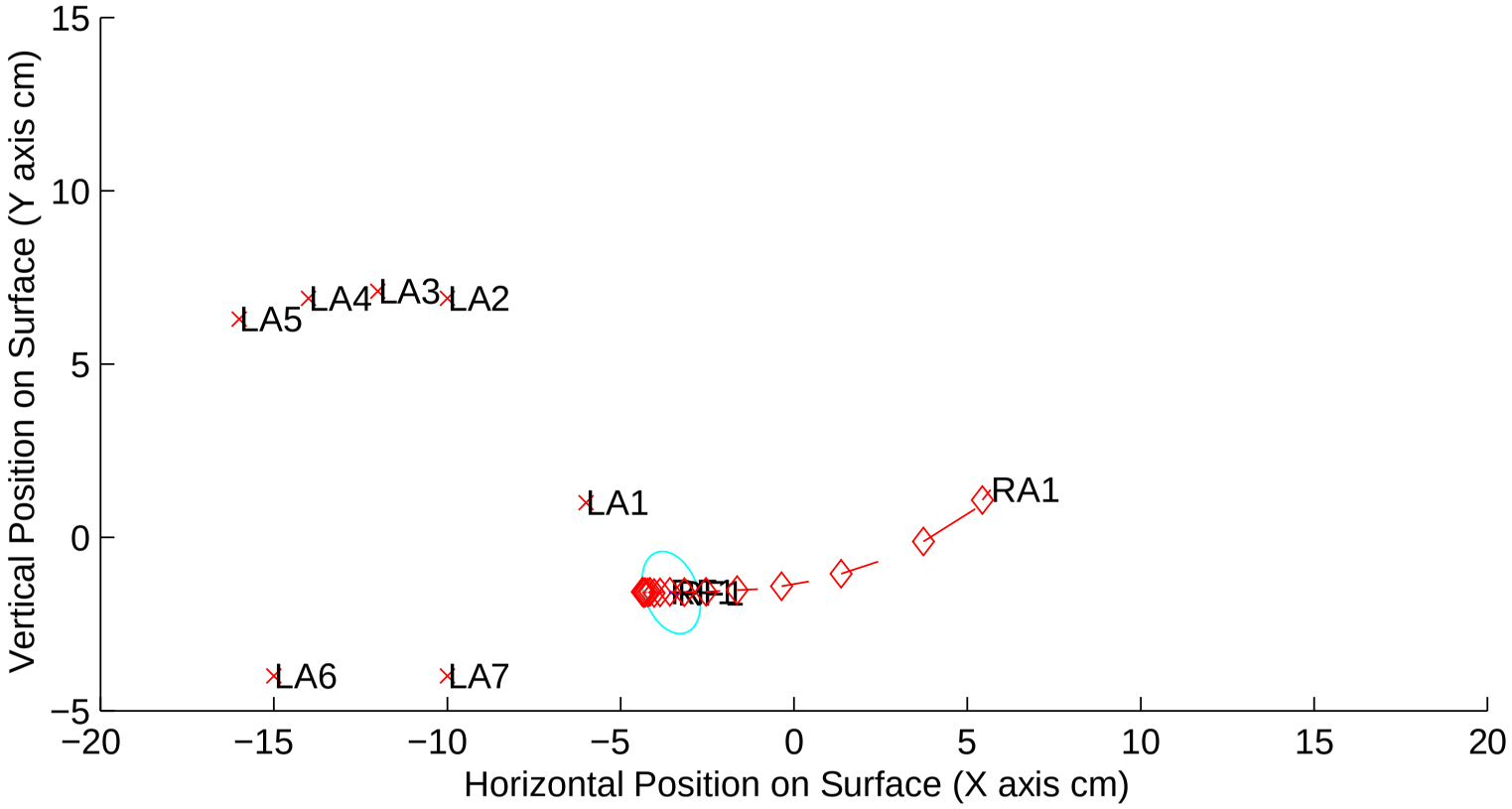


Figure 4.42: A right thumb (RF1) placed in the left middle of the surface is correctly identified solely by virtue of its orientation. Note that the contact ellipse's major angle is about 120°. Even though this right thumb is placed quite near the left thumb attractor (LA1), the right thumb orientation factor (Section 4.4.6.4) is strong enough to override attraction by LA1 and cause immediate attribution to the right hand upon touchdown. The right hand position estimate moves left in response, bringing the right thumb attractor (RA1) with it to stabilize the identification should the unique orientation information disappear in future time steps. Similarly, a lone contact with a 60° orientation placed in the right middle of the surface would be identified as the left thumb.

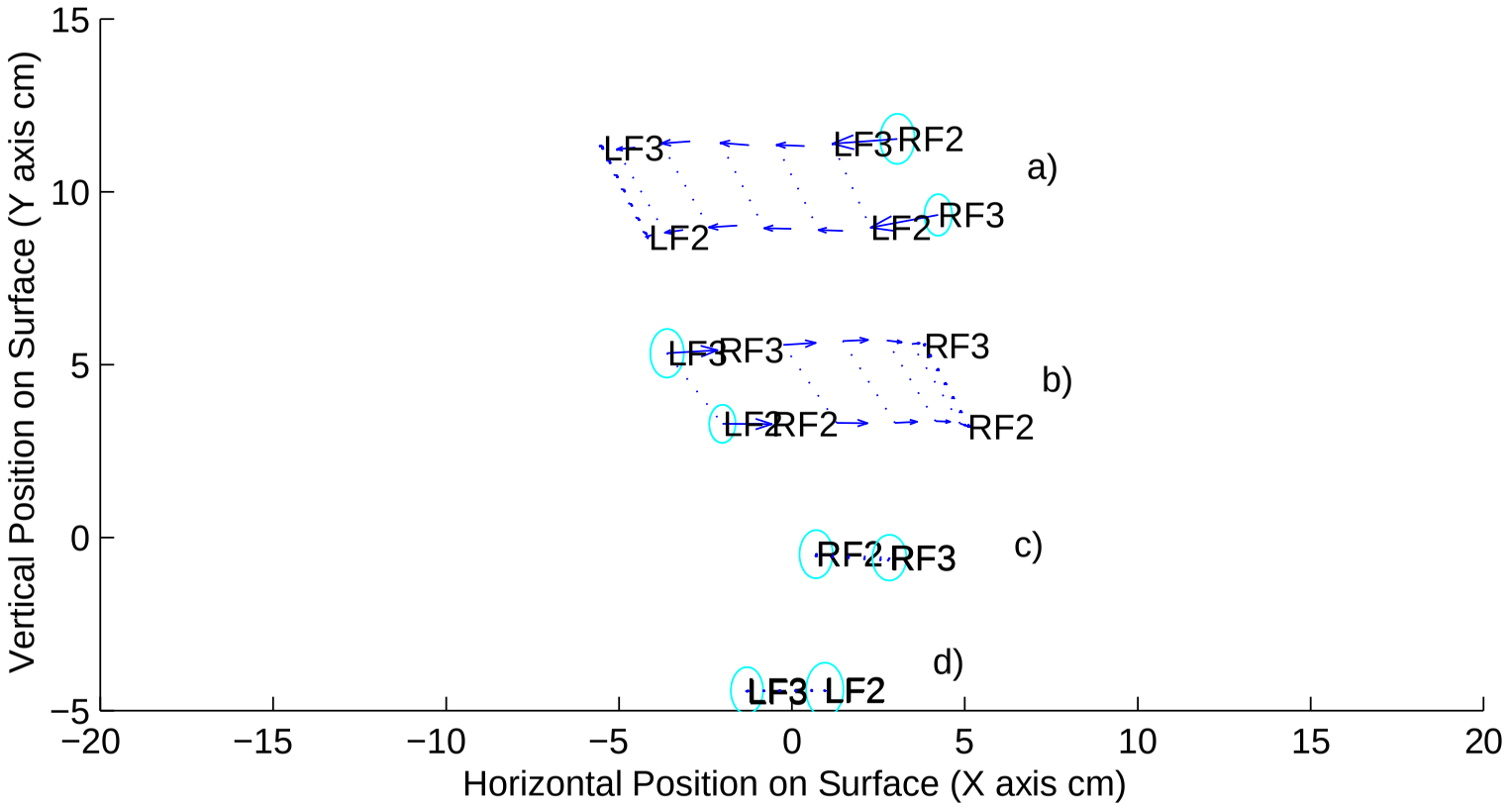


Figure 4.43: Effect of clutching velocity factor on hand identification near middle of surface. For the stationary fingertip pairs in c) and d), both fingertips get the same hand identity but the pair identity depends on whether the pair touches down just to the left d) or right c) of board center. Without the inter-hand separation factor, the fingertip pairs in c) and d) would be misidentified as left and right thumbs. However, in a) (top) the pair touches down on the right half but upon detection of its leftward velocity the clutching velocity factor causes it to be reattributed to the left hand. The opposite occurs in b), with initial identification as left fingers but a quick reversal to identification as right fingers.

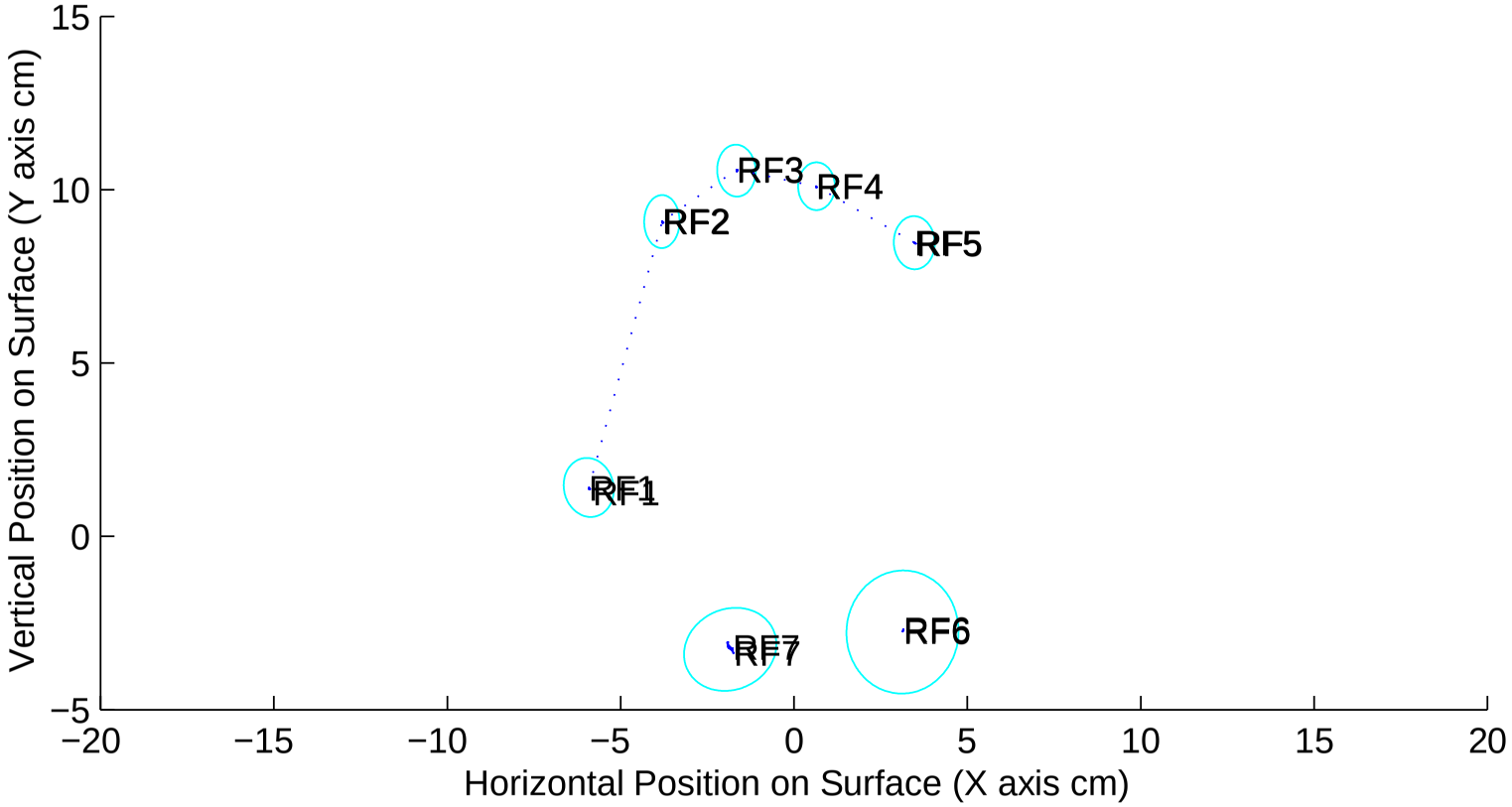


Figure 4.44: A right hand touches down straddling the middle of the surface. The inter-hand separation factor ensures all contacts are attributed to the same hand, and since the thumb is present, the thumb orientation and handedness factors ensure the contacts are correctly identified as a right hand cluster. However, if the thumb had not been present, the cluster would have had a 50-50 chance of being attributed to the left hand since the fingertips and palm heels straddle the middle.

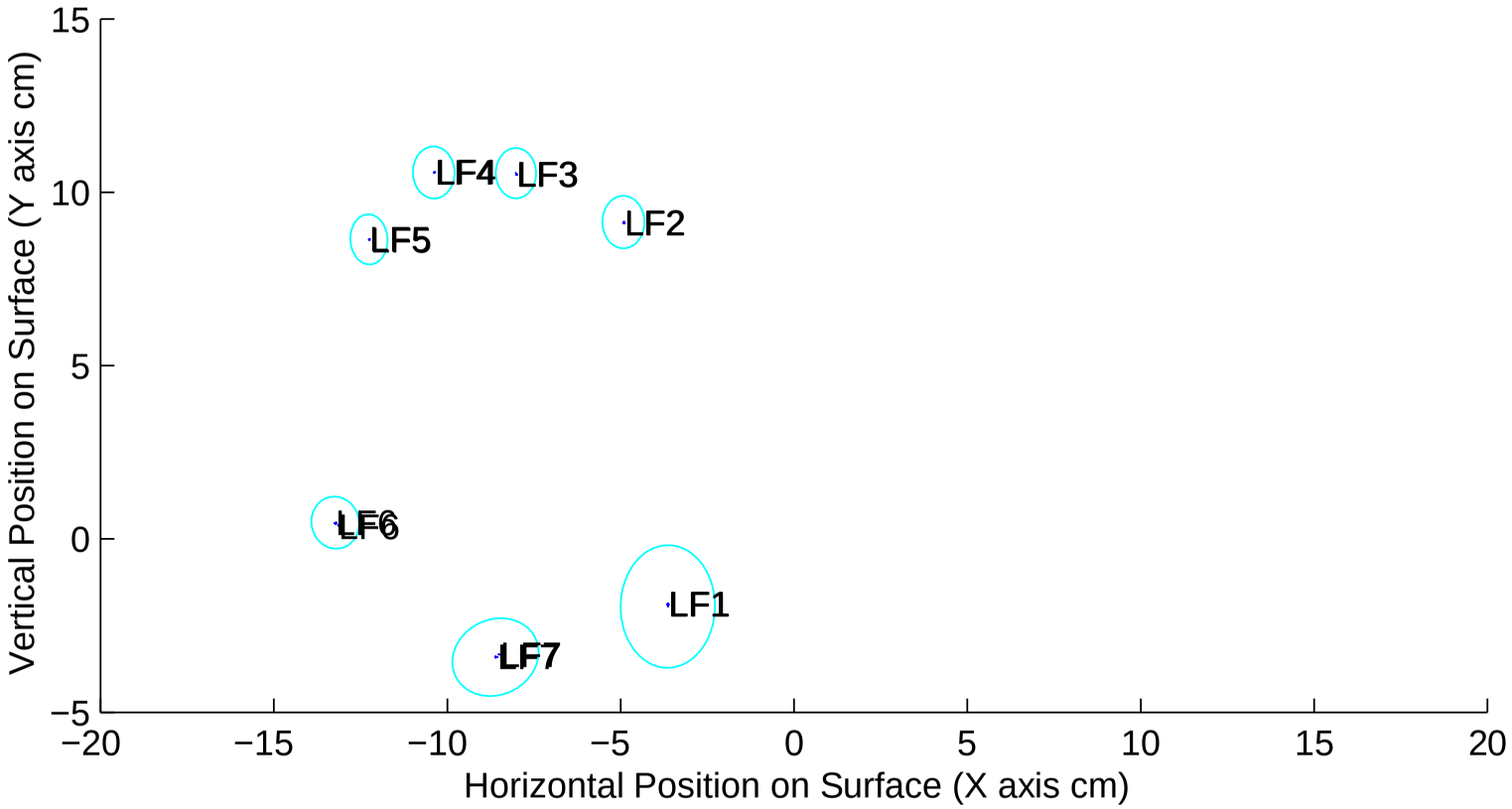


Figure 4.45: A right hand touches down well into the left half of the surface and is misidentified as the left hand. To avoid potentially destabilizing ambiguities, the strength of a hand's thumb orientation and handedness factors is intentionally limited at the far side, and in this case they are unable to override the nearness of the left hand attractors. The only thing that could possibly distinguish this right hand contact cluster from a left hand cluster is the relatively large size of the outer palm heel contact (erroneously labeled LF1) compared to the thumb (erroneously labeled LF6), but this comparison is only distinguishing when the weight of the hand fully rests on the outer palm heel.

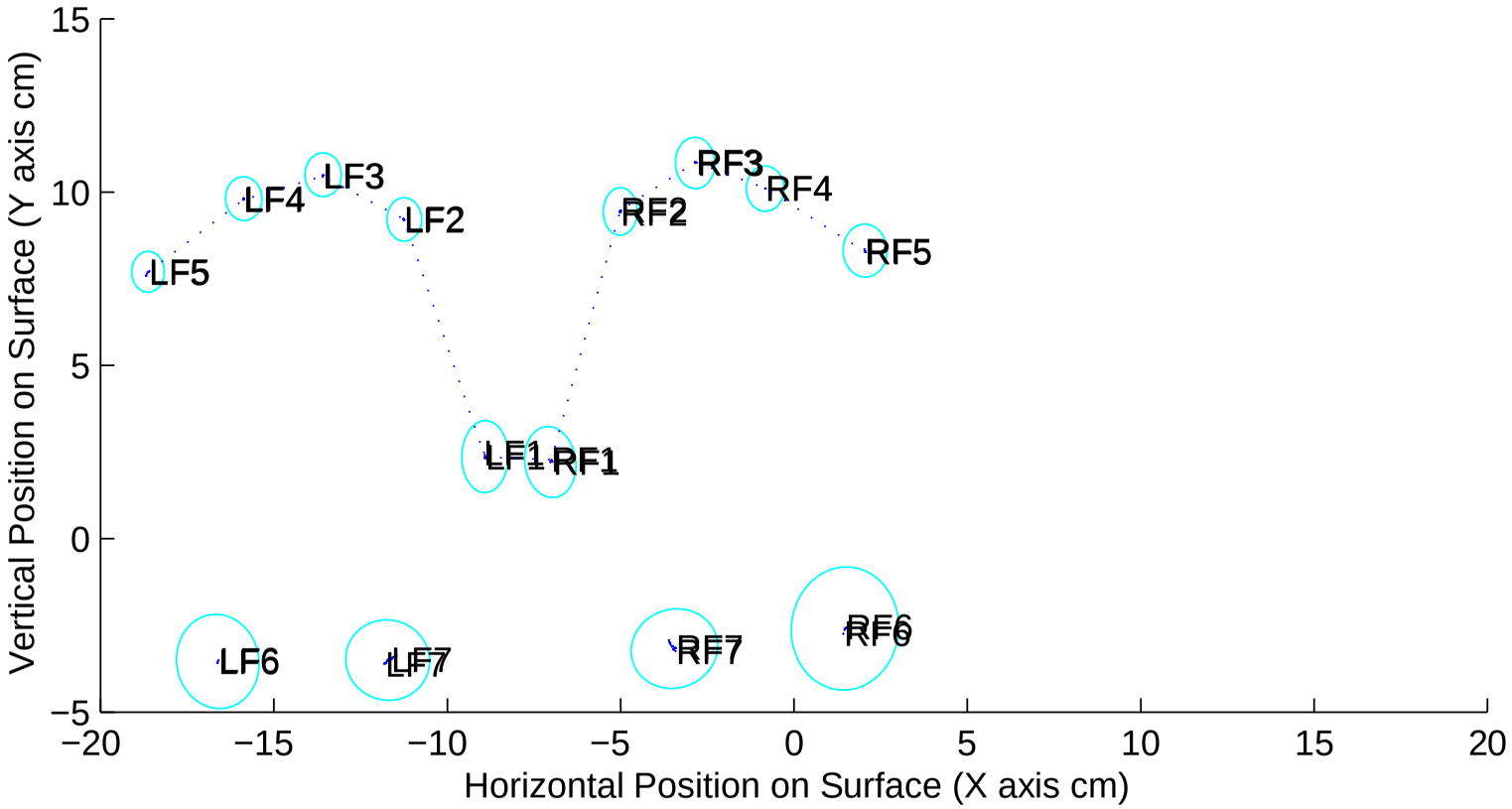


Figure 4.46: Two full hands placed side by side on the left half of the surface are partitioned correctly, the clusters being split right between the thumbs. The inter-hand separation factor, handedness factors, thumb orientation factors, and palm cohesion factors all compete to provide this correct result.

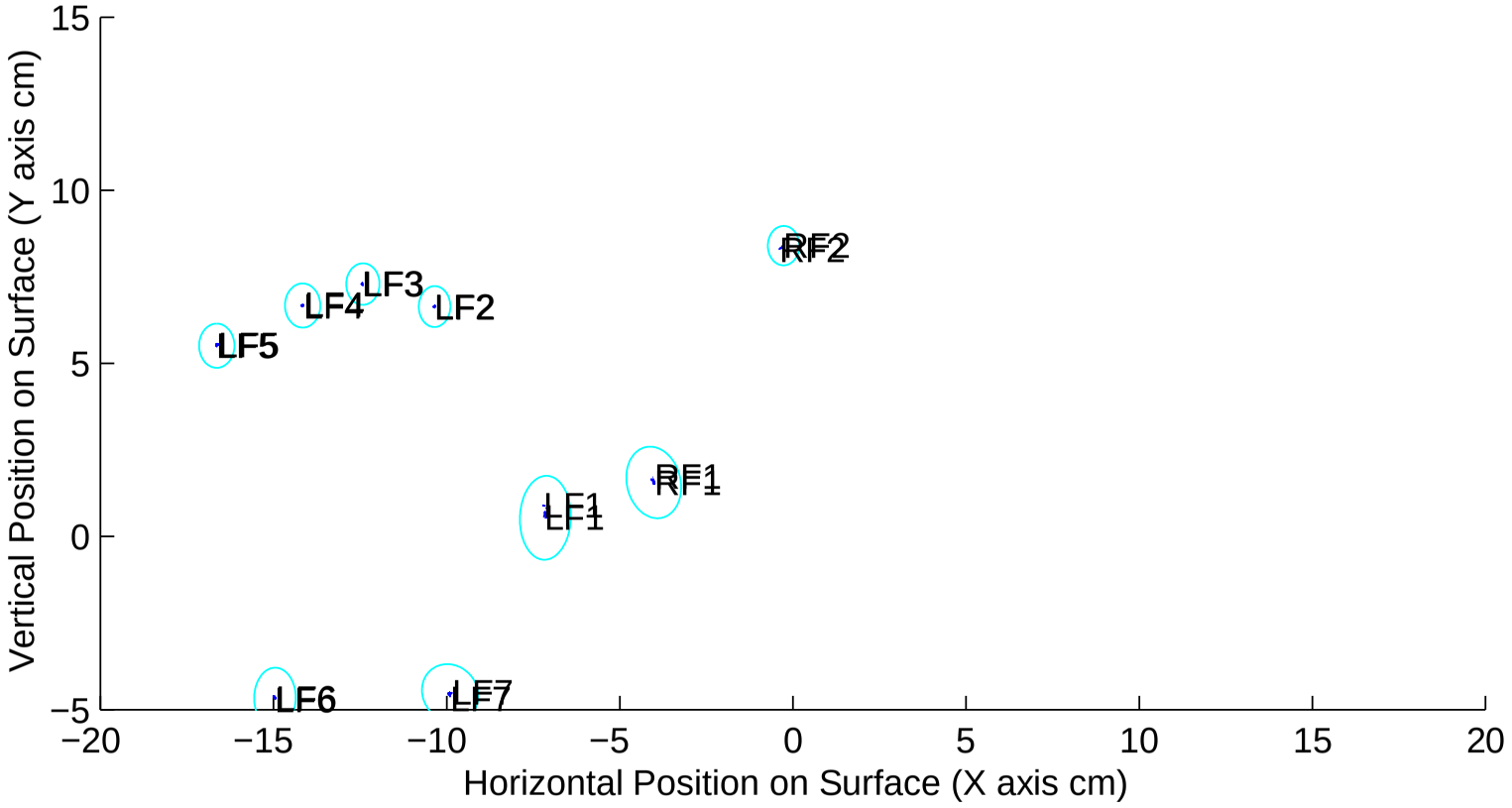


Figure 4.47: The entire left hand plus the right thumb and middle finger placed side by side on the left half of the surface are partitioned correctly, the clusters being split right between the thumbs. The inter-hand separation factor, handedness factors, thumb orientation factors, and palm cohesion factors all compete to provide this correct result. However, the presence of the left palm heels is critical to effectiveness of the palm cohesion factor, as will be seen when the palm heels do not touch in the Figure 4.48.

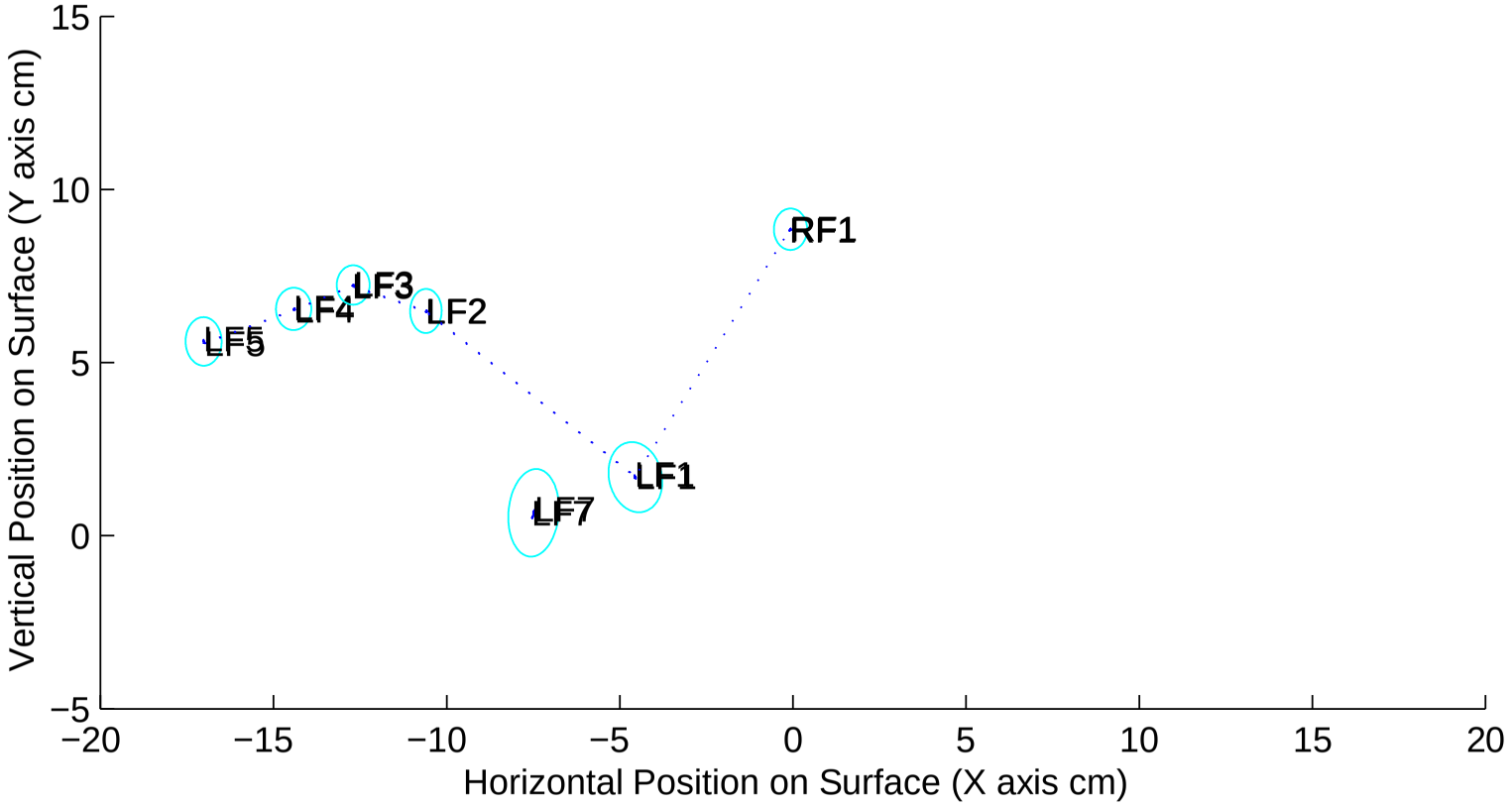


Figure 4.48: The five left hand fingers plus the right thumb and middle finger placed on the left half of the surface are not partitioned correctly. Since the left palm heels do not actually touch and therefore leave their attractors open, the system finds a bad partition in which the left thumb is misidentified as the left inner palm heel (LF7) and the right thumb is misidentified as the left thumb (LF1). Since only one contact is attributed to a palm heel, the palm cohesion cannot be measured, and its weighting factor is ineffective.

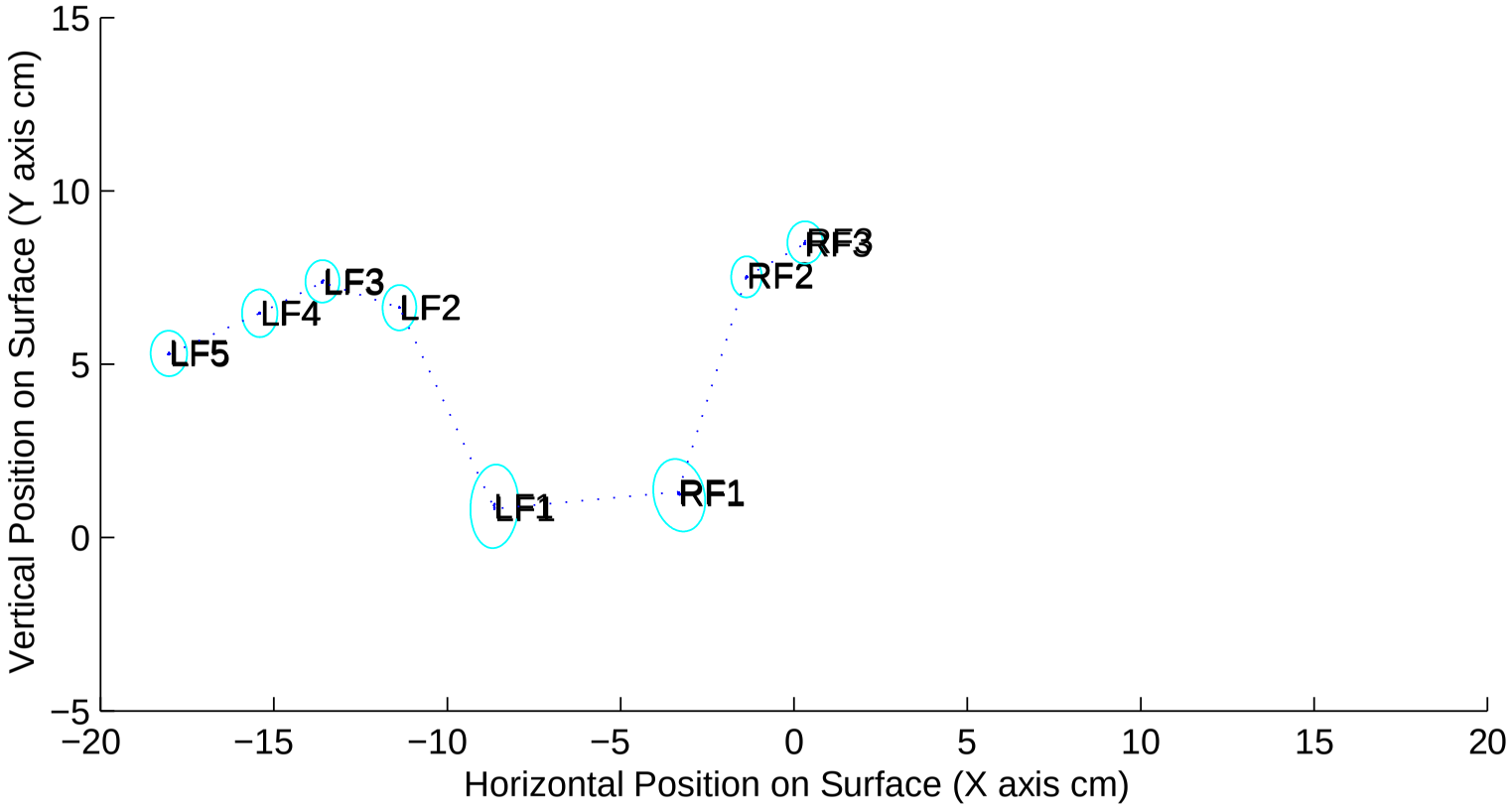


Figure 4.49: The five left hand fingers plus the right thumb, index, and middle fingers placed side by side on the left half of the surface are partitioned correctly. The only difference between this and Figure 4.49 is that the right index finger has been added and the thumbs are spaced apart a bit more, weakening the inter-hand separation factor. Nevertheless, the system handles this hand configuration correctly.

(Figures 4.43, 4.44).

Since thumb orientation, handedness, and clutching velocity features are fairly ambiguous even when the thumb is present, they are only used in the middle of the surface where hand position is often not distinguishing, not at the far sides of the surface, where false positive reversals could cause, for example, a left hand on the left side of the surface to be misidentified as a right hand. Therefore any sudden touchdown well to the left side of the surface which was not preceded by a hand sliding over from the right side will always be attributed to the left hand (Figure 4.45).

The contacts of two hands which touch down uncrossed yet close together will be partitioned correctly if palm heels are touching (Figures 4.46, 4.47). However, the inter-hand separation factor needed for clustering a hand which straddles the middle also tends to cause fingers from two adjacent, partially touching hands to be clustered into one hand, often erroneously filling the one hand's available palm attractors (Figure 4.48). Though the palm cohesion factor addresses this in some cases, additional methods of detecting nonsensical finger-palm arrangements will be needed to cover all combinations of finger/palm presence.

4.6 Conclusions

Though individual proximity images are often under-constrained, the identification system employs somewhat redundant stabilization mechanisms to achieve robust performance [161]. Ratcheting of identification accuracy upon the very dependable path tracking system stabilizes the discrete system state, *i.e.*, the finger and hand identities. Basing hand position estimates upon these identities rather than just a centroid of hand contacts also ensures that the continuous system state held in the estimates has a stabilizing effect on new identifications. When at least the four fingertips of a hand touch the surface, distance-squared assignment is constrained well enough that attractor ring alignment is irrelevant, making the hand

position estimates redundant in this case as well. Similarly, while the various attractor and hand partition weighting factors are designed to improve identification accuracy when contact arrangements or hand position is ambiguous, the weighting factors become redundant when the surface is fully populated with contacts from both hands.

The most surprising aspect of the architecture is that so many identification mechanisms coexist so peacefully, rarely contradicting one another. Manual tuning of system parameters is practical because each identification mechanism or weighting factor tends to dominate disambiguation of a specific hand configuration. Though balance with other mechanisms must be kept in mind, the selected mechanism's parameters can be tuned for its specific hand configuration without causing unmanageably complex side effects.

Though slight improvements could be made to thumb verification for the case that a thumb and fingertip are close together or to hand partitioning for the case when fingers from both hands are close together, the system distinguishes fingertips from palm heels and thumbs almost perfectly under operating conditions. Since the chordic manipulation system of the next chapter will be susceptible to human performance errors as well as recognition errors in segmentation, identification, and motion extraction, such identification flawlessness is crucial to keeping the overall error rate tolerable.

Chapter 5

CHORDIC MANIPULATION

This chapter will demonstrate how the tracking and identification capabilities developed in previous chapters are applied to integrate typing and chordic manipulation on the MTS. This integration of techniques for entry of text, commands, and graphics is founded upon a novel concept: *synchronous touchdown of multiple fingers should initiate pointing, command gestures, or hand resting, while asynchronous activity of individual fingers should be reserved for typing on a conventional key layout.* This concept and its implementation as described in this chapter enable the MTS operator to switch instantaneously between typing, pointing, and gesturing with a simple change in hand configuration, avoiding heavyweight mode switches such as reaching for another device, a mode-switch button, or a certain region of the surface.

Surprisingly, no one is known to have derived more than 2-DOF of control from hand or finger motion on a proximity-sensing surface such as a touchpad or touchscreen. This can probably be attributed to the small form factor of touchpads and to the scarcity of multi-touch sensing technologies for independently tracking motions of multiple fingers. This chapter will present techniques for weighting and filtering motions of particular fingers to integrally extract rotation and scaling degrees of freedom from unbalanced finger motions.

The chapter starts with a review of input devices which offer high-DOF manipulation or integration of typing and pointing. This review also explains the human-computer interaction principles and design criteria which influenced the development of chordic manipulation on the MTS. Next, the four modules of the

integration system are introduced: the finger subset synchronization detector, the typing detector, the hand motion extractor, and the chord motion recognizer.

The finger subset synchronization detector feeds the typing detector and chord motion recognizers with signals necessary to distinguish chordic manipulation from typing. The typing recognizer initially registers all finger touchdowns as keypresses but cancels those which are later found to be synchronized, sliding, or resting. The hand motion extractor filters four independent yet simultaneously accessible degrees of freedom from finger motion on each hand. The chord motion recognizer detects motion of particular finger chords in particular directions and generates the appropriate command or manipulation signals. Finger subset synchronization signals also gate selection of chordic manipulation channels within the chord motion recognizer. This gating improves ergonomics and mapping flexibility by allowing operators to drop all fingers to the surface after selecting a channel. Operators can also select a new channel from the hand resting posture by momentarily lifting a new finger subset instead of lifting the whole hand.

5.1 Related Input Devices

5.1.1 Fitts' Law and Pointing Performance

The basic targeting speeds of pointing devices are usually compared within the framework of Fitts' law [140]. Fitts' law states that the movement time MT for a targeting task is proportional to the index of difficulty ID for the task divided by the index of performance IP for a particular appendage operating a particular pointing device:

$$MT = ID/IP \tag{5.1}$$

Normally the movement time is measured in seconds, the index of difficulty in bits of position information, and the index of performance in bits per second. For the

task of moving the hand sideways toward a tall, columnar target area, Fitts [38] originally found that the index of difficulty could be expressed as:

$$ID = \log_2(\frac{2A}{W}) \tag{5.2}$$

where A is the horizontal distance from the starting hand or finger position to the target, and W is the horizontal width of the target. Other formulations such as Mackenzie's [93]:

$$ID = \log_2(\frac{A}{W} + 1) \tag{5.3}$$

have been proposed for the index of task difficulty.

Pointing device performance studies typically fit targeting time data to some version of Fitts' Law to obtain the index of performance for the given device [97]. A large body of research has extended Fitts' Law to variously shaped targets in two and three dimensions [2, 49, 76, 94–96], compared performance of devices which are controlled by flexing of different appendages such as fingers, wrist, and forearm [6, 97], and compared performance with different control-to-display gains [73], known more commonly as mouse cursor sensitivity and acceleration. Some of the more notable results have been that nonlinear mouse motion to cursor motion transfer functions which causes disproportionately large cursor motions at faster hand speeds do not decrease total targeting times but do decrease the distance the hand must move. Also, a higher index of performance is achieved with a stylus manipulated between the thumb and forefinger than by devices which sense only lateral motion generated at a single finger, the wrist, or forearm [6].

5.1.1.1 Tracking Delay

Temporal lags in tracking introduced by motion sensors and motion processing can also have a significant effect on manipulation performance. Mackenzie and Ware [96] found that lags as small as 75 ms increased time to target 10% and target

selection errors 36%, while lags of 225 ms increased movement time 64% and selection errors by 214%. Hoffman [64] had similar results, and both authors propose extensions to Fitts' Law to model these performance degradations. In light of this, the MTS chord motion recognizer is designed to keep motion initiation or channel selection lags less than 100 ms from the first finger touchdown of a chord. The actual lags depend on the rate of finger proximity stabilization, convergence of finger identification, and the lateral finger velocities, but generally they hover under 100 ms. However, once the channel is selected and the first motion control signals are generated, lag drops to at most two image frames, or 40 ms.

5.1.2 Integrating Typing and Pointing

A fundamental, difficult to circumvent, dichotomy exists between manipulation tasks, *e.g.* pointing, dragging, or scrolling, and discrete specification tasks such as command and text entry. Manipulation involves continuous, bidirectional adjustment of parameters in some coordinate space, such as the position, size, and hue of an on-screen object. In discrete specification, speech or a preordained hand gesture must specify a character, word, or command from a finite set. Though graphical user interface software has evolved so that users can freely intermix these approaches, the input devices which support each approach, typically mouse and keyboard, have remained substantially separate. This review will outline the capabilities of existing direct manipulation devices and analyze past attempts to support manipulation and discrete specification in an integrated device.

5.1.2.1 Embedding Pointing Devices in Mechanical Keyboards

Though people have long lamented the need to reach back and forth between the mouse and keyboard, the most visible attempts at integrating typing and pointing have been driven by the miniaturization demands of laptop computers. Most manufacturers have replaced the tiny thumb-operated trackballs of the early

1990s with credit-card-sized touchpads located below the spacebar. While this is certainly an improvement, some users have a tendency to accidentally tap the pad with their thumbs while typing, causing random mouse clicks. Synaptics, Inc., the primary touchpad OEM, claims to have developed special filters which address this problem [143].

IBM and Toshiba laptops continue to feature the Trackpoint pointing stick, a tiny force-sensitive joystick embedded between the 'g', 'h', and 'b' keys. Mouse pointer velocity is proportional to the directional force applied to the stick by the fingertips, and physical buttons below the spacebar serve as mouse buttons. Rutledge and Selker [131] were the first to study the pointing stick, and they expected to find that tasks involving a mixture of pointing and typing would be faster with the embedded stick since reaching off the key layout for the pointing device was unnecessary. They actually found that though the homing time to switch from pointing to the keyboard was reduced to 90 ms, switching from typing to pointing still took about 400ms, 2/3 as long as for the mouse, because the pointing stick is such a small target for the finger to find. The mouse also remained 25% faster than the pointing stick for pure pointing tasks.

Similar joysticks have been built into a key of the keyboard such as the 'j' key [40]. Pointing mode is entered after holding the key down for a short time interval such as 200 ms. Clicking is accomplished by pressing another key while still in pointing mode, *e.g.* while the 'j' key is pressed. Douglas and Mithal [32] found that though the homing time for the key joystick was again about 2/3 that of the mouse (438 ms compared to 667 ms), the mouse was still about twice as fast at pointing and dragging and therefore performed better overall even in mixed pointing and typing tasks. Several subjects also complained of fatigue from having to hold the 'j' key down to remain in pointing mode.

Though the MTS integration method will also be susceptible to accidental

activations, pointing mode can be entered at any time, anywhere on the surface (including directly over home row) by placing two adjacent fingertips on the surface synchronously. Similarly, dragging can be initiated by synchronously placing and sliding three adjacent fingertips. Typing can resume any time after these fingertips lift off the surface. Therefore the homing time for the MTS going both from typing to pointing and from pointing to typing is simply the time needed to lift the fingers off the surface and put them back down again. This time ranges between 150 ms and 300 ms depending on how hurried the operator is. Because the MTS offers such a wide range of hand movement, its pointing speed and accuracy should be much closer to that of a mouse than these tiny joysticks and touchpads. Therefore one would expect better overall performance in mixed typing and pointing tasks on the MTS as long as its typing performance is not degraded.

5.1.2.2 Detecting Pointing Gestures Above a Keyboard

An interesting but less commercially successful approach has been to sense finger pointing either remotely or with ring attachments while the hand floats above the keyboard. For example, Levine [91] attached a stylus ring to the thumb and a palette ring to the index finger. Users could wear the rings while typing and rub the stylus ring against the palette ring to move the pointer and click. Sibert and Gokturk [136] similarly placed an infrared-emitting ring on the index finger. Four infrared sensors mounted at the corners of a laptop display inferred the direction the ring is pointing. Thus users would point their finger directly at the desired object on the display.

Quek [121] applied his sophisticated video-based hand gesture recognition system to track pointing gestures over a keyboard. The camera was mounted to look down at the keyboard, and pointing mode began when the user extended the index fingers and curled all the others. Pointing performance was only 18% poorer than with a mouse. Without a blue dot to demarcate the fingertip, Quek's video-based

approach could only track the finger at 7 fps on a 150-MHz Silicon Graphics Indigo 2 workstation. Clearly the ring attachments, camera, or intensive computations required with these approaches are undesirable if not impractical.

5.1.2.3 One Hand Points, the Other Types

Typing or issuing key commands with one hand while manipulating a mouse or drawing tablet puck with the other is a natural approach often adopted by operators of computer-aided-design (CAD) software [82, 166]. A few companies [56, 69, 103–106, 119, 153] have devised one-handed keyboards for this purpose which use unique chording schemes or layouts to ensure all letters are within easy reach of one hand. The primary disadvantage of these devices is that the operator must learn a new key layout or chord typing scheme.

A strict allocation of manipulation tasks to one hand and discrete specification to the other has more organizational clarity than the MTS mappings, which spread both manipulations and typing equally over both hands. However, even if the operator's task requires equal amounts of typing and pointing, the risk of overuse injuries in the hands is presumably greater for the one-handed keyboards because the typing load for one hand is doubled, and the mousing hand will use only the mousing muscles over and over. Assuming that typing and pointing do not load exactly the same muscles, spreading these activities evenly over both hands as the MTS does decreases the likelihood that any subset of muscles or tendons in one hand will be overused. Finally, the one-handed typing, one-handed pointing approach precludes bimanual manipulations such as panning the background with the non-dominant hand while pointing with the dominant hand.

5.1.2.4 Touch Pads and Screens

Touch screens and touchpads often distinguish pointing motions from emulated button clicks or keypresses by assuming very little lateral fingertip motion

will occur during taps on the touch surface which are intended as clicks. Inherent in these methods is the assumption that tapping will usually be straight down from the suspended finger position, minimizing those components of finger motion tangential to the surface. This is a valid assumption if the surface is not finely divided into distinct key areas or if the user does a slow, "hunt and peck" visual search for each key before striking. For example, in a patent to Logan [92], taps with less than about 1/16" lateral motion activate keys on a small keypad while lateral motion in excess of 1/16" activates cursor control mode. In both patents cursor mode is invoked by default when a finger stays on the surface a long time.

However, fast touch typing on a surface divided into a large array of key regions tends to produce more tangential motions along the surface than thresholding of lateral finger motion can tolerate. Such an array contains keys in multiple rows and columns which may not be directly under the fingers, so the user must often reach with the hand or flex or extend fingers to touch key regions. Quick reaching and extending imparts significant lateral finger motion while the finger is in the air which may still be present when the finger contacts the surface. Glancing taps with as much as 1/4" lateral motion measured at the surface can easily result. Attempting to filter or suppress this much motion would make the cursor seem sluggish and unresponsive. The MTS gets around this problem by only mapping pointing and other manipulations to chords of two or more fingers, basically ignoring lateral motion by lone fingers.

5.1.3 Manipulation in more than Two Degrees of Freedom

Each hand offers a total of 29 degrees of freedom (DOF) of motion [141], 23 of which come from the finger joints above the wrist. The remaining 6 come from the overall hand position and orientation as measured from the palm center. A wide range of devices have been developed to capture and reduce this wide range of hand and finger flexibility into 3-6-DOF of motion control for two and three-dimensional

graphical manipulation tasks. Such devices include 6-DOF force-sensitive balls such as the Spaceball and Elastic General Purpose Grip Controller [169], the 6-DOF Polhemus [19] and Bat [153] free-space hand trackers, the 4-DOF Rockin' mouse [5], 4-DOF tilt-sensitive styli [149], and the 3-DOF two-ball mouse [98].

Human-computer interaction researchers have devised various classification and evaluation schemes for these devices. For example, Mackinlay *et al.* argue that:

> ... input devices are transducers of any combination of linear and rotary, absolute and relative, position and force, in any of the six spatial degrees of freedom. [100], Page 145

The performance evaluation research, in turn, can be summarized by the assertion of Jacob *et al.* that:

> performance improves when the perceptual structure of the task matches the control structure of the device. [71], Page 6

Thus position-sensing devices perform best for position control tasks [167], force-sensitive devices perform best for rate or velocity control tasks [167], rotation-sensing devices perform best for rotation control tasks [63], and so on. This suggests that the MTS will be particularly adept at rotating and scaling two-dimensional documents and objects. While such 2D document manipulation may not be as glamorous as full 6-DOF navigation in 3D virtual worlds, almost all existing software applications could benefit from more accessible rotation and scaling capabilities.

5.1.3.1 Integrality vs. Separability

The correspondence between device control structure and task perceptual structure is particularly important with regard to bimanual manipulation and integrality of degrees of freedom. Integral degrees of freedom are those that can be controlled simultaneously with a device to transmit diagonal motion. Separable degrees of freedom can only be accessed one at a time, limiting movement to directions along an axis like the orthogonal drawing mode of many CAD programs. Jacob *et*

al. [71] found that integral tasks such as simultaneously sizing and positioning an object in two dimensions are best controlled with a device providing three integral degrees of freedom. This is not surprising since one can usually reach a target more quickly if allowed to travel along the diagonal if the target is in a diagonal direction.

However, Jacob *et al.* also found that for cognitively separable tasks such as changing object color and positioning the object in two dimensions, a separable device such as a 2DOF mouse with a button to switch between color adjustment and position adjustment performed better. Thus the integrality of separability of the device should be matched with that of the task. Other researchers have also found in some cases that having too many integral degrees of freedom available simultaneously can hurt performance because as users zero in on the target in one set of axes, instabilities in hand motion may nudge the cursor away from the target along the other axes.

For example, auxiliary scrolling controls for mice such as the pointing stick on the IBM ScrollPoint mouse and the middle finger roller on the Roller Mouse of Gillick and Lam [52] provide more than the two degrees of freedom standard for mice. However, these scrolling degrees of freedom should be considered separable from the two pointing degrees of freedom because, as Zhai *et al.* [170] note, manipulation of more than two degrees of freedom at a time is very difficult with these devices, preventing simultaneous panning, zooming and rotating.

Like the recent dual-pointing-stick bulldozer-interface of Zhai *et al.* [168], the hand motion extractor presented in this chapter will try to strike a compromise between integrity and separability. The motion filters will pass simultaneous motions in multiple components or degrees of freedom which are truly in diagonal directions, thus allowing fast manipulation across shortest path diagonals. But when one of the rotation, scaling, or translation degrees of freedom dominates, indicating the direction of motion is nearly along an axis, the non-dominant components will

be suppressed so that control occurs exactly along the axis. Thus operators should receive the speed gains of diagonal motion for coarse movements but be automatically switched to a separable mode on the final approach to a target, protected from the instabilities and non-uniformities of their own hand motions.

5.1.3.2 Bimanual Manipulation

Bimanual manipulation refers to simultaneous manipulation using both hands. Guiard's kinematic chain model [55], which posits that the coarse motions of the non-preferred hand have evolved to function as a dynamic frame of reference for the fine motions of the preferred (right) hand, has stimulated considerable experimentation in the human-computer interaction community. Noticing that people normally use their left hand to keep the position and orientation of a piece of paper optimal for handwriting with the right hand, Guiard and Athenes [55] found that handwriting speed drops up to 20% if subjects are instructed not to manipulate the page with the non-preferred hand.

In one of the earliest studies, Buxton and Myers found a 15-25% performance increase in a mixed scaling and positioning task when one hand positioned with a puck while the other scaled with a slider, and subjects actually adapted the parallel hand usage strategy without prompting. Kabbash *et al.* [77] verified that the left hand is faster at long-distance pointing with mouse, trackball, or stylus, and the right hand is faster at precision movements toward small targets. Leganchuk *et al.* [90] found 15-30% better performance sweeping out rectangles with two-hands, *i.e.*, each hand controlling an opposite corner. Hinckley *et al.* [63] claim that users can internally sense the position of the preferred hand with respect to the other and manipulate accordingly without visual feedback.

Most of these experiments have been carried out with combinations of conventional devices for each hand such as a stylus and puck on a Wacom tablet [86, 90], a mouse and a touchpad [61], a mouse and a pointing stick [170, 171], or two pointing

sticks [168]. While such bimanual manipulation techniques have not been explored yet on the MTS, its wide surface and hand identification ability are perfectly suited for them. Moreover, the MTS is the only device known to be able to support typing and bimanual manipulation in the same space. This could make bimanual manipulation practical for a much wider population of computer users.

5.1.4 Channel Selection

Channel selection corresponds to pressing buttons on a mouse to activate alternative modes such as dragging or scrolling. Most mice have one to three buttons which, when used in chorded combination, can select up to 7 channels. Specialized pucks for CAD systems may offer many more buutons. The finger identification system of the MTS can distinguish seven channels, but the seventh is reserved for whole hand resting. The primary advantage of MTS channel selection over mouse channel selection is that the MTS does not require any sustained button-pressing forces.

The multiple channel capability of the MTS also allows it to avoid the cumbersome tap-drag sequence of single finger touchpads. Unlike drawing tablets, which can sense the difference between a stylus hovering over the tablet and one pressing on the tablet, touchpads can only track a finger when it is touching the surface [61]. Thus they require a special tap-drag timing sequence to distinguish finger motions meant for pointing from those meant for dragging. The MTS is free to allocate dragging to one of its extra finger chord channels, avoiding the awkward tap-drag sequences.

Fitzmaurice and Buxton [39] essentially argue that the device structure/perceptual structure correspondence should extend to channel selection as well. They advocate distinct graspable tools, each with a physical transducer, which are shaped as rulers,

stretchable squares, bricks, or rotor to match the task of manipulating a set of corresponding virtual objects which look the same onscreen. They find that subjects are faster at tracking four randomly moving objects onscreen with these specialized, space-multiplexed devices than with generic space-multiplexed devices such as labeled pucks. Time-multiplexing by using one device to manipulate the cursor, select, and drag each virtual object is also significantly slower. Again, the Wacom tablet technology, which can track and identify multiple, cordless devices such as pucks, styli, and airbrushes on the same tablet at the same time, was used for this experiment.

This capacity to physically pick from multiple, specialized tools is commendable and well-suited for certain applications such as virtual painting and character keyframe animation [39]. In contrast, the MTS's chordic channel selection is closest to generic space multiplexing and therefore may require additional cognitive load to memorize the function of each channel. However, switching between channels on the MTS is comparably instantaneous since there is no need to put down one tool and pick up another. Thus if MTS operators are faced with the same tasks day in and day out, they will likely achieve better tool or mode-switching performance, and again they will not have to worry about tools impeding typing by cluttering the work surface when put down.

5.2 Synchronization and Typing Detection

This section will explain how the MTS uses the timings of finger touchdowns and liftoffs to distinguish between asynchronous key taps and synchronous chord taps or hand resting.

5.2.1 Keypress Registration

Figure 5.1 contains a flowchart of the keypress registration loop. As each new

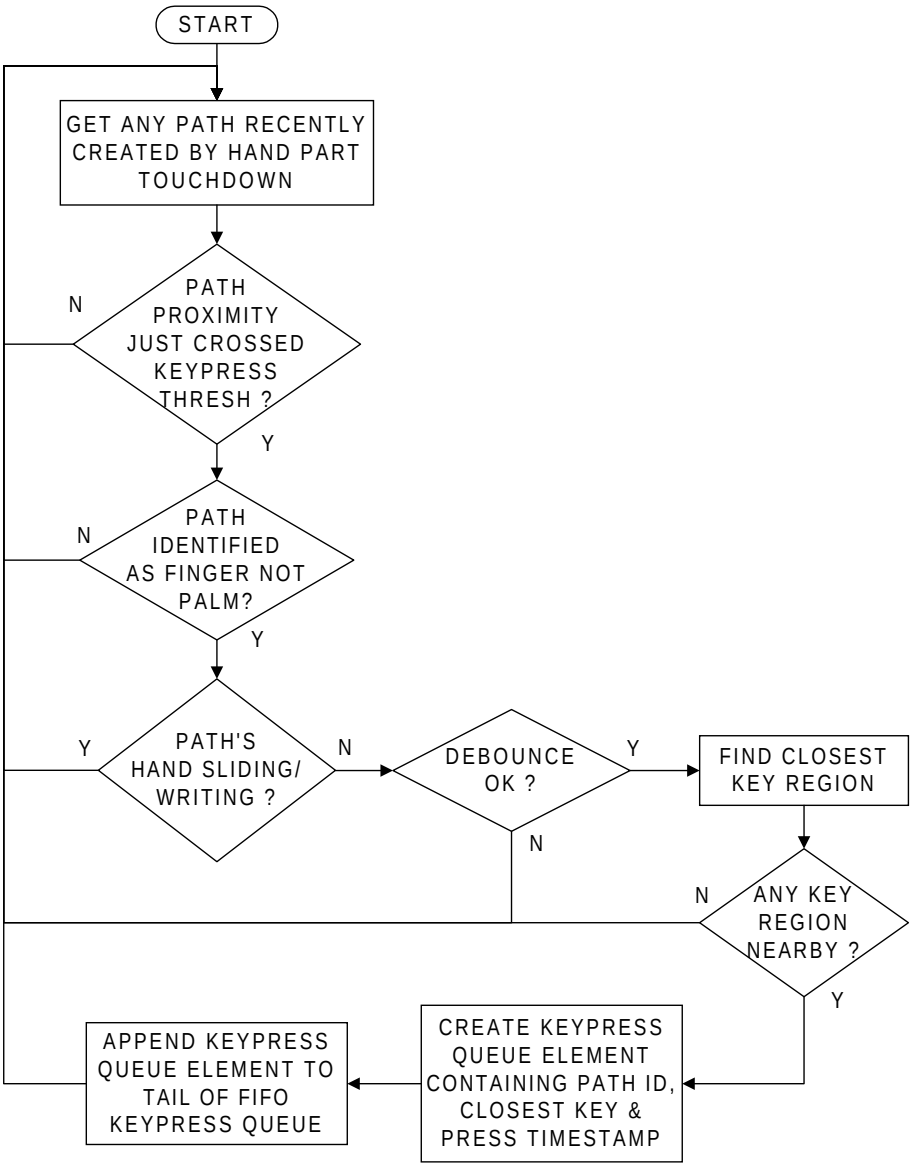


Figure 5.1: Flow chart of the keypress registration process.

hand part touches down and its contact proximity surpasses a small threshold, it is registered in a keypress queue unless any of the following conditions are true:

- its contact path has been identified as a palm instead of a finger.
- the hand it is associated with is currently involved in a chordic manipulation.
- it fails debounce testing, *i.e.*, the same hand part lifted off the surface less than 100–150 ms prior to the current touchdown.

In these cases the touchdown must be ignored by the typing detector to avoid generation of keypresses from hand motions clearly not intended as typing. The path tracking module (Section 3.3.3) facilitates debounce testing by reactivating a finger's old path if the finger lifts off and quickly touches back down over the same spot. Upon reactivation the timestamp of the last liftoff by the old path is preserved for comparison with the timestamp of the new touchdown.

Assuming the finger touchdown passes these registration tests, the current position of the fingertip centroid ($Fi_x[n]$, $Fi_y[n]$) is used to find the nearest key in a predefined QWERTY key layout (*e.g.* see Figure 1.1 on Page 6). The touchdown may be ignored if there are no key regions within a centimeter of the fingertip. Assuming a key region is close to the finger, a keypress element data structure is created containing the path index and finger identity, the closest key region, and a timestamp indicating when the finger crossed the keypress proximity threshold. The final step then appends this keypress element data structure to the tail of the FIFO (first-in first-out) keypress queue. This accomplished, the loop continues to process or wait for touchdowns by other fingers.

Note that the keypress queue effectively orders finger touchdowns by when they pass the keypress proximity threshold. It thus fixes the order in which key symbols from each finger tap will be transmitted to the host. However, an element's key symbol is not assured transmission to the host once in the keypress queue. Any

of a number of conditions to be discussed in the following sections such as being part of a synchronized subset of fingers can cause an element to be deleted from the queue before being transmitted to the host. In this sense the keypress queue should be considered a keypress *candidate* queue. Unlike the ordered lists of finger touchdowns and releases maintained for each hand separately in the synchronization detector below, the keypress queue includes and orders the finger touchdowns from both hands.

5.2.2 The Synchronization Detector

Figure 5.2 shows a flowchart of the finger synchronization detection algorithm. The flowchart continues into Figure 5.3 to show chord tap detection. This synchronization detection process is repeated independently for the contacts assigned to each hand. Within each hand, the process takes as input the current finger identifications and life cycle markers (P_{press_t} and $P_{release_t}$ in Section 3.3.4) of each contact path. The identities are needed to ignore palm paths and distinguish different chords or combinations of synchronized fingers, while the life cycle markers record the time at which each contact path first exceeds a press proximity threshold and the time at which each contact path drops below a release proximity threshold prior to total liftoff. These proximity thresholds are currently set to about one-fifth the average fingertip proximity. Higher thresholds should be tolerable with faster image frame rates (see Section 6.1.5.1), but with lower proximity thresholds, the measured press and release times become imprecise, making comparisons and sorting of them unreliable.

5.2.2.1 Sorting Paths by Press and Release Times

After the path identities and life cycle markers for the current proximity image have been retrieved, the synchronization detection algorithm first searches for subsets of fingers which touch down at about the same time and for subsets of

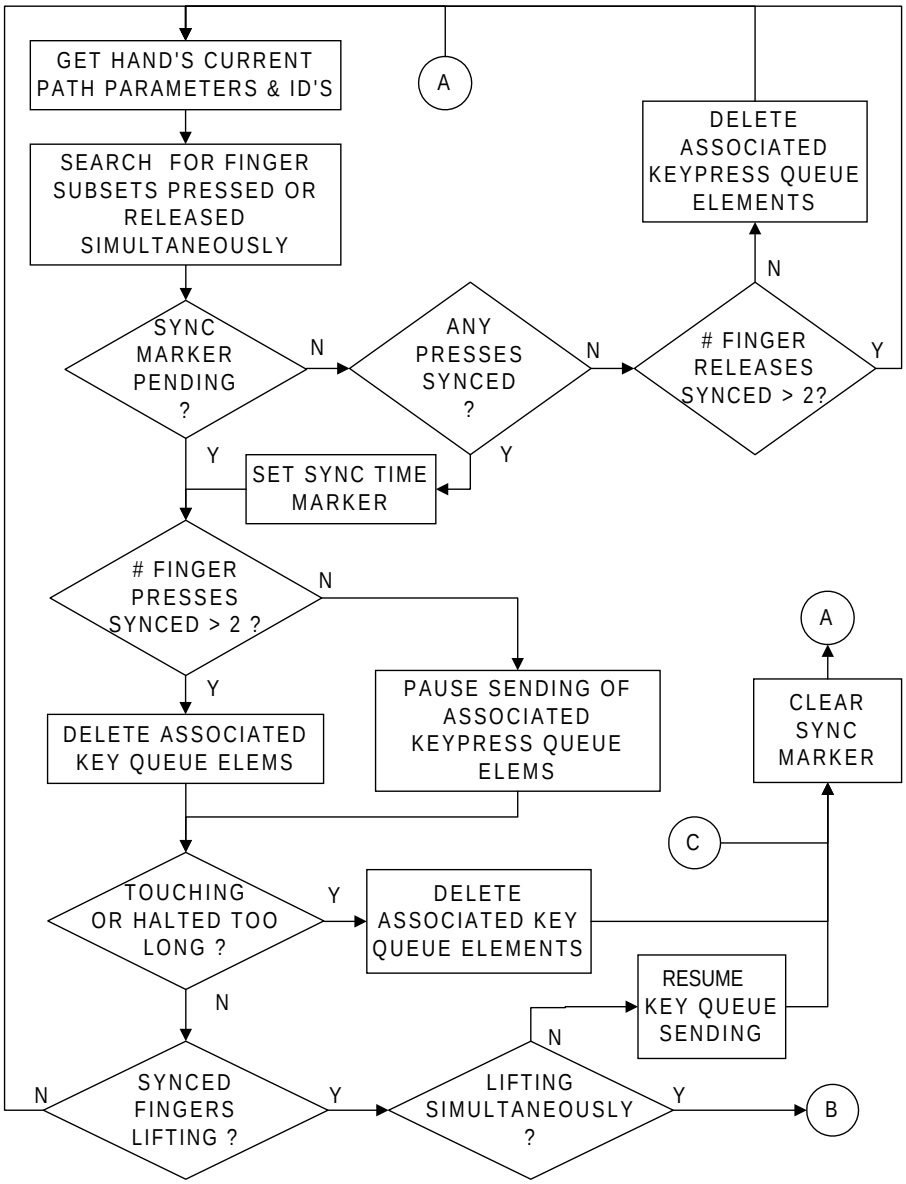


Figure 5.2: Flow chart of the finger synchronization detection process.

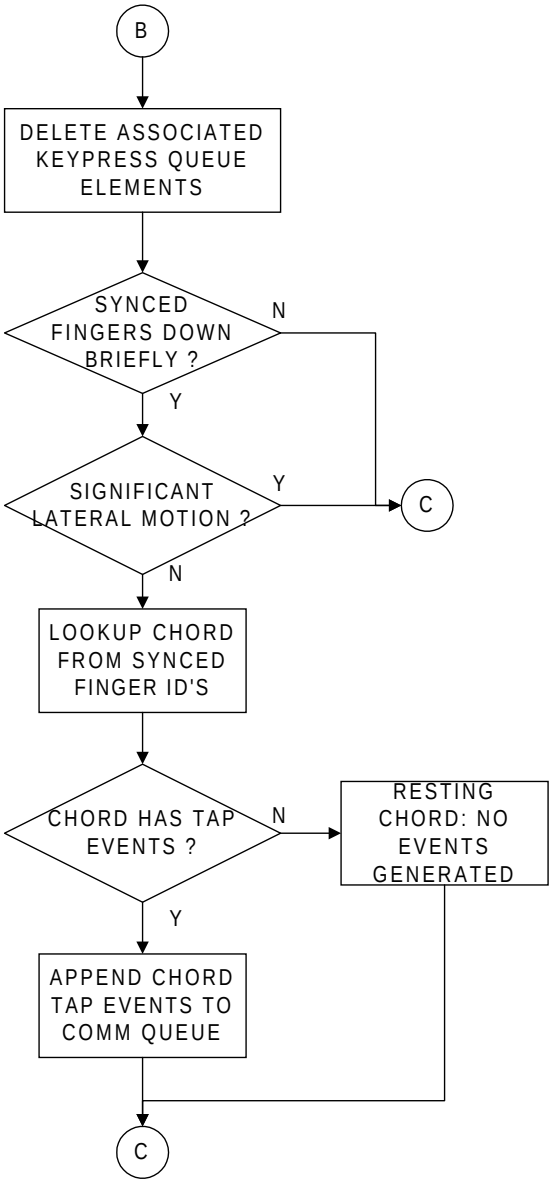


Figure 5.3: Continuation of Figure 5.2 showing chord tap detection and transmission.

fingers which lift off at about the same time. This can be done by recording each finger path along with its press time in a list as the finger crosses the press proximity threshold. The list will therefore be ordered according to path press times. A similar but separate list is maintained for path release times. Since the primary function of the palms is to support the forearms while the hands are resting, not to actively participate in typing or chordic manipulation, palm heel presses and releases are excluded from these lists and most other synchronization tests.

5.2.2.2 Searching for Synchronized Finger Subsets

To check for synchronization between the two most recent finger presses, the press times of the two most recent entries in the list are compared. If the difference between the two press times is less than a temporal threshold, the two finger presses are considered synchronized. If not, the most recent finger press is considered asynchronous. Synchronization among three or more of the most recent fingers up to five is found by comparing press times of the three, four, or five most recent list entries. If the press time of the most recent entry is within a temporal threshold of the nth most recent entry, synchronization among the n most recent finger presses is indicated.

To accomodate imprecision in touchdown across the hand, the magnitude of the temporal threshold increases slightly in proportion to the number of fingers being tested for synchronization. To provide some hysteresis between typing and chordic manipulation modes, the threshold also depends on the time since the last typing-related touchdown or liftoff on either hand. The temporal threshold for press synchronization detection can therefore vary between 0 ms and about 150 ms. The largest set of recent finger presses found to be synchronized is recorded as the synchronized subset, and the combination of finger identities comprising this subset is stored conveniently as a finger identity bitfield. The term subset is used because the synchronized press subset may not include all fingers currently touching the

surface, as happens when a finger touches down much earlier than other fingers yet remains touching as they simultaneously touch down. The list of path identities sorted by release times is searched similarly to check for synchronous release of any finger subset.

5.2.2.3 Synchronization Detector Decisions and Actions

The actions taken by the synchronization detector in Figure 5.2 can be summarized as follows:

- Synchronized liftoff of three or more fingers always causes the keypresses associated with those fingers to be canceled, regardless of whether the original touchdowns of those fingers were synchronized.

- Synchronized touchdown of three or more fingers always causes the keypresses associated with those fingers to be canceled immediately, before anything is known about liftoff synchronization.

- Synchronized touchdown of two fingers is ambiguous in itself, so a hold is placed on the keypress processing queue which prevents either of the associated keys from being transmitted until the fingers releases can be checked for synchronization. In case of asynchronous liftoff, the finger motions are most likely keypresses, so the hold on the keypress queue is released, allowing transmission of the keypresses to the host computer. In case liftoff of the two fingers is synchronized or both fingers remain on the surface more than about half a second, the associated keypresses are canceled, indicating that the fingers are either just resting or part of a chord tap.

- Synchronized touchdown followed by synchronous liftoff of a subset of two or more fingers without significant intervening lateral motion is considered a chord tap.

Note that finger pair synchronization detection must be treated as a special case because sometimes when striking adjacent keys, the fingers roll from one key to another so quickly that either touchdown or liftoff appears synchronized, but there will still be some asynchrony in either touchdown or liftoff. Such rolling does not, however, cause synchronization of touchdown or liftoff across more than two fingers, so either touchdown or liftoff synchronization of three or more fingers is a sure sign those fingers are not involved in typing. Instead of thresholding the touchdown time difference and liftoff time difference separately, the finger pair liftoff and touchdown time differences are added together and thresholded once for more robust detection. However, even with such averaging of touchdown and liftoff synchronization, in a few borderline cases the 50 fps sensor array scan rate is simply too slow to differentiate barely asynchronous adjacent key strikes from synchronized finger pair chord taps. This will be discussed further in Section 6.1.5.1.

5.2.2.4 Issuing Chord Taps

If the chord tap conditions are met, the bitfield of finger identities for the synchronized subset is used to check a lookup table for any input events such as mouse clicks or keyboard commands assigned to the combination of fingers in the chord tap. Though there are 26 possible combinations of identities for two or more fingers, combinations containing the same number of fingertips all refer to the same chord channel, and there are only seven unique channels per hand. The unique channels are illustrated in Table 5.1 below. The channel for all five fingers is reserved for hand resting, so chord taps of the whole hand will produce no input events. The chord tap event lists of many of the other channels (especially the three and four fingertip channels) may also be left empty to encourage hand resting during typing by novices. Mouse clicks are the only events which absolutely need to be generated by chord taps, so one channel must be allocated for each mouse button to be emulated for a given operating system. See Tables 6.1–6.4 on Pages 289–292

Table 5.1: The seven unique finger chord channels.

Channel Icon	*Finger Combination*
	Any 2 fingertips (excluding thumb).
	Any 3 fingertips (excluding thumb).
	All 4 fingertips (excluding thumb).
	Thumb and any fingertip.
	Thumb and any 2 fingertips.
	Thumb and any 3 fingertips.
	Thumb and all 4 fingertips.

for examples of other chord tap event mappings used by the author. Though event generation from chord taps needs to be restricted to encourage hand resting, a wide range of input events can safely be generated on most channels in response to lateral hand motions, as will be discussed in Section 5.4.3.

5.2.2.5 Avoiding Accidental Mouse Clicks

As a further precaution against accidental generation of mouse clicks during typing, the chord tap event generator ignores the first chord tap which quickly follows a valid keypress without an intervening lateral chord slide. This avoids spurious mouse clicks which can randomly reposition the text cursor, yet it rarely causes intentional chord taps to be lost since usually after typing the user will need to reposition the mouse cursor before clicking, requiring an intervening chord slide. If the mouse cursor happens to already be in place after typing, the user may have to tap the finger chord a second time for the click to be sent, but this is much less aggravating than undoing unintentional mouse clicks in the middle of a typing session.

5.2.3 Keypress Acceptance and Transmission

Figure 5.4 shows the steps within the keypress acceptance and transmission loop. This loop performs final keypress timing and identity tests upon finger release before sending the key's symbol or associated events to the host computer. The first step is to peek at the element at the head of the keypress queue. This head queue element represents the oldest finger touchdown which has neither transmitted its associated key symbol nor been deleted from the queue as an invalid keypress candidate. This head queue element can be deleted at any time prior to liftoff of its associated contact path if any of the following conditions become true:

- the path's identity is changed by the identification system from any finger identity to a palm heel or forepalm identity.

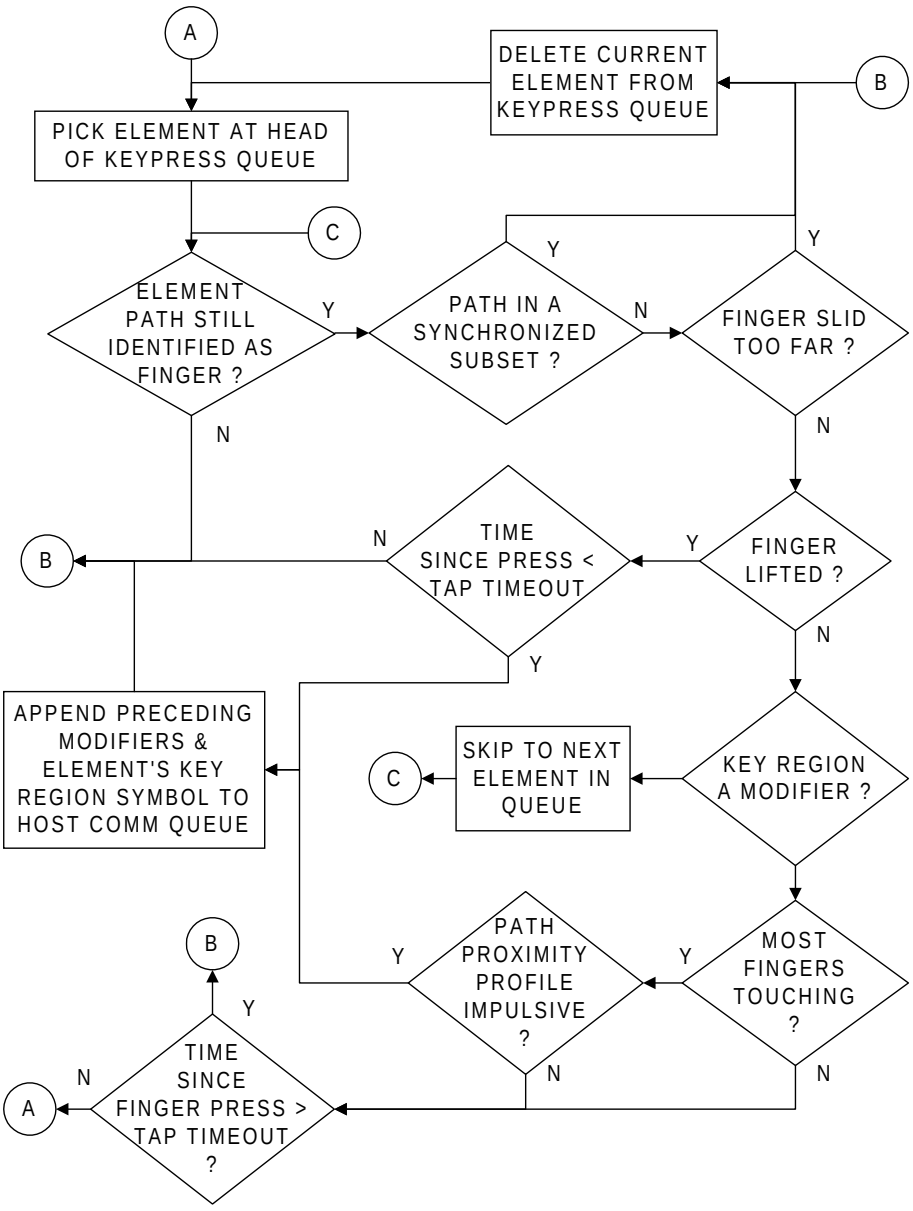


Figure 5.4: Flow chart of the keypress acceptance and transmission process.

- the path is found to be a member of a synchronized finger subset as described in the previous section.

- the contact has been on the surface more than about half a second without liftoff and is clearly not a modifier key or typematic finger hold (see below).

Because users may be touch typing on the surface, several millimeters of lateral motion are allowed to accommodate glancing fingertip motions which often occur when quickly reaching for keys. This is much more glancing tap motion than is tolerated by touchpads which employ a single finger slide for mouse cursor manipulation and a single finger tap for key or mouse button click emulation.

Assuming the keypress element has not been deleted by the above tests, the algorithm next checks whether the finger whose touchdown created the keypress element has since lifted off the surface. If the finger has lifted off soon enough after touchdown to qualify as a normal key tap, the associated key symbol is transmitted to the host and the keypress element is deleted from the head of the queue. The MTS also generates a clicking sound for feedback to the operator as the key symbol is transmitted to the host. Note that a keypress is always deleted from the queue upon liftoff, but even though it may have stayed on the surface for a time exceeding the tap timeout, it may have still caused transmission as a modifier key, as an impulsive press with hand resting, or as a typematic press, as described below.

5.2.3.1 Handling Modifier Keys

To handle modifier keys such as <shift>, <ctrl>, or <alt>, if the head element's finger has not yet lifted but the finger is over a modifier key, processing advances to the next element in the queue without deleting the head. If the next element is a valid key tap and successfully reaches the transmission stage, the transmission stage will scan back toward the head of the queue for any modifier regions

which are still pressed. The next element's key symbol can then be sent to the host along with the modifier flags of any preceding modifier regions.

5.2.3.2 Alternatives to Full Taps from Suspended Hands

Normally operators must touch the finger on the surface and lift back off within a few hundred milliseconds for a key to be sent. Like the activation force threshold of mechanical keyswitches, this timing constraint provides a way for the operator to rest the finger on the key surface asynchronously without invoking a keypress. This is necessary because operators sometimes begin hand resting by simultaneously placing the central fingertips on the surface, but they follow asynchronously with the pinky a second later and the thumb a second after that. These latter presses are essentially asynchronous and will not be invalidated by the synchronization detector, but as long as they are not lifted within a couple hundred milliseconds, they will essentially time out and be deleted without transmission. However, the requirement that fingers quickly lift off, *i.e.*, crisply tap, the surface to cause key generation makes it very difficult to type long sequences with most fingers resting on the surface to support the hands. Basically, words cannot be typed quickly without floating the hands above the surface. This is acceptable typing posture except that the operators' arms will eventually tire if the operator fails to rest the hands back on the surface between sequences.

To provide an alternative typing posture which does not encourage suspension of the fingers above the surface, the MTS has a second key acceptance mode which does not require quick finger liftoff after each press. Instead, the user must start with all five fingers of a hand resting on the surface. Then each time a finger is asynchronously raised off the surface and dropped onto a key region, that key's symbol will be transmitted, regardless of subsequent liftoff timing. To allow the operator to gently set down a raised finger without generating a key, the impulsivity

of the proximity profile is measured according to the time taken for fingertip proximity to saturate. If the proximity profile increases to its peak very slowly, say over a 100 ms time interval, no key is generated from the finger touchdown. Such typing from a resting hand posture requires minimal effort to support the hands or strike keys, but this technique limits typing speed to about 20 words per minute (wpm); thus it is intended mainly for people with repetitive strain injuries so severe that the slightest exertion hurts.

5.2.3.3 Potential Typing Speeds

Though additional enhancements such as tactile feedback of key locations and typing sequence recognition algorithms will be necessary to make touch typing on the MTS as accurate as typing on a mechanical keyboard, a few remarks can be made about how interaction of tapping motions with a hard surface ultimately limits typing speeds. The best typing speeds seem to be obtainable with an intermediate hand posture in which the MTS is placed on the lap to slope downward by 5-10°, the palms rest on the surface, and the fingertips float very close to the surface. The downward slope reduces the exertion by the wrist extensors needed to keep the fingertips floating when the palms are planted on the surface. The fixed position of the palms serves as a reference for more carefully regulating the height that fingertips float above the surface without letting them accidentally touch. With the palms planted, average floating finger height can be as little as 1/4". This reduces the downward travel necessary to strike a key region and may eventually support typing speeds up to about 80 wpm. The closeness of the fingers to the surface also makes it all but impossible to strike the surface so hard that the fingertips get jarred.

If the operator keeps palms floating above the surface, the regulation of floating fingertip heights is not so stable, and the fingertips must be kept floating about 1/2" above the surface to avoid accidental touches. This appears to limit the maximum typing rate attainable to about 60 wpm when hands are fully suspended

above the surface. The increased and unstable variations in floating fingertip height also cause more variation in the impulsiveness of finger impact. This may result in occasional fingertip jarring.

5.2.4 Typing Summary

The typing detection process described above thus allows the multi-touch surface to ergonomically emulate both the typing and hand resting capabilities of a standard mechanical keyboard. Crisp taps or impulsive presses on the surface generate key symbols as soon as the finger is released or the impulse has peaked, ensuring prompt feedback to the user. Fingers intended to rest on the surface generate no keys as long as they are members of a synchronized finger press or release subset or are placed on the surface gently and remain there along with other fingers for a second or two. Once resting, fingers can be lifted and tapped or impulsively pressed on the surface to generate key symbols without having to lift other resting fingers. Glancing motions of single fingers as they tap key regions are easily tolerated since chordic manipulations can only be initiated by synchronized slides of two or more fingers.

5.3 Hand Motion Extraction

Technically, each hand has 23 degrees of freedom of movement in all finger joints combined, but as a practical matter, tendon linkage limitations make it difficult to move all of the joints independently. Measurements of finger contacts on a surface yield ten degrees of freedom in motion lateral to the surface, five degrees of freedom in individual fingertip pressure or proximity to the surface, and one degree of freedom of thumb orientation. However, many of these degrees of freedom have limited ranges and would require unreasonable twisting and dexterity from the average user to be accessed independently.

The purpose of the motion component extraction algorithm is to glean from the 16 observable degrees of freedom enough degrees of freedom for manipulation of two-dimensional graphics. In two dimensions, the four basic degrees of freedom are horizontal translation, vertical translation, rotation within the surface plane, and zooming or resizing within the surface plane. For full 6-DOF manipulation in three dimensions, two more rotational degrees of freedom are needed about the horizontal and vertical axes. Though these could plausibly be obtained from differences in hand tilt pressure across the surface, only 4 degrees of freedom in velocity will be extracted here.

When only four degrees of freedom are needed, the basic hand and finger motions can be whole hand translation, hand scaling by uniformly flexing or extending the fingers, and hand rotation either about the wrist as when unscrewing a jar lid or between the fingers as when unscrewing a nut. Not only are these hand motions easy to perform because they utilize motions which intuitively include the opposable thumb, they correspond cognitively to the graphical manipulation tasks of object rotation and sizing. Their only drawback is that the translational motions of all the fingers during hand rotations and scalings do not cancel perfectly. As will be seen in Figure 5.6, usually they add up to a net translation in some direction in addition to the desired rotation or scaling. This makes it difficult for translation to be integral with or performed simultaneously with scalings and rotations. To prevent non-uniformities in rotation and scaling motions from bleeding into the extracted translations, the translation extractor will preferentially weight fingers such as thumb and pinky whose translations cancel best. To provide uniform motion gain even when some fingers remain stationary, it will also nonlinearly scale velocity components depending on the finger speeds relative to one another.

5.3.1 Inputs to the Extraction Algorithm

The steps of the motion extraction algorithm are shown in Figure 5.5. The algorithm takes as input the identified contact paths for the given hand. These paths contain the proximities and lateral velocities to be used in the motion calculations. The identifications are needed so that motion of certain fingers or palm heels which would degrade particular motion component calculations can be deemphasized. Since thumb motion is much more independent of the other fingers than the fingertips are of one another, scalings and rotations are easier for the operator to perform if one of these paths is from the opposable thumb. However, the extraction algorithm really depends only upon proper ordering of the finger paths. It will continue to function the same if the thumb is not present or is misidentified as a fingertip.

5.3.2 Scaling and Rotation Component Extraction

Since the weightings of particular fingers in the translation velocity average will depend on the polar component speeds, the polar velocity components must be measured from scaling and rotational motions before translation is measured. Unless a rotational velocity is extracted from changes in thumb contact orientation, at least two contacting fingers are necessary to compute hand scaling or rotation velocities. If less than two fingers from the hand are touching the surface, the rotation and scaling velocities are simply set to zero.

To further illustrate the unbalanced finger motions which occur during scaling and rotation, Figure 5.6 shows trajectories of each finger during a contractive hand scaling. The thumb (F1) and pinky (F5) travel in nearly opposite directions at roughly the same speed, so that the sum of their motions cancels for zero net translation, but the difference in their motions is maximized for a large net scaling. The central fingers (F2-F4) also move toward a central point but the palm heels remain stationary, failing to complement the flexing of the central fingers. Therefore

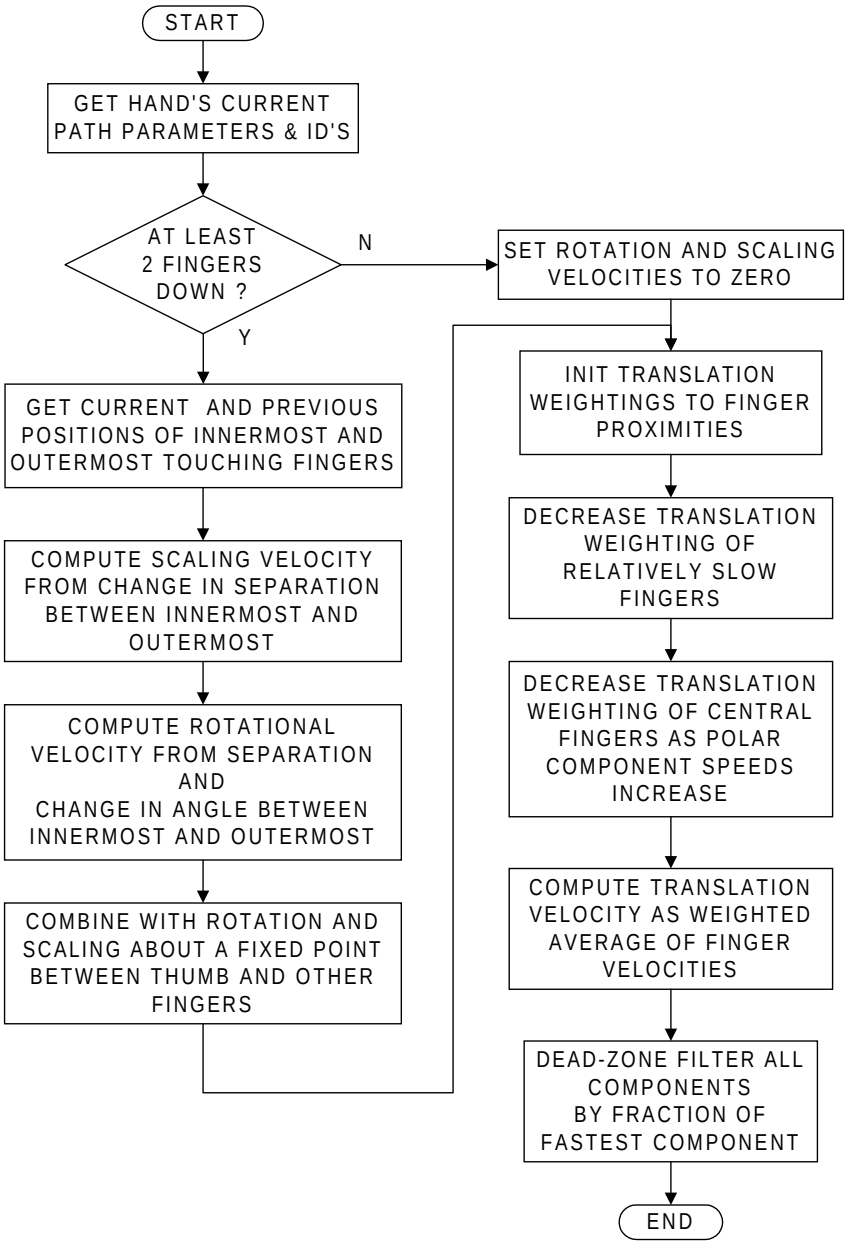


Figure 5.5: Flow chart of the algorithm for extracting hand scaling, rotation, and translation velocities from individual finger velocities.

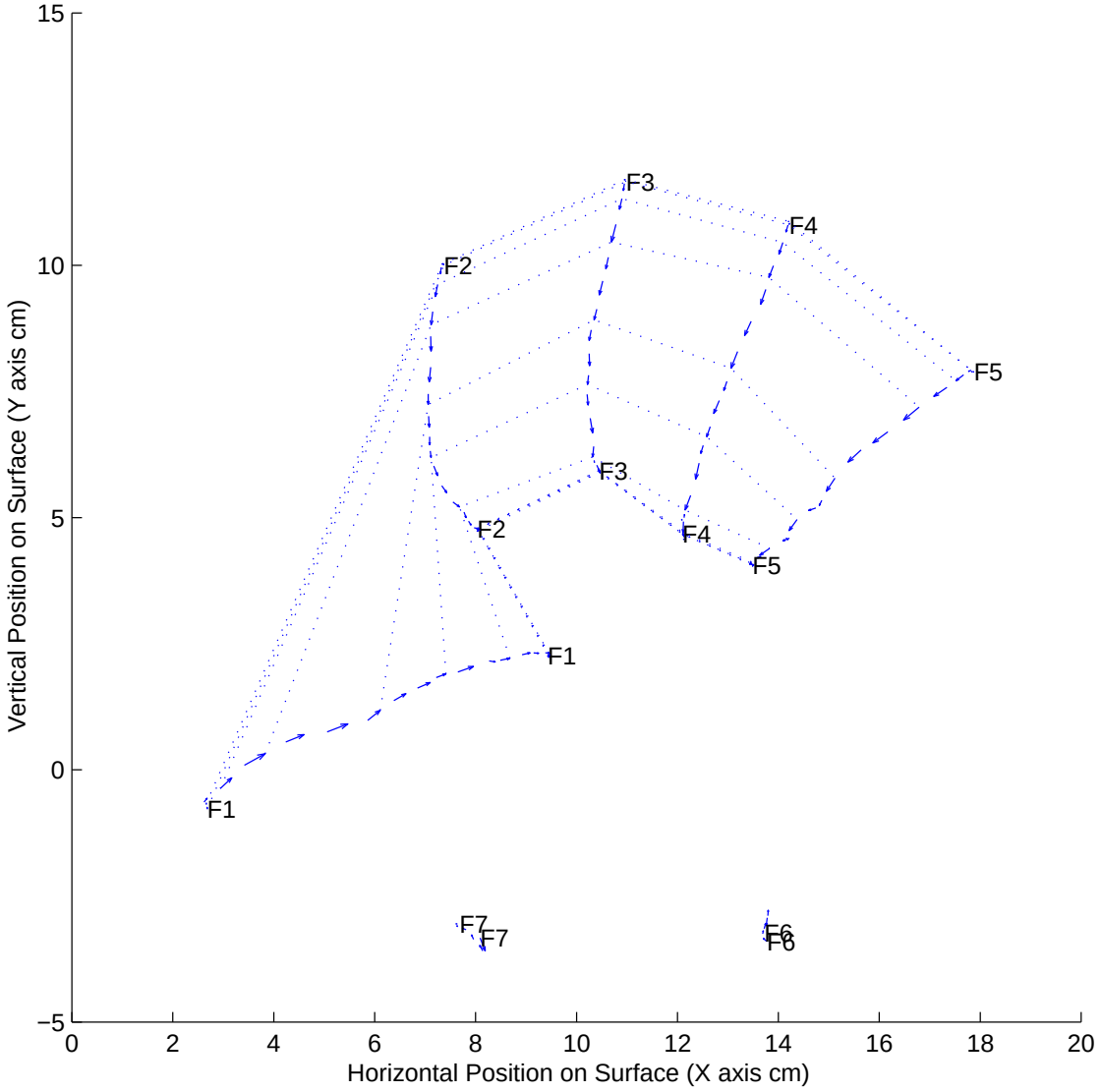


Figure 5.6: Typical flexing finger trajectories when performing a hand scaling. Note that if added as translations the thumb and pinky motions cancel, but since the palms are stationary, the motions of the central (index, middle, and ring) fingers go uncanceled and result in a net downward translation.

the difference between motion of a central finger and any other finger is usually less than the difference between the pinky and thumb motions, and the sum of central finger velocities during a hand scaling adds up to a net vertical translation. Similar phenomena occur during hand rotations, except that if the rotation is centered at the wrist with forearm fixed rather than centered at the forepalms, a net horizontal translation will appear in the sum of motions from any combination of fingers.

Since the differences in finger motion are usually greatest between thumb and pinky, only the current and previous positions of the innermost and outermost touching fingers are used for the initial hand scaling and rotation measurements. The hand scaling velocity H_{vs} is computed from the change in distance between the innermost finger FI and outermost finger FO:

$$H_{vs}[n] = \frac{d(FI[n], FO[n]) - d(FI[n-1], FO[n-1])}{\triangle t} \tag{5.4}$$

where $d(FI[n], FO[n])$ is the Euclidean distance between the fingers FI and FO. If one of the innermost or outermost fingers was not touching during the previous proximity image, the change in separation is assumed to be zero. Similarly, the hand rotational velocity H_{vr} is computed from the change in angle between the innermost and outermost finger:

$$\begin{aligned} H_{vr}[n] &= \left(\frac{\angle(FI[n], FO[n]) - \angle(FI[n-1], FO[n-1])}{\triangle t}\right) \\ &\quad \times \left(\frac{d(FI[n], FO[n])}{\pi}\right) \end{aligned} \tag{5.5}$$

The change in angle is multiplied by the current separation to convert it to the same units as the translation and scaling components. These equations capture any rotation and scaling components of hand motion even if the hand is also translating as a whole, thus making the rotation and scaling degrees of freedom integral with translation.

Another reason the computations above are restricted to the thumb and pinky or innermost and outermost fingers is that operators may want to make fine

translating manipulations with the central fingers, *i.e.*, index, middle, and ring, while the thumb and pinky remain stationary. If changes in distances or angles between the central fingers and the thumb were averaged with Equations 5.4–5.5, this would not be possible because central finger translations would cause the appearance of rotation or scaling with respect to the stationary thumb or pinky.

However, Equations 4.46–4.50 applied in the thumb verification process are only sensitive to symmetric rotation and scaling about a fixed point between the fingers. They approach zero if any significant whole hand translation is occurring or the finger motions are not complementary. In case the operator fails to properly move the outermost finger during a rotation or scaling gesture, equations of the approximate form of Equations 4.46–4.50 are applied between the innermost FI and any touching fingers $\{Fc : I < c < O\}$ other than the outermost to supplement the rotation and scaling velocities:

$$\begin{aligned} H_{vs_c}[n] &= -\sqrt{FI_{speed}[n] \times Fc_{speed}[n]} \\ &\quad \times \cos\left(FI_{dir}[n] - \angle(FI[n], Fc[n])\right) \\ &\quad \times \cos\left(Fc_{dir}[n] - \angle(FI[n], Fc[n])\right) \qquad (5.6) \\ H_{vr_c}[n] &= -\sqrt{FI_{speed}[n] \times Fc_{speed}[n]} \\ &\quad \times \sin\left(FI_{dir}[n] - \angle(FI[n], Fc[n])\right) \\ &\quad \times \sin\left(Fc_{dir}[n] - \angle(FI[n], Fc[n])\right) \qquad (5.7) \end{aligned}$$

The resulting velocities ($H_{vs_c}[n]$,$H_{vr_c}[n]$) are combined with the results of Equations 5.4–5.5 via a maximum operation rather than an average in case translational motion causes the fixed point rotations or scalings to be zero.

5.3.3 Translation Component Extraction

The simplest way to compute hand translation velocities would be to simply average the lateral velocities of each finger. However, the operator expects the

motion or control to display gain to be constant regardless of how many fingers are being moved, even if some are resting stationary. Furthermore, if the operator is simultaneously scaling or rotating the hand, a simple average is sensitive to spurious net translations caused by uncanceled central finger motions.

Therefore the translation component extractor carefully assigns weightings for each finger before computing the average translation. The translation weighting Fi_{vw} of each finger is first initialized to its total contact proximity, *i.e.*, $Fi_{vw}[n] \approx Fi_z[n]$. This ensures that fingers not touching the surface do not dilute the average with their zero velocities. Similarly, fingers which only touch lightly have less influence since their position and velocity measurements may be more noisy. The next step decreases the weightings of fingers which are relatively stationary so that the control to display gain of intentionally moving fingers is not diluted. This can be done by finding the fastest moving finger, recording its speed as a maximum finger speed and scaling each finger's translation weighting in proportion to its speed divided by the maximum of the finger speeds:

$$Fi_{vw}[n] := Fi_{vw}[n] \times \left(\frac{Fi_{speed}[n]}{\max_j Fj_{speed}[n]} \right)^{ptw} \tag{5.8}$$

where the power ptw adjusts the strength of the speed dependence. Note that this step can be skipped for applications such as computer-aided-design in which users desire both a normal cursor motion gain mode and a low gain mode. Lower cursor motion gain is useful for fine, short range positioning, and would be accessed by moving only one or two fingers while keeping the others resting on the surface but stationary.

The final weighting step decreases the translation weightings for the central fingers during hand scalings and rotations, though it does not prevent the central fingers from making fine translational manipulations while the thumb and pinky are stationary. The formulas below accomplish this seamlessly by downscaling the

central translation weightings as the magnitudes of the rotation and scaling velocities become significant compared to a speed constant named $K_{polarthresh}$:

$$Fc_{vwx}[n] \approx \frac{Fc_{vw}[n] \times K_{polarthresh}}{K_{polarthresh} + |H_{vr}[n]|} \tag{5.9}$$

$$Fc_{vwy}[n] \approx \frac{Fc_{vw}[n] \times K_{polarthresh}}{K_{polarthresh} + |H_{vr}[n]| + |H_{vs}[n]|} \tag{5.10}$$

These equations are applied only to the central fingers whose identities $\{c : I < c < O\}$ are between the innermost and outermost. Note that since hand scaling does not cause much horizontal translation bias, the horizontal translation weighting $Fc_{vwx}[n]$ need not be affected by hand scaling velocity $H_{vs}[n]$, as indicated by the lack of a hand scaling term in Equation 5.9. The translation weightings of the innermost and outermost fingers are unchanged by the polar component speeds, *i.e.*, $FI_{vwx}[n] \approx FI_{vwy}[n] \approx FI_{vw}[n]$ and $FO_{vwx}[n] \approx FO_{vwy}[n] \approx FO_{vw}[n]$.

With the translation weightings complete, the hand translation velocity vector $(H_{vx}[n], H_{vy}[n])$ is computed from the weighted average of the finger velocities:

$$H_{vx}[n] = \frac{\sum_{i=1}^{5} Fi_{vwx} Fi_{vx}}{\sum_{i=1}^{5} Fi_{vwx}} \tag{5.11}$$

$$H_{vy}[n] = \frac{\sum_{i=1}^{5} Fi_{vwy} Fi_{vy}}{\sum_{i=1}^{5} Fi_{vwy}} \tag{5.12}$$

5.3.4 Dead Zone Filtering

Despite the care taken to measure the rotation, scaling, and translation velocities in such a way that the resultant velocity components are independent of one another, uneven finger motion during hand scaling, rotation, or translation can still cause minor perturbations in measurements of one degree of freedom while primarily attempting to move in another. Non-linear filtering is necessary to remove the remaining motion leakage between dominant components and nearly stationary components. Each velocity component is passed through a separate dead-zone filter which produces zero output velocity for input velocities less than a speed threshold

but produces output speeds in proportion to the difference between the input speed and the threshold for input velocities that exceed the threshold. However, the speed threshold or width of each dead zone varies according to the distribution of current and past component speeds.

For instance, the width of the translation dead zone can be set to about 1/5 of the rotation or scaling speeds, whichever is greater. If the operator is primarily translating, this translation dead zone width will then be negligible compared to the actual translation speed, and the only effect will be to downscale the translation speed by a few percent. But if the operator is primarily rotating so that translation speeds are less than 1/5 of rotation speeds, the translation velocity component will be entirely suppressed to zero. Dependencies of the dead zone width on past averages of component speeds relative to one another provide filter hysteresis to ignore spurious transitions from hand rotation to translation or scaling.

5.3.5 Motion Extraction Results

Figures 5.7–5.9 show the four motion components for various whole-hand slides across the surface. In each plot, the dotted (green) line represents a simple average of finger translation velocities for the translation components. For the rotation and scaling components, the dotted line represents the average of the changes in angle or separation between all pairs of adjacent fingers, including the thumb-pinky pair. The dashed line (cyan) represents weighted averages in translation velocity for the translation components. For rotation and scaling, the dashed line is derived only from the change in angle or separation between thumb and pinky over time. The solid black line represents the finger-weighted (dashed) components after they have been passed through the variable-width deadzone filters.

In Figure 5.7, the right hand slides in a circle with fingers expanding and rotating counter-clockwise and then slides up while extending the fingers. Note how

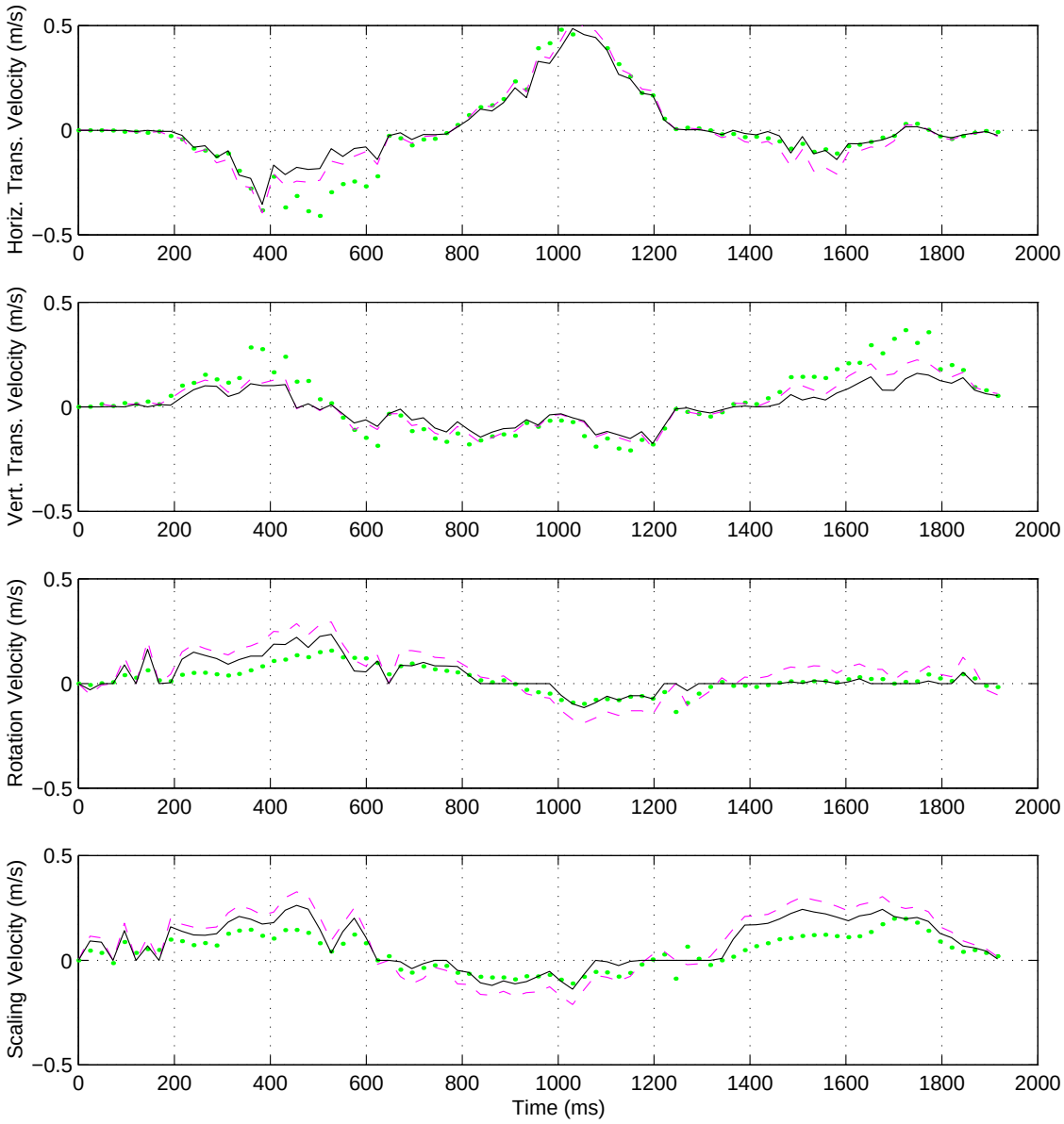


Figure 5.7: Velocity components extracted from simultaneous hand translation, rotation, and scaling. Up to 700 ms, a right hand slides in a circle with fingers expanding and rotating counter-clockwise. From 700–1300 ms, the hand continues in a circle while fingers flex and the wrist rotates back clockwise. From 1300–2000 ms, the hand slides up and expands at the same time. Dotted lines are uniform averages of finger motions, dashed are finger-weighted averages, and solid are finger-weighted averages after dead-zone filtering.

the finger-weighted translation components are much smaller than the simple average translations because they ignore the unbalanced motions of the central fingers during the strong rotations and scalings. The dead zone filters downscale the finger-weighted translations somewhat but never zero them altogether. For rotations and scalings, the thumb-pinky differences are actually stronger than the average differences even after dead-zone filtering because they are not diluted by the relatively weak changes in angle and separation between fingertips. Note that when only sliding up and extending the fingers, the dead-zone filtered rotation component remains zeroed most of the time.

In Figure 5.8, the whole hand slides roughly in a circle from the elbow while fingers and wrists remain relaxed. All versions of the translation components are nearly the same, though there is still slight leakage into the rotation and scaling components from slight shifts in relative finger posture. Nevertheless, dead-zone filtering is able to keep the rotation and scaling components zeroed most of the time.

In Figure 5.9, the fingers first extend and flex back smoothly. After being picked up briefly, the hand touches down again and rotates counter-clockwise and back clockwise. Though the finger extension and flexion cause noticeable interference in the translation and rotation components, dead-zone filtering again suppresses this. Notice the large vertical translation interference when the uniform average of all finger velocities is used. The hand rotation case is more troublesome. Filtering is only able to suppress interference with the other components about half the time. The thumb and pinky tend to separate as the hand becomes fully rotated clockwise, causing substantial crossover into the scaling components. Notice the large disturbances in horizontal translation when it is computed from the uniform average of finger translations.

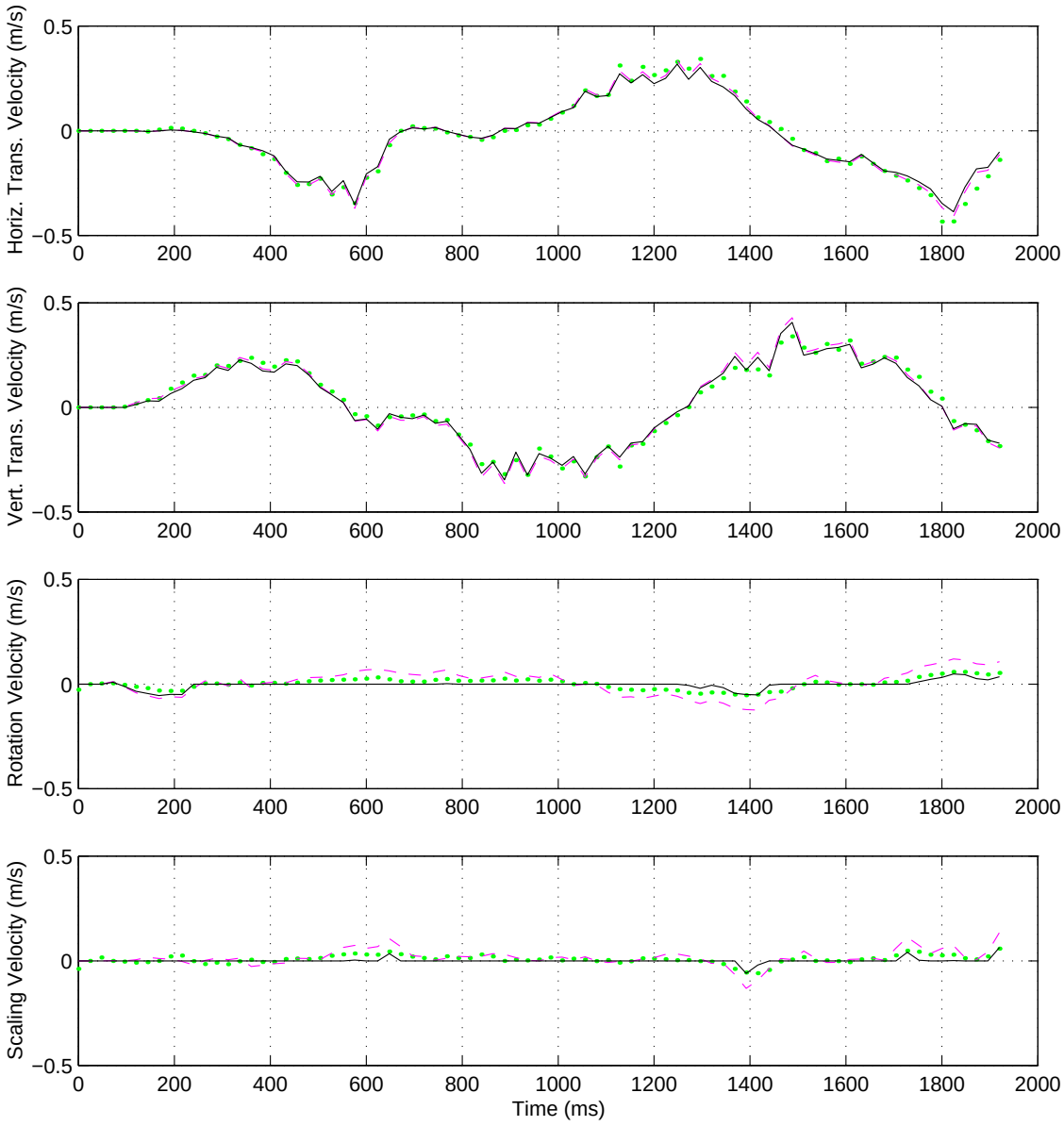


Figure 5.8: Velocity components extracted from whole-hand translation. The hand moves in a rough circle, causing the horizontal and vertical components to resemble sine and cosine waves. The fingers are not flexed nor the wrist rotated actively, but undoubtedly slight passive shifts occur in their posture as the hand slides. Dotted lines are uniform averages of finger motions, dashed are finger-weighted averages, and solid are finger-weighted averages after dead-zone filtering.

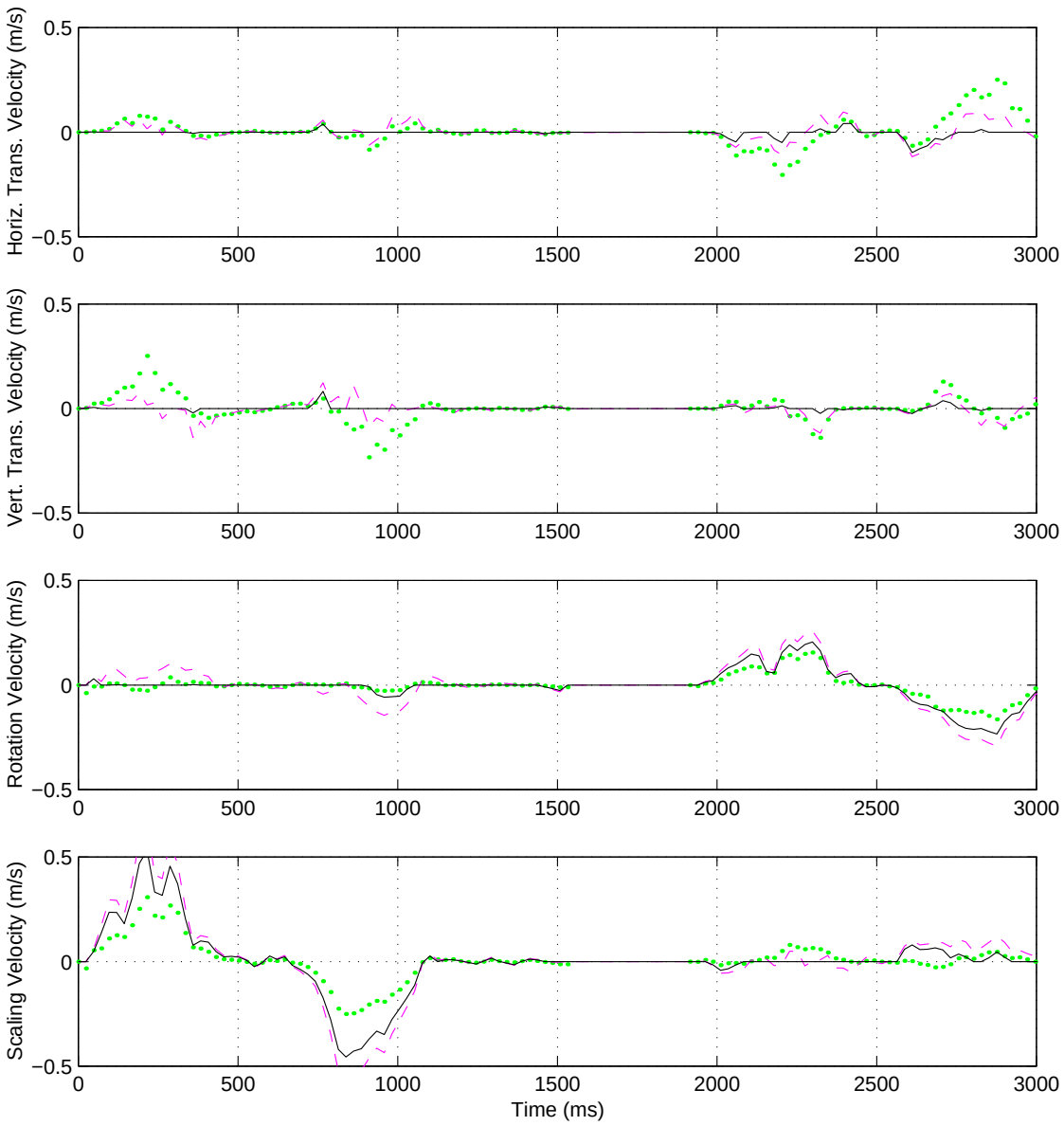


Figure 5.9: Velocity components extracted from separate hand rotation and scaling motions. Up to 1000 ms, the fingers extend and flex back smoothly. From 1000–3000 ms, the whole hand rotates counter-clockwise and then back clockwise. Dotted lines are uniform averages of finger motions, dashed are finger-weighted averages, and solid are finger-weighted averages after dead-zone filtering.

5.3.6 Motion Extraction Conclusions

Favoring the thumb and pinky motions while the hand is rotating or scaling greatly improves the independence of the extracted motion components. This will ultimately allow integral 4-DOF manipulation on the MTS. Though the currently implemented dead-zone filters successfully prevent leakage from non-uniform translational motions into rotation and scaling components, further optimization of dead-zone width dependencies will be necessary to completely suppress leakage of imperfect hand rotation or scaling motions into the extracted translation components.

5.4 Chord Motion Recognition

The chord motion recognizer is the final module of the typing and chordic manipulation system. It has the responsibility of determining from the combination of touching fingers which chordic manipulation the operator has selected at the beginning of a hand slide. Then, once the hand is in motion, it sends out appropriate command or manipulation events depending on the directions and speeds of the extracted motion components. Thus it requires as input the identities of all touching hand parts and the extracted hand scaling, rotation, and translation velocities. The chord motion recognizer also receives finger subset synchronization signals from the synchronization detector. Note that the chord motion recognition process is repeated for each hand independently.

5.4.1 Channel Selection

Am important question in the design of a chord motion recognizer is whether the selected channel should change in the middle of a slide if the combination of fingers touching the surface changes, or whether the channel selection be fixed at the beginning of a slide so later touchdowns or liftoffs of a finger or two have no effect.

5.4.1.1 Channels Follow Finger Combinations

The former solution is illustrated by the simple state diagram of Figure 5.10. This appears to be the chord motion state machine used on recent Logitech touch-

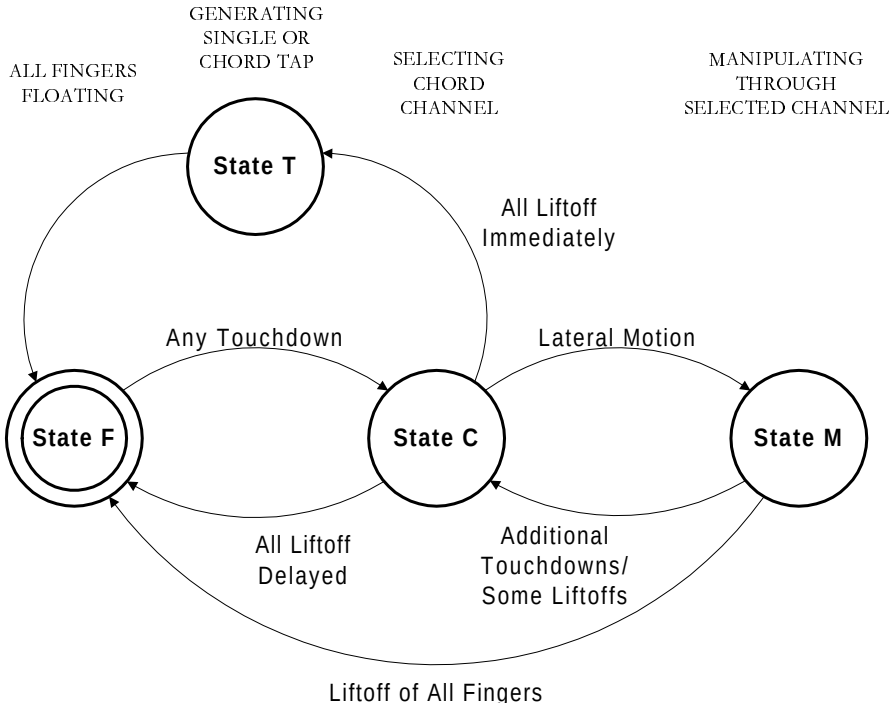


Figure 5.10: State diagram for 3-finger touchpad tapping and sliding.

pads [15] which detect and count up to three fingertips. When a single finger touches down, a transition occurs from the floating state F to the channel selection state C. State C counts additional touchdowns until lateral motion or total liftoff is detected. If the finger or fingers lift off quickly, a transition occurs to the tap state T, where a button click is issued. The identity of the emulated mouse button depends on the maximum number of fingertips counted while in the channel selection state. Likewise, when lateral motion is detected, a transition to the manipulation state M occurs. If the single finger channel has been selected, pointing events are generated in proportion to the motion; the two finger channel generates dragging events. However, the most notable characteristic of this state diagram is that if the number

of touching fingers changes without total liftoff, control returns to state C for an update of the channel selection. Thus the operator can transition immediately from pointing to dragging by dropping a second finger on the surface in the middle of a one-finger slide.

5.4.1.2 Initial Finger Combination Sets Channel

While this is an appropriate design for a small touchpad which has only three channels, and this design may be advantageous for certain application, it was not chosen for chordic manipulation on the MTS for a number of reasons. First, the MTS offers quite a few more chord channels (see Table 5.1) which are mapped to a much wider variety of commands and manipulations than just pointing and dragging. Switching channels upon every change in touching finger identity could be very confusing and accident-prone. For example, an accidental touchdown of the thumb during a three-finger slide could switch from a horizontal drag to issuing the "Back" command for a web browser. Fearing such accidents, operators might suspend those fingers not included in the chord high above the surface. This could be bad ergonomically, as typists who suspend their thumbs high above the space bar are prone to overuse injuries such as DeQuervain's syndrome [117].

Therefore, the MTS gives operators the freedom to drop any or all suspended fingers to the surface for whole hand manipulation once their initial motion and finger combination has selected a chord channel. Likewise, operators should be able to continue translations on a selected channel as long as at least one finger remains on the surface. Selecting a different channel always requires momentary liftoff of all fingers. Though such liftoff also requires some finger extensor effort, the MTS design assumes the effort of such liftoff for 100-200 ms to switch channels pales in comparison to the fatigue from holding certain fingers suspended above the surface for seconds at a time during slides. Another reason for tolerating additional touchdowns is that for channels whose initiating chords do not include the thumb,

operators can set the thumb down shortly after slide initiation to access the full dynamic range of the rotation and scaling degrees of freedom.

5.4.2 MTS Chord Motion State Machine

The MTS's more tolerant state machine design is diagrammed in Figure 5.11. The first difference from Figure 5.10 is that to distinguish slides from glancing finger taps during typing, the transition from state F to state C requires at least two fingers from a hand to be touching the surface. Thus a channel cannot be selected nor a chordic manipulation start from motion of a single finger. Similarly, chord taps require quick, synchronous release following synchronous touchdown of two or more fingers as previously described in Section 5.2.2.3. Single finger taps are of course interpreted as typing, which is not shown in the diagram.

5.4.2.1 State C: Channel Selection

State C continually checks for changes in the combination of fingers touching the surface and for lateral finger or hand motion. As with chord tap recognition in Section 5.2.2.4, the channel selector uses the combination of finger identities to look up a channel from Table 5.1 along with the channel's event mappings and motion sensitivity parameters. The motion sensitivity parameters include speed thresholds which determine how fast or how far the fingers must slide before triggering the transition to the manipulation state M.

When state C detects significant motion on all touching fingers and advances to the manipulation state M, the channel selection is locked in. Additional finger touchdowns or liftoffs will not affect the channel selection during manipulation unless they meet the special synchronization sequence to be discussed below. To prevent premature lock in of the channel before all fingers in a chord have reached the surface, finger and extracted hand motions are downscaled for about 50 ms after each new finger touchdown or release, making it less likely that the initiation speed

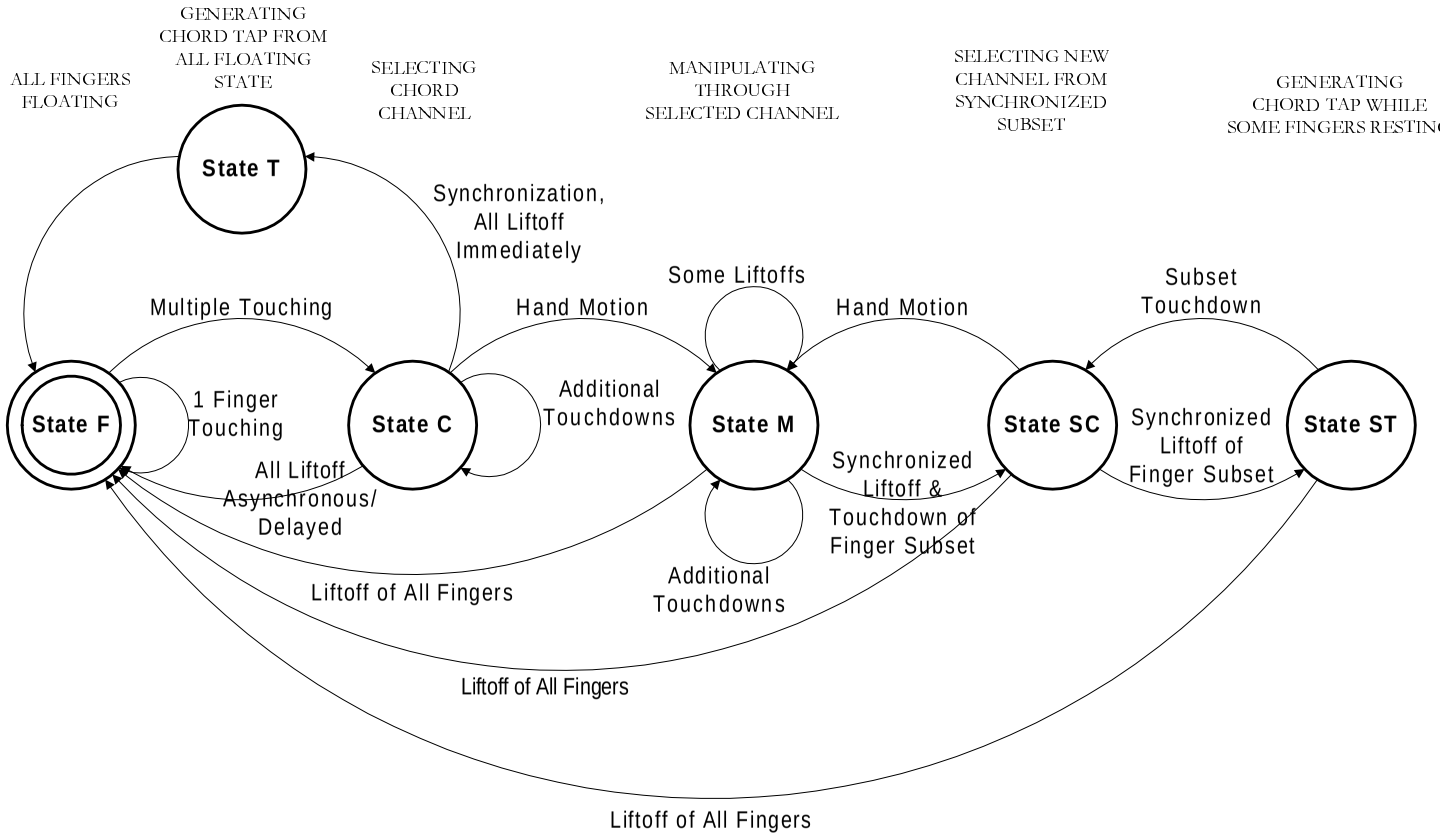


Figure 5.11: State diagram for the MTS chord motion recognizer. States C and T allow channel selection and chord tapping when the hand starts in state F floating above the surface. State M issues commands and manipulation events in response to hand slides, and states SC and ST allow a new chord channel to be selected and chord taps to be generated by synchronously lifting and dropping a subset of fingers when all or most fingers are resting on the surface.

threshold will be crossed until finger presence and identifications have stabilized. Also, the speeds of all touching fingers are required to pass the speed threshold and be within a fraction of neighboring finger speeds to ensure chord motion is initially coherent.

Note that there is no touchdown synchronization requirement for the transition from state C to state M. First of all, one is not necessary because coherent motion in all the touching fingers is sufficient to distinguish sliding fingers from resting fingers. Also, novice operators may erroneously try to start a slide by placing and sliding only one finger on the surface, forgetting that multiple fingers are necessary. Tolerance of asynchronous touchdowns allows them to seamlessly correct this by subsequently dropping and sliding the rest of the fingers desired for the chord. The manipulation mode will then initiate without forcing the operator to pick up all fingers and start over with synchronized finger touchdowns.

5.4.2.2 State SC: Synced Subset Channel Selection

States SC and ST provide a way to change channels when the hand is resting on the surface without lifting all fingers off the surface. This is important because otherwise operators may always tend to keep the hand suspended after being forced to lift it off to reset the state machine and change channels. The synchronized channel selection state SC can be entered from an existing channel manipulation mode M by synchronously lifting some of the fingers, usually just two or three out of five, and synchronously dropping them back to the surface. It can also be entered after all fingers have been resting on the surface without sliding, *i.e.*, from state C directly without going through M, but this transition is not shown in the diagram to avoid clutter.

Once in state SC, the new channel is determined from the combination of fingers in the synchronized subset, not from the combination of all touching fingers. From state SC a chord tap can be issued on the new channel through state ST by

once again raising and dropping the same finger subset. Likewise, coherent lateral motion of the fingers in this subset will cause a transition back to the manipulation state, irrespective of the motion of the resting fingers. This transition also locks in the new channel selection as before. Again, state SC offers the advantage that the operator does not have to lift a whole resting hand from the surface before starting a manipulation, but can instead leave most of the weight of the hands resting on the surface and only lift and press the two or three fingers necessary to choose the most common finger chords.

5.4.2.3 State M: Manipulation

The manipulation mode operates in several different ways depending on the type of manipulation or command events which have been mapped to the selected channel. For mouse pointing or dragging events, it simply integrates the extracted velocity components over small, regular time intervals and sends mouse motion packets to the host computer just like a mouse or touchpad would. For editing keys whose actions are reversible such as the arrow keys, it integrates hand motion in a particular direction. The motion recognizer then sends out the appropriate key when a threshold is reached, resets the integrators for each direction or arrow key, and begins integrating again. For one-shot commands such as cut which are not easily reversible and which seldom need to be repeated, the motion recognizer ceases integrating after the first issuance of the command, ensuring such commands are only issued once per slide. Thus the operator must pick up the hand and begin a new slide to perform a second cut. Table 5.2 shows the directional motion gestures which the chord motion recognizer currently implements.

5.4.3 Chord Mappings

The chord motion recognizers for each hand function independently, and the input events for each chord can be configured independently. This allows the system

Table 5.2: The simple manipulations and lateral motion gestures recognized by the MTS.

Motion Icon	*Type of Chord Motion*
	Brief tap on surface (one-shot).
	Translation (slide) in any direction.
	Reversible translation up or down.
	Reversible translation left or right.
	Reversible up or down translation, irreversible right translation.
	Translation in a particular direction (one-shot).
	Contractive hand scaling (one-shot).
	Expansive hand scaling (one-shot).
	Clockwise hand rotation (one-shot).
	Counter-clockwise hand rotation (one-shot).

to allocate tasks between hands in many different ways and to support a variety of bimanual manipulations. For example, mouse cursor motion could be allocated to the fingertip pair chord on both hands and mouse button drag to a three fingertip chord on both hands. This way either hand could point and drag on either half of the surface. Primary mouse clicks would be generated by a tap of a fingertip pair on either half of the surface, and double-clicks could be ergonomically generated by a single tap of three fingertips on the surface. Window scrolling could be allocated to slides of four fingers on either hand.

Alternatively, mouse cursor manipulations could be allocated as discussed above to the right hand and right half of the surface, while corresponding text cursor manipulations are allocated to chords on the left hand. For instance, left fingertip pair movement would generate arrow key commands corresponding to the direction of motion, and three fingertips would generate shift arrow combinations for selection of text.

For host computer systems supporting manipulations in three or more degrees of freedom, a left hand chord could be selected to pan, zoom, and rotate the display background while a corresponding chord in the right hand could translate, resize and rotate a foreground object. These chords would not have to include the thumb since the thumb can touch down anytime after initiating chord motion without changing the selected chord. The operator need only add the thumb to the surface when attempting rotation or scaling.

Finger chords which initially include the thumb can be reserved for one-shot command gestures. For example, the common editing commands cut, copy and paste can be intuitively allocated to a pinch or contractive hand scaling, a chord tap, and an anti-pinch of the thumb and an opposing fingertip. See Tables 6.1–6.4 on Pages 289–292 for the mappings used by the author for text editing and general navigation of standard graphical user interfaces.

5.5 Conclusions

This chapter has demonstrated how a system for integrating typing and versatile chordic manipulations on the MTS can be built upon the hand tracking and finger identification systems of Chapters 3 and 4. Robust path tracking from Chapter 3 is necessary for the finger press and release times used for finger subset synchronization detection to be reliable. Palm contacts must be identified as such so that they can be fully ignored by the synchronization detector, typing detector, hand motion extractor, and chord motion recognizer. Since manipulation channel selection depends upon the presence of the thumb and number of fingertips, finger and hand identification must also be robust and converge within about 100 ms of the first finger's touchdown. The hand motion extractor requires that the finger identifications remain in proper order from innermost to outermost.

This chapter has also introduced several novel concepts for human-computer interaction, the most fundamental being that typing can be distinguished from chordic manipulation over the key layout fairly reliably by checking for synchronization of finger motions on the same hand [160]. Full hand chords have been used here not for typing but to enhance graphical manipulation, which led to the problem of extracting four integral degrees of freedom from hand rotation, scaling and translation. A channel selection state machine has been designed to encourage hand resting on the surface. While the systems of this chapter are not yet bulletproof enough that anyone can walk up to the MTS and use it without training or practice, they already work quite well for a skilled operator, as will be seen in a case study of this author in the next and final chapter.

Chapter 6

PRELIMINARY EVALUATION, FUTURE DIRECTIONS, AND CONCLUSIONS

This chapter begins with a detailed testimonial of my experiences and a case study of my RSI symptoms while using the typing and chordic manipulation capabilities of the MTS to prepare this dissertation. Next, I outline usability, long-term fatigue, and RSI case studies which could more formally and objectively evaluate the efficiency and ergonomics of the MTS in the future. I end with a discussion of future enhancements to the MTS and commercial operating systems which would be necessary to support handwriting recognition or bimanual manipulation on the MTS.

6.1 Testimonial and Case Study of the Author

Though construction of the MTS was completed in November 1998, the MTS software did not function well enough to support daily use until early January 1999. Since then I have used the MTS as the sole input device on my primary personal computer to edit this dissertation and prepare results. My adviser, John Elias has also been using a second prototype since February 1999 as his primary input device. Here I will offer my impressions and observations from use of the MTS over this period, pointing out issues which should be examined in the future by more extensive, formal studies of user populations. Based upon the experiences of friends and colleagues who have tried the MTS momentarily, I will also discuss difficulties which novice operators are likely to have.

6.1.1 My Fitness as an Evaluator

As developer of the MTS software, I am undoubtedly somewhat biased, sometimes in ways I cannot foresee. For example, because I have a notion of how the typing and chordic manipulation are supposed to integrate, I unconsciously avoid motions which confuse the MTS algorithms. Every new person who has tried the MTS has at least one strange motion habit which I never anticipated and which demands enhancement of the motion filters. Because I received lessons in classical piano performance for twelve years, my manual dexterity and precision is undoubtedly above average. During my first typing class as a freshman in high school, I gained speed and accuracy much faster than the rest of the class, so the ease with which I have learned to operate the MTS would not be representative of the general population even if I were not its designer.

However, I do have a lot of experience evaluating input devices, especially for ergonomics. Over the years I have used the Kinesis contoured keyboard models 110-130, standard keyboards with various keyswitch stiffnesses, thumb and palm operated track balls, mice and touchpads. During my struggles with RSI, I have become very attuned to my body's pain signals, so that I can tell the difference between superficial muscle soreness, which can usually be ignored without consequence, and the deep, burning pain in the forearms which warns that continued use of the computer for another day can cause spiraling inflammation that incapacitates my hands for weeks.

Much of this ergonomics awareness has been learned the hard way, as when three-and-a-half years ago I failed to fix a malfunction in a left thumb roller device I had added to a Kinesis keyboard. After a week using this malfunctioning, unreliable roller intensely, I apparently tore a thumb adductor or index finger flexor tendon or sheath, an injury which plagues me to this day. Though I asked two doctors and several physical therapists for an exact diagnosis of this injury, none were able to

offer one. For the first few months after the initial injury I could not extend my left wrist past the neutral position. In the years since I have regained full wrist flexibility through stretching and strengthening exercises, but this old injury has remained troublesome, even as flare-ups in my other tendonitis hot spots such as the epicondylitis in both elbows have become less severe.

Even after staying pain free for weeks on vacation, I have not been able to type more than a page or two per day on any mechanical keyboard without causing pain and tenderness where the left index finger flexor tendon passes through the wrist. Though otherwise I prefer the Kinesis keyboard to a standard keyboard, the raised posture and editing keys it imposes on the thumbs seemed to exacerbate this injury more than a flat keyboard. Hence during the months prior to this MTS trial, while I was writing the MTS software using the Kinesis, my index finger pain and tenderness were actually worsening. Thus I can clearly compare my symptoms during this MTS trial to long-term symptoms leading up to the trial which other alternative input devices, physical therapy, and time had failed to eliminate.

Finally, a three-month case study such as this one can provide information on long-term use and effects that a short trial with a population of novices cannot. For example, I have used the MTS long enough for frequently-performed gestures to enter motor memory. Thus I can easily distinguish the truly useful chordic manipulations from those performed so rarely that I must still pause to remember them.

6.1.2 Equipment and Methods

The MTS was connected to an IBM-compatible PC with a 200 MHz Cyrix processor running IBM's OS/2 operating system. Since the MTS emulates PS/2 keyboard and serial mouse protocols in hardware, no custom device drivers were necessary. However, an operating system extension called Hotscroll by Samual Audet [4] enabled continuous scrolling of windows via emulated mouse events.

This dissertation was written in the LaTeX typesetting language rather than with a conventional word processor; therefore, formatting codes, references and text were all included in ASCII text files editable with any common text editor. I used an advanced programmer's editor, Visual SlickEdit by MicroEdge, for this purpose. It has a number of features which make editing more efficient, such as interactive word and line completion, infinite undo and redo, and cut or copy of the line containing the text cursor without first selecting it.

Though the MTS was the sole input device for my primary workstation, at times I relied on other means to input text and graphical data. Many of the figures were plotted with Matlab running on a Sun workstation with a Sun 4 keyboard and Mouse-trak trackball attached. Also, I wrote out the first draft of long sections of the text by hand and had a person type them in for me. Then I performed all editing and page-sized additions through the typing and chordic manipulation capability of the MTS prototype. Because of the slowness and inaccuracy of the speech recognition software available for OS/2 (IBM's Voicetype discrete dictation software), I avoided use of speech recognition software during this period, preferring the MTS for entry of small-to-moderate amounts of text and a typing assistant for large new sections. Nevertheless, many days I typed 3-6 pages with the MTS, more than I had been able to type with a mechanical keyboard prior to the MTS trial without causing significant pain for several days.

6.1.3 Typing

I was able to touch type [158] at my normal speed of up to 60 words per minute on the MTS, making about twice as many errors as I would on a mechanical keyboard. Because it was so easy to edit and correct errors with the MTS, this increased error rate did not become an annoyance. However, entering text such as C code which contained a lot of numbers or punctuation from the periphery of the

key layout usually required a glance at the symbols printed on the MTS, which did become annoying at times.

Figure 1.1 on Page 6 is a rendition of the QWERTY key layout used for the first month of the MTS trial. I drew the key symbols on the surface with a marker, but the surface was perfectly smooth, so there was no tactile indication of their location. Therefore, to type with decent accuracy, the fingers either had to remain resting on home row, picking up and placing one finger at a time, or they had to be carefully suspended in the air above home row without drifting. With the MTS on my lap sloping downward it was also comfortable to rest my palms on the surface while the fingers hovered over home row. Both John Elias and I have successfully learned to do this, but it is apparently not natural. All the people who have momentarily tried to type on the MTS without any tactile feedback have been able to hunt and peck by looking at the symbols drawn on the surface but have not been able to touch type without looking. One conflating issue is that the alphabetic key columns are straight vertical on the MTS, like on the Kinesis, rather than slanted as on a conventional QWERTY keyboard. In any case, two issues need to be examined in the future:

1. How hard is it for people to learn to keep their hands steady over home row, *i.e.*, how long does it take them to learn and can everyone learn?

2. How much tactile feedback of the key and home row positions is necessary to speed adaption to and accuracy of touch typing on the MTS?

The second question needs to be determined very precisely because more tactile feedback implies a rougher surface. Too rough a surface will impede smooth chordic manipulation.

6.1.4 Weekly Symptoms

6.1.4.1 First Two Weeks

During the first week or two using the device, I was still recovering from prior use of the Kinesis keyboard. All-day use of the MTS would still cause superficial inflammation and soreness by the end of the day, but deeper pain did not build up from one day to the next like it recently had with the Kinesis. I had the feeling that I was just mildly irritating already-inflamed tendons with the MTS, not making the inflammation worse in the long-term.

However, during the first week or two I experienced acute neck and shoulder pain. I attributed this to MTS software bugs which would not allow me to rest my hands on the surface without causing spurious key activations. Bugs in the chordic manipulation system also made pointer manipulation somewhat erratic or unresponsive. I then spent a couple days fixing these bugs and enabling all five fingers to drop to the surface in the middle of manipulation without switching or disabling the selected channel.

6.1.4.2 Third and Fourth Weeks

At this point my neck and shoulder stiffness went away. Unfortunately I cannot say whether this was because my body had finally adapted to the slightly different postural requirements of the MTS or if it was due to my improvements in system reliability and resting hand tolerance. As I continued use of the MTS over the next couple weeks, all the burning in my forearms which usually accompanies moderate typing went away except the tenderness in my left index finger flexor. Because of this my index finger was reluctant to stretch off home row for keys such as the 't' key, and often made typing errors.

6.1.4.3 Fifth and Sixth Weeks

During the fifth week of the trial I decided the only way to improve my typing endurance in the face of this nagging index finger injury would be to learn a key layout which utilized the index finger less. In the QWERTY layout, the left index finger handles 17.75% of the English typing load, more than any other finger on either hand [101]. The Dvorak layout, on the other hand, only allocates 12% of the typing load to the index finger, and most of that load is from home row keys. Changing the MTS layout only required swapping of key centroid positions in a configuration file and redrawing the symbols on the surface with a marker. After the first couple days I realized that the Dvorak layout places the 'i' key to the right of the index finger home key even though the letter 'i' appears twice as often in English as the index home key 'u'. Presumably, Dvorak did this to optimize digraphs involving either of these vowels, but since I wished to minimize index finger stretching, I swapped the 'u' and 'i' keys. To speed relearning in the transition from QWERTY to Dvorak, I also kept some of the rarer letter and punctuation keys in their QWERTY positions. Figure 6.1 shows the modified version of the Dvorak layout which I finally settled upon.

As most people who have learned Dvorak after QWERTY have noted, the first week or two of the transition is quite disorienting. The lack of tactile feedback from the MTS probably exacerbated this. Having no motor memory yet of finger motion sequences on the Dvorak layout and no key edges with which to feel around for keys, I once again found myself constantly holding my hands above the surface and looking down to find my way around the foreign layout. This caused a resurgence in my neck and shoulder stiffness and some arm fatigue.

6.1.4.4 Seventh and Eighth Weeks

However, within about two weeks, typing on the modified Dvorak layout started becoming automatic, and these neck and shoulder symptoms subsided. Most

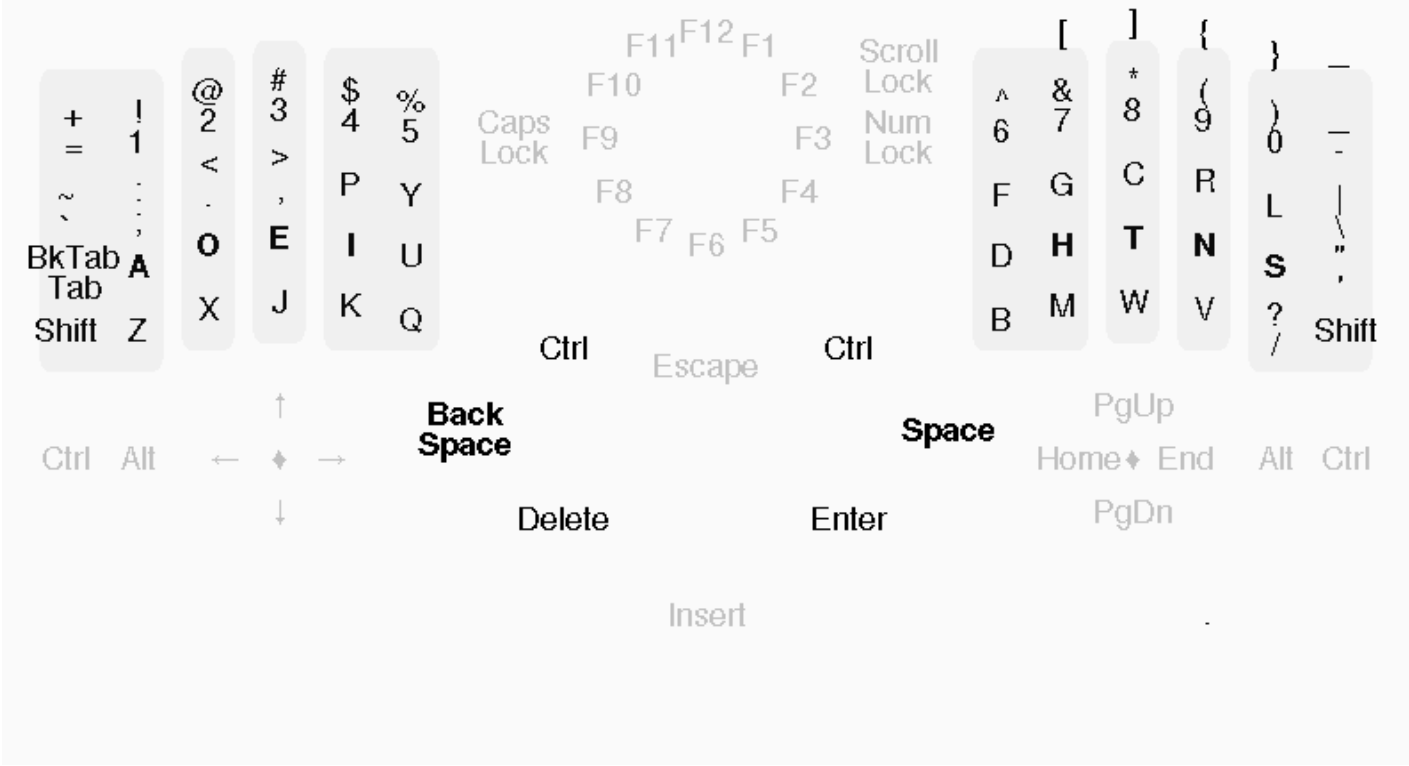


Figure 6.1: Modified Dvorak key layout adopted by the author starting in the fifth week of the trial. Note 'i' and 'u' are swapped from normal Dvorak to decrease frequency of reach by left index finger. Also, 'x' and 'q', '.' and ',', 'z' and '/' and ' and ';' are swapped to decrease differences from QWERTY for punctuation and infrequent alphabetic keys.

importantly, the reduced index finger workload gave my index finger flexor tendon a chance to recover. Since the first week after switching key layouts, my only mild recurrences of pain or tenderness in this finger or my other weak spots have occurred after typing 4-5 pages on consecutive days.

6.1.4.5 Ninth and Tenth Weeks

Since I was finishing this document, but my typing assistant was unavailable, I typed 3-6 pages per day nearly every day of this two week period. This caused sporadic tenderness in my index finger tendon but no other pain, and the index finger never became so aggravated that I would have had to rest for days at a time, which I could not have afforded. This period was the first time in four years that I felt free to compose large amounts of text through typing rather than handwriting.

6.1.4.6 Conclusions

These experiences suggest that though I may not have infinite endurance typing on the MTS, I have two or three times as much endurance as on a mechanical keyboard. This is consistent with my previous experiences with zero-activation force devices such as optical mouse buttons, Pilot rolling ball pens, and touchpads. Minimal force devices still do not allow indefinitely long, intense use or unlimited repetitions, but I can operate them 2-4 times as long per day before fatigue or inflammation begins to build up.

Though my brief spouts of neck and shoulder pain could have been brought on by other activities such as watching movies with my head propped against the high armrest of a couch, they suggest that postural loading on the shoulders may increase temporarily if operators cannot rest their hands on the surface or are unfamiliar with the key layout.

6.1.5 Recognition Errors and Accidental Activations

While the essentially zero activation force of the MTS has obvious ergonomic advantages, it also makes accidental activation by accidental contact with the surface more likely. Though one can synchronously place palms and all five fingers on the surface at any time without consequence, one has to be careful not to accidentally tap the surface with a finger. As a skilled operator who knows how to undo any accidental activations, this has not really been a problem. For me, the number of errors due to accidental touches is about the same as the gesture recognition error rate. For example, a few times a day the system will misinterpret a primary-clicking finger pair tap as a thumb-finger tap or a hand scaling gesture as a rotation gesture. Since the thumb-finger tap is mapped to copy, the former error will have no consequence unless one is in the middle of a cut-paste sequence, in which the clipboard contents could be overwritten. Similarly, a couple of times a day I will accidentally touch the surface inserting a key while actually trying to float above the surface. Since the command keys are in the middle of the MTS and the hands are usually near home position, most such accidental activations involve insertion of an alphabetic character rather than a command invocation by a function key. Thus they are easy to correct as long as the error is noticed. Audible feedback from all key activations ensures such insertions will be noticed in most cases.

The accidental activation problem is potentially much more serious for novices. First, they may not know to precisely synchronize their chord taps or avoid accidental touches of less than five fingers. Second, when they do accidentally activate a key or chord command, they may become confused and not know how to undo their error. Several modifications have already been made to the MTS software to address this, and undoubtedly more will be needed in the future.

Disabling all but the most basic chordic manipulations when novices are first learning MTS operation will avoid many accidental activations and much confusion.

It is especially important to disable most chord taps so novices can rest fewer than five fingers without generating spurious commands or mouse clicks. Novices trying the system have been observed to accidentally activate the 3-finger chords during typing and hand resting. I have these mapped to Escape on the left hand and double-click on the right, and while these may be convenient for me, Escape is also a key on the key layout, and successive finger pair taps also emulate double-click, so novices can access these commands more safely without having them mapped as chord taps. Only the right hand finger pair tap mapped to primary mouse clicking is really needed for basic operation. The thumb-finger tap for the copy command is particularly useful and less likely to be activated accidentally than three or four finger chord taps. Once a person becomes more comfortable with the synchronization requirements of MTS operation, they can map and memorize additional chord taps as they see fit.

6.1.5.1 Benefits of Higher Frame Rates

More selective touch filtering can also prevent accidental activations. To be reliable, such filtering will require proximity image frame rates on the order of 100 fps instead of the current 50 fps. For example, the proximity threshold for key activation can be raised given higher frame rates without requiring more forceful fingertip impacts by the operator. This threshold is currently set to about one-sixth the average fingertip tap proximity, not because some key taps are that light, but because the peaks in proximity of extremely quick taps can fall between the current 20 ms scanning cycles. Since the peaks in proximity as the finger bottoms out can be missed, sometimes only the slight proximities at the ends of the tap life cycle are detected. Raising this threshold will lower the likelihood that accidental brushes against the surface will be interpreted as keypresses. Other options include measuring the impulsiveness of finger touchdown to ignore hesitant, unintentional

taps or stringently adapting debounce timing requirements to the operator's average tap speed.

Synchronization detection (Section 5.2.2) can also benefit from a higher frame rate. Sometimes when tapping two horizontally adjacent keys in quick succession, taps will be nearly synchronous, causing them to be erroneously interpreted as a finger pair chord tap. Since the most frequent characters are scattered so far apart in the QWERTY key layout, this does not happen that often. I observed it a couple times for the 'oi' bigram. However, in the Dvorak layout frequent bigrams are much closer together, the most notable being 'th' allocated to the middle and index finger of the right hand. Therefore if I am not careful about my finger timings, the system often misinterprets 'the' as a primary mouse click followed by an 'e'. The 50 fps frame rate is simply too slow to reliably determine that my 'th' taps are in fact slightly asynchronous.

6.1.6 Chordic Manipulation Performance

Even though I have only used the chordic manipulation system for editing and desktop navigation so far and have not yet explored its 3D navigation potential, I have found it to be just as effortless, efficient, and fluid as I had envisioned. I quickly became so accustomed to the ability to switch instantaneously between typing and pointing, dragging or scrolling as to take it for granted. I frequently used the channels for mouse and text cursor pointing, selection by mouse or text cursor, and scrolling. Though the channels for manipulation of Visual Slickedit features such as word and line completion or infinite undo and redo were not used as frequently, being able to "roll back" a document to any previous stage of editing as easily as moving a cursor was intriguing. Using a stand-alone pointing device now seems oddly primitive and inhibiting.

While I became unconsciously dependent on the cursor manipulation channels, performing the one-shot command gestures is positively fun because they can

accomplish so much with quick series of small, precise motions. In other words, they enable the "chunking and phrasing" of gestures espoused by Buxton [20–22]. As a replacement for keyboard hotkeys and macros, they are just as easy to learn, but require much less effort than pressing a key. Moreover, since recognition of chordic command gestures is independent of absolute position on the surface, they can always be performed at the current hand position without reaching or looking for a particular region of the surface.

Tables 6.1–6.4 detail the chord tap and motion mappings for each hand which were programmed into the MTS during my trial. Refer back to Pages 9–10 for Tables 1.1–1.2 containing legends for the chord channel and motion icons.

The mapping tables list the finger combination and motions necessary to generate the noted command or manipulation. Each command and manipulation is rated according to how useful it was for me, *i.e.*, how frequently I performed it, how useful or necessary it might be for general GUI manipulation by novices, and how safe it is for novices. By safety I mean how prone the finger combination and motion is to accidental activation or recognition errors, not how ergonomic it is. Ratings are on a scale of 0-5, with 0 being the worst and 5 the best. Combinations which have a high novice necessity rating but a low novice safety rating definitely require further improvements to the motion or accidental activation filters.

I have used the one-shot command gestures for word search and replace, file save, window close, browser back and forward, and code compile often enough for them to become automatic. However, it is the cut, copy, and paste gestures which really transform the editing experience since they function so well in tandem with the cursor manipulations. For example, after performing a right hand 3-fingertip drag to emulate object selection via mouse or a left hand 3-fingertip drag to emulate text selection via arrow keys, one can copy the selected object with a thumb-fingertip chord tap, reposition the mouse or text cursor with a fingertip pair manipulation by

Table 6.1: Mappings for right hand manipulation channels.

Right Hand Channel	*Chord Motion*	*GUI Action*	*Useful for Author (0-5)*	*Needed by Novices (0-5)*	*Safe for Novices (0-5)*
••		Primary mouse button click.	5	5	3
••		Mouse cursor manipulation.	5	5	5
•••		Primary mouse button *double*-click.	5	3	1
•••		Dragging/Selection via primary mouse button.	5	5	4
••••		No mapping to avoid accidents.	-	-	0
••••		Continuous scrolling/panning of current window.	5	4	4
•••• '		Key layout homing.	2	1	1
•••• '		No mapping to tolerate shifts in resting hand posture.	-	-	0

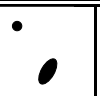
Table 6.2: Mappings for left hand manipulation channels.

Left Hand Channel	*Chord Motion*	*GUI Action*	*Useful for Author (0-5)*	*Needed by Novices (0-5)*	*Safe for Novices (0-5)*
••		No mapping to avoid accidents.	-	-	1
••		Text cursor manipulation via arrow keys.	4	3	3
•••		Escape key (cancel command).	1	1	0
•••		Selection via text cursor (<shift>arrow keys).	4	3	3
••••		No mapping to avoid accidents.	-	-	0
••••		Page Up and Page Down keys.	4	3	2
••••		Home (beginning of line) key.	2	2	2
••••		End (of line) key.	2	2	2
•••• /		Key layout homing.	2	1	1
•••• /		No mapping to tolerate shifts in resting hand posture.	-	-	0

Table 6.3: Mappings for right hand command gesture channels.

Right Hand Channel	*Chord Motion*	*GUI Action*	*Useful for Author (0-5)*	*Needed by Novices (0-5)*	*Safe for Novices (0-5)*
		Cut (to clipboard).	4	4	5
		Copy (to clipboard).	5	5	3
		Paste (from clipboard).	5	5	5
		Interactive word completion.	4	2	2
		Popup word completion list.	1	1	2
		Secondary mouse button click (popup menu).	3	4	3
		Dragging/Selection via secondary mouse button.	3	3	5
		Popup application window list.	3	2	2
		New file.	3	2	4
		Open file dialog.	1	2	4
		Save the current file.	4	2	4
		Close the current file or subwindow.	4	2	4

Table 6.4: Mappings for left hand command gesture channels.

Left Hand Channel	*Chord Motion*	*GUI Action*	*Useful for Author (0-5)*	*Needed by Novices (0-5)*	*Safe for Novices (0-5)*
		Cut (to clipboard).	4	4	5
		Copy (to clipboard).	5	5	3
		Paste (from clipboard).	5	5	5
		Undo or Redo.	3	2	2
		Alt (to access menubar).	1	1	2
		-> (the pointer punctuation for C code).	3	0	4
		_ (the underscore character).	3	1	4
		Find (<ctrl>F).	3	2	4
		Replace (<ctrl>R).	3	2	4
		Make project (compile code).	3	1	4
		Next or previous subwindow.	1	1	3
		Next or previous virtual desktop.	1	1	3

either hand, and paste the selection with a thumb-fingertip anti-pinch, all with the hand nearly staying in place. While this sequence may sound complex, the entire sequence can be performed in 3-5 seconds if the selection is only to be moved a short distance, about half the time it would take if cut and paste buttons had to be accessed off a toolbar. These MTS editing gestures match the convenience of the drag-and-drop capability of the Macintosh and the middle-mouse-button paste feature common to the X11 Windowing System, yet they are much more flexible since one can choose to either cut or copy the selected text to the clipboard and not worry about losing the clipboard after moving the text cursor.

Replacing a word with another word from a different paragraph is a particularly quick variation of such copy and paste sequences. One simply performs a 3-fingertip tap to emulate double-click and select the replacement word, copies the selection with a thumb-fingertip chord tap, moves the mouse over the word to be replaced, performs another 3-fingertip chord tap to select it, and a thumb-finger anti-pinch to paste. The only motions in the sequence limited by Fitts' law (Section 5.1.1) involve positioning the mouse cursor over the words. The other motions are instantaneous, open-loop gestures unaffected by Fitts' law.

The best evidence I can give of the integration efficiency of cursor manipulation and editing commands on the MTS is that I now rely much more on the mouse cursor than I ever did before. Always before I shunned editing with the mouse cursor in favor of keyboard hotkeys and the text cursor manipulation rollers I had installed on my Kinesis keyboard. Even using a touchpad next to the Kinesis was just too inefficient and tiring for constant manipulations. But since the one-shot editing gestures integrate equally well with mouse or text cursor manipulation on the MTS, I am no longer reluctant to use the mouse cursor. The following factors may also contribute to my new tolerance for the mouse cursor:

- the MTS provides a much larger surface over which to manipulate than the

tiny touchpads.

- the MTS dedicates a manipulation channel to emulation of double-clicks and tap drags so the operator need not perform these awkward motions.
- the MTS eliminates homing motions between the mouse and keyboard.
- the 4-fingertip scrolling channels obviate tasks such as scrollbar manipulation which are awkward using the mouse cursor.

The freedom to use any combination of two fingertips for pointing also turns out to be quite a boon. I often alternate between the index and middle pair, the middle and ring fingertip pair, and the ring and pinky fingertip pair to vary my hand posture. Such alternation does not seem to cause any cognitive confusion; remembering to use a particular number of fingertips seems as easy or easier than remembering to use a particular set of fingertips. I often prefer using the ring and pinky pair because they require less forearm pronation than sliding the index and middle fingers while holding up or curling under the ring and pinky fingers.

6.2 Future Evaluations

Formal evaluation of the MTS can proceed along three tracks, the first being short-term usability trials, the second being RSI case studies, and the third being fatigue studies.

6.2.1 Usability Trials

The goal of usability trials would be to measure how quickly people learn and adapt to the MTS and how efficient they become at interactive editing tasks. Depending on the target market, the tasks could involve editing term papers, Java source code, CAD drawings, or color images. A trial would consist of four ninety-minute sessions spread over four consecutive days, plus a follow-up session a week

Table 6.5: Schedule for MTS usability study.

Activity	*Day 1*	*Day 2*	*Day 3*	*Day 4*	*Day 11*
Pretest: Standard	√				
MTS	√	√	√	√	√
Tutorial	√				
Practice	√	√	√	√	√
Post-test: Standard				√	√
MTS	√	√	√	√	√

later to measure retention [128, 133] of learned chordic manipulation skills. About twelve students who were already familiar with software for the chosen editing task would participate in each study. To encourage uniformity in the subject's backgrounds, the students would preferably be members of the same composition, programming, drafting or art class, and the trial would take place in the middle of the semester when they were already familiar with the requisite editing software.

Table 6.5 shows the daily session schedules for the usability trials. At the beginning of the first session, subjects' performance with a standard mouse and keyboard would be evaluated for about 15 minutes. Then untrained performance on the MTS would be evaluated for the same amount of time. The performance measure would be the time taken to make pre-specified changes to a sample document. Erroneous mouse or key activations would also be logged. For the next 15 minutes the subjects would participate in an interactive tutorial of MTS operations such as typing, pointing, dragging, and scrolling. They would also be taught one-shot gestures for a few commands such as cut, copy, paste, file save, and file close. For the following 30 minutes, subjects would be allowed to work on standardized class assignments using the MTS to develop familiarity with it. During the last 15 minutes of the session, subjects' performance on the MTS would be evaluated with another sample document.

Each session on subsequent days would begin with a 15-minute evaluation on the MTS, allow uncontrolled use of the MTS on class assignments for 60 minutes, and end with another performance evaluation in the last 15 minutes. To control for learning of the editing task rather than learning of MTS operation, the fourth session and retention sessions would end with another evaluation of performance on standard mouse and keyboard. At the end of the first, fourth, and retention sessions, subjects would be asked to fill out a questionnaire with their impression of MTS operation.

To discourage growth in familiarity with the evaluation tasks, a different sample document requiring similar changes would be used in each successive evaluation. Also, sample documents would be presented to each subject in different, random order across sessions to control for variations in the difficulty of edits to each document. To force subjects to adopt the most basic one-shot command gestures such as copy and paste, alternative methods for invoking the commands such as menus and hotkeys would be disabled for all sessions.

Such a study would establish the learning curve for MTS adoption by people already familiar with computers. Logs of all hand activity on the MTS during the evaluations would also indicate which MTS activities are the most troublesome to learn. Finally, such a study would determine if people can achieve greater efficiency on the MTS or some subset of MTS operation than on a standard keyboard-mouse combination in just a few days.

6.2.2 RSI Case Studies

The subjects in the RSI case studies would be composed of individuals like myself who have a history of moderate-to-severe repetitive strain injuries to the hands or forearms. Subjects would maintain a daily log of pain symptoms throughout the study. They would also be asked to fill out using the computer a standard pain questionnaire or scale [18, 28, 124] at the beginning and end of each work day.

Subjects would continue to use their conventional input devices during the first week or two of the study to firmly establish reference symptom levels. Subjects would receive two one-on-one tutorial sessions to learn basic motions and postures so they could use the MTS to their best advantage. Then subjects would adopt the MTS for all computer use and continue to use it for several weeks.

The MTS software would be adapted as necessary to accomodate particularly severe injuries such as my left index finger tendonitis. Hand activity logging software on the MTS would keep track of total repetitions and intensity of usage each day so subjects would not have to estimate their time at the computer each day. Such logs might expose phenomena such as increases in endurance or daily MTS usage while reported pain remained constant. After pain symptoms and MTS usage patterns had reached a plateau for at least two weeks, subjects would be asked to switch back to their former input devices for a week or two to check for recurrence or changes in symptoms.

Unfortunately, conducting truly "blind" studies on the ergonomic effectiveness of the MTS would be impossible because its shape and feel are radically different than those of other input devices. One way to root out possible placebo effects from switching to a new device would be to have some subjects switch from their standard keyboard to an equally radical keyboard such as the Kinesis instead of the MTS. Presumably all subjects would experience short-term worsening of symptoms while getting used to their new devices, but medium-term improvement from either device would probably follow. The hypothesis would be that in the long-term, the minimal activation force of the MTS would allow more complete recovery, especially under heavy workloads.

6.2.3 Typing Fatigue Studies

One goal of the fatigue study would be to determine how the muscle work load during typing on the MTS is distributed about the body. Since chordic manipulations can be performed with at least part of the hand weight supported by the surface most of the time, and chordic manipulations require minimal activation force, they are expected to be less fatiguing than any other method of graphical manipulation or command gesture entry. Similarly, finger flexor exertion will almost surely be minimal when typing on the MTS. However, it is possible that finger extensor and shoulder muscle exertion may be more than for a standard keyboard during fast typing since the fingers must remain fully suspended above the surface to make quick strokes. This is likely to depend on whether subjects rest their palms on the surface while typing, and if they do rest their palms, the downward slope of the surface may be important. Therefore the study should measure through muscle electrical activity the relative exertions of various muscle groups during typing on the MTS. It would also measure how patterns of hand resting and finger impact forces change as subjects adapt to the MTS.

Ten subjects with good typing skills such as secretaries or transcription typists would be chosen. The study could be modeled closely after Gerard's long term studies [45] on the effects of keyswitch stiffness and auditory feedback on typing forces. The MTS would be supported by load cells to monitor finger impact force during typing and the degree of resting hand support. During evaluation sessions, subjects would be outfitted with EMG (electromyogram) sensors on selected finger flexor, finger extensor, shoulder, and neck muscles to monitor the electrical activity which drives muscles.

During the first one hour evaluations session, subjects would be trying to type on the MTS for the first time. Muscle tension would likely be heightened

during this session while subjects were trying to adapt to surface typing. On following days subjects would return for 15-minute EMG evaluations at the beginning and end of 90-minute MTS typing practice sessions. To reduce the accumulation of high frequency EMG data, only the mean, standard deviation, and median EMG frequency computed over two-minute overlapping windows would be stored. Decreases in median EMG frequency are one possible measure of fatigue in fast twitch muscle fibers [144, 165]. Alternatively, near infrared spectroscopy [113] could measure changes in tissue oxygenation levels to infer fatigue. After a few days of such sessions, each subject would be given an MTS for use on the job at their personal workstations. They would return for two-hour evaluations at one-week intervals for a month to measure long-term adaption of muscle exertion during typing on the MTS.

Analysis of MTS load cell forces during successive evaluation sessions would indicate whether subjects became more comfortable resting hands on the MTS over time. Analysis of muscle exertion patterns versus hand resting patterns could establish the strength of the link between hand suspension and muscle exertion. This could also establish whether finger extensor exertion is minimized with resting palms or floating palms. Typing activity patterns observed in subjects with especially low exertion could suggest ways for all MTS operators to avoid fatigue from floating hands. Comparison of EMG's between the first and last session would indicate how much extra tension people will experience during initial adaption to the MTS and how quickly, if ever, such tension drops to negligible levels.

6.3 Future Directions for MTS Development

6.3.1 Increased Array Resolution

Higher resolution electrode arrays should permit segmentation of more specialized hand configurations such as pen grips or fists. The pen grip configuration (Figure 2.10), in which outer fingers curl under toward the palm so their knuckles

touch the surface, may offer better index finger control for handwriting and drawing applications than the "default" hand configuration (Figure 2.8), in which outer fingertips are normal to the surface. Preliminary experiments suggest that if the pinched fingers and outer knuckles are segmented properly, the existing finger identification system will correctly identify knuckles in a pen grip. Simple contact size and vertical separation measurements [162] can reliably distinguish the outer knuckles of the pen grip from the outer fingertips of the partially closed hand configuration (Figure 2.9). Knowing that pen grips are distinguishable from the hand configurations used in typing and chordic manipulation, one can envision a system in which the operator switches to handwriting mode with a hand configuration change just as easy as the configuration changes which switch between typing and pointing with the current MTS prototype. Preferably the index finger would be tracked as the inking stylus so the operator would not even have to stop and pick up an external stylus.

6.3.2 Handwriting Recognition

Integration of handwriting recognition would assuage certain deficiencies in typing and chordic manipulation on the MTS. Operators who have trouble typing on the MTS or who do not know how to touch type could rely on the handwriting mode for text entry. Good typists might rely on it only for entry of punctuation and symbols which are either hard to reach on the key layout without looking down or which do not exist at all on the key layout. Alternatively, handwriting mode could be used to invoke command macros by drawing particular alphabetic symbols as in Pen for OS/2 [142]. This could supplement the chordic manipulation gestures which can be performed more simply and quickly but lack the mnemonic associations possible with alphabetic symbols. Preferably, chordic manipulations would still invoke the most frequent commands such as cut, copy, and paste, while handwritten symbols would be mapped to a larger set of less-frequently-used commands.

6.3.3 Universal Access

People with handicaps such as loss of fine motor control in the fingers may have trouble performing the precise chordic manipulations described in Chapter 5. For these people the system could be modified to recognize grosser manipulations such as rotation and translation of a fist. Because the hand parts in a fist are closer together than in a chording hand, the fist would also require finer sensor arrays, at least in the vertical dimension, for reliable segmentation.

6.3.4 Fault Tolerant Segmentation

Segmentation algorithms which are more tolerant of individual sensor failures should also be investigated as finer sensor arrays become available. A single malfunctioning electrode sensor which produces random, zero, or full scale proximity measurements can easily disrupt the outward search for contact boundaries from local maxima (Section 3.2.5). This does not really matter on the relatively coarse sensor array of the current prototype because even if the segmentation search managed to continue past a bad electrode to find the correct contact boundary, a single bad electrode could still perturb the contact centroid so much as to make accurate pointing or typing impossible near the faulty electrode. However, as electrodes become smaller in finer sensor arrays, any one electrode will have a smaller influence on the fingertip centroid. At sensor array pitches of 1 mm $\times$ 1 mm, for example, failure of a single electrode should cause only barely noticeable biases in contact centroids. But with the current segmentation search pattern which tests only nearest neighbor pixels, a single bad electrode under a finger could cause many electrodes in the same or adjacent rows to be falsely excluded from the search, precipitating major disturbances in the fingertip centroid even for fine arrays.

6.3.5 Upgrading Operating Systems for High-DOF, Bimanual Manipulation

The chordic manipulations introduced in Chapter 5 are only the most basic examples of the rich bimanual and high-DOF interactions enabled by the MTS. Though researchers have been demonstrating intuitive bimanual applications such as tool glass menus [14], bezier curves [86, 147], and simultaneous pointing and scrolling [171] for over a decade, few widely available input devices and therefore few operating systems support them [61]. High-DOF devices such as drawing tablets are currently supported by specialized drivers for drawing or pen computing application.

If the MTS can be manufactured cheaply enough, the MTS has the potential to bring bimanual manipulation capability to a much wider population of computer users. To take full advantage of this capability, operating systems and graphical user interface frameworks should incorporate dual-stream manipulation messages into event queues at the same level where mouse and keyboard messages are currently processed. Current mouse messages in Windows 98, OS/2, XWindows, and Java support manipulation in only two dimensions and selection by only three or four mouse buttons. These message formats should be extended to at least six degrees of freedom of manipulation and at least five independent buttons, one for each finger on a hand. For maximum flexibility, a message format for absolute positioning should be optional as an alternative to velocity or relative positioning formats. Such enhancements could aid acceptance of all 3D navigational devices, not just the MTS, and should be based on generalized classifications of input devices such as Buxton's taxonomy [26]. These issues are partially addressed by drawing tablet protocols such as the XInput extensions [118] to XWindows. However, even the XInput format currently supports only 5DOF in a manner specific to drawing tablets.

The software enhancements necessary for dual stream input, *i.e.*, simultaneous two-handed manipulation, are more fundamental. Though most operating

systems allow more than one pointing device to be connected simultaneously, the motion commands of each device are merged into one stream so they all control the same cursor. While having a mouse cursor for each hand would be even more confusing than controlling the text cursor with the left hand and mouse with the right, research has shown [86] that it is useful and cognitively natural to pan, resize, and orient the background with the non-dominant hand while pointing with the dominant hand. People do this everyday without even noticing when they optimally orient a piece of paper with the left hand while writing on it with the right.

However, in current systems, the operator cannot scroll and move the pointer simultaneously yet independently because scrolling is usually activated as an auxiliary button drag in the single pointer input stream. For such operations to be supported simultaneously, the input stream from the non-dominant panning hand needs to be kept separate and possibly processed on a separate thread from the dominant pointing stream, *i.e.*, one thread would move and redraw the background in response to messages in the left hand stream while the other thread moves and redraws the pointer or other foreground object being manipulated. Programmers will have to deal with interesting synchronization issues such as arise when laying ink in a drawing application by panning the background with the left hand while simultaneously sliding the inking tool with the right hand.

6.4 Conclusion

This dissertation has demonstrated that, despite the limitations of proximity imaging, a large multi-touch surface can support a rich assortment of text entry, command entry, and graphical manipulation methods in a precise and non-fatiguing manner. Though the system occasionally misrecognizes the thumb or a palm or a directional slide, the frequency of these errors is already so small as to not frustrate the author under daily use. By phasing out parallelogram electrodes, increasing the frame rate, and continuing to tune the finger and hand identification algorithms,

those errors which can be blamed on faulty recognition of somewhat ambiguous hand configurations and motions will in all likelihood be eliminated entirely.

Nevertheless, some human errors due to performance of the wrong hand configuration or motion will always slip through. The price of integrating typing and 4-DOF chordic manipulations so seamlessly is increased demand for operator precision: finger touchdowns must be better synchronized and finger flexions more uniform than corresponding activities on stand-alone devices. The author's experiences operating the MTS indicate that these requirements for precision are easy for a skilled-operator to meet and therefore well within the range of human capability. The questions that remain are: how hard will it be for novices to master these requirements for relatively precise finger motion, and who will be willing to learn these chordic manipulations which can make interaction with computers so much more fluid? At the very least, the minimally-fatiguing nature of MTS activation offers new hope for people suffering from repetitive strain injuries to fingers, wrists and forearms. At best, the combination of extensive graphical manipulation capability and support for legacy touch typing skill will make the MTS practical enough to replace both mouse and keyboard in the personal computing tasks of the 21st Century.

Appendix A

ERGONOMICS FOR ENGINEERS

Full evaluation of any engineering design assumes an understanding of the physical limitations in each mechanism's materials. Human soft tissues are the most likely material to fail as a result of poor input device design and overuse. Therefore, the conditions under which soft tissue damage accumulates should be understood before comparing device designs.

A.1 Risk Factors for RSI

Various epidemiological studies of industrial jobs have identified the following risk factors for repetitive strain injuries [87]: repeated and sustained exertions, forceful exertions, localized mechanical stress, posture, joint kinematics, recovery time, and exposure to low temperatures. Some of these risk factors apply only marginally to computer use. For example, localized mechanical force is not likely to be encountered unless the user rests the forearms on a sharp table edge or taps too hard on a surface. Likewise, low temperature is less of a concern now that computers no longer need to be kept in frigid rooms. However, repetitive and forceful exertion, poor posture, and insufficient recovery time all contribute to input device overuse injuries such as tendonitis, tenosynovitis, epicondylitis, DeQuervain's syndrome, and carpal tunnel syndrome [37, 45, 117].

A common component of these injuries is inflammation and weakening of tendons, which connect muscle to bone, and of surrounding tissues. For example, tenosynovitis involves inflammation of the tendon sheath. Epicondylitis refers

to inflammation of the tissue where the finger flexor and extensor muscles connect to the elbow bone. The painful nerve damage of tunnel syndromes can occur when swelling of nearby tendons increases pressure on the nerves inside the tunnels through joints [120].

A.2 The Role of Force × Repetition in Soft Tissue Damage

The tendon tissue itself is a viscoelastic matrix of collagen fibers and filler cells. If too much tensile force is applied to the tendon or the tendon is stretched so often that the fibers don't have time to snap back, creep will occur, tearing some fibers and weakening the matrix [37, 45]. Unlike bones, which heal stronger than the original when broken, damaged tendons never fully regain their original strength because the replacement collagen fibers are more elastic but weaker and oriented less effectively than the originals.

Because repetition prevents proper mechanical and physiological recovery, the degree of tendon damage is actually related to the *product* of the applied force and frequency, rather than the sum [37]. For example, risk of injury in jobs which require both high force and high repetition can be nearly thirty times higher than the risk from high-force/low-repetition or low-force/high-repetition jobs [137, 138]. These data are thought to fit a monotonic increasing exposure-response relationship like the dose-response relationship for poisons, though the risk at intermediate activity levels has not been successfully measured [87]. Goldstein [53] hypothesizes that when the microtrauma from repetitive loads accumulates past a *critical cumulative trauma threshold*, long-term tissue damage and inflammatory processes set in. Resting allows the tissues to recover from microtrauma at an exponential rate, with partial recovery in seconds but full recovery taking days [84].

The remaining risk factors amplify the effects of repetition by further slowing recovery of tissue. Extreme joint postures compress tissues, thereby hindering blood circulation and increasing friction between the tendons and their sheaths [122]. For

example, extreme wrist flexion or extension increases the fluid pressure in the carpal canal. Sustained elevation of canal pressure is in turn considered both a sign and cause of carpal tunnel syndrome. Holding awkward postures may also require low-level static muscle contraction. While dynamic muscle contraction pumps blood in and out of the muscles, static contraction keeps blood out, eventually causing oxygen and nutrient depletion [54]. Strained tissue clearly cannot recover without ample oxygen and nutrients [113].

While the gradual nature of these injuries makes the recovery rate and critical cumulative trauma threshold very difficult to pinpoint, elevated physiological measures of fatigue have been observed at the moderate-force, high-repetition conditions typical of mouse and keyboard use. Johnson et al. [74] measured significant fatigue in the muscles which grip the sides of the mouse after 3 hours of mouse operation. Even though the average gripping force was only 0.6% of maximum voluntary contraction (MVC), the "low frequency fatigue" persisted up to 40 minutes after mouse usage stopped. Murthy et al. [113] found mean oxygenation of extensor carpi radialis brevis dropped to 89% of resting baseline after one minute of 5% MVC and to 81% after 10% MVC. Gerard et al. [44] measured 5-9% MVC in finger flexors and extensors during skilled typing on a standard keyboard, while exertion dropped to 3-6% MVC on a Kinesis ergonomic keyboard. In both keyboards exertion hovers around 5% MVC, a proposed threshold above which prolonged exposure may cause overuse injury. A study of carpal tunnel pressure during fingertip loading by Rempel et al. [125] indicates that the peak fingertip forces encountered during typing can temporarily elevate carpal tunnel pressure as much as extreme wrist posture does.

A.3 Activation Forces of Input Devices

The above analysis suggests that among conventional input devices, the sustained force of gripping the mouse, sustained postural loads from reaching too far for the mouse, and the repetitive clicking of stiff buttons can contribute to overuse

injuries. These concerns are addressed by capacitive touchpads which encourage neutral postures and extremely light tapping [24]. The splits in various ergonomic keyboards have successfully neutralized the awkward wrist postures of conventional keyboards [44, 65, 66, 114, 164], but less progress has been made in minimizing typing force.

Human factors studies have uncovered interesting relationships between typing force and keyswitch construction. Though make force, the force which must be applied before the key makes electrical contact, ranges between .3 and 1.0 N on most keyboards, users apply 2.5 to 7.9 times more force than necessary as a safety factor to ensure keys are always activated [43, 45, 126, 139]. The safety factor decreases with increasing make force, but overall applied force is still minimized with lower make forces [45, 126]. Other keyswitch parameters such as make point travel, the distance the key depresses before making electrical contact, and overtravel, the distance the key depresses after making electrical contact but before bottoming out, can also affect applied force. Radwin [123] found that peak applied force dropped 24% as overtravel increased from 0.0 to 3.0 mm, while pretravel had no significant effect. Gerard [45] found that typists applied roughly the same force to keyswitches with .72 N buckling springs, which produced an abrupt change in force feedback and an audible click, as they did to cheaper .28 N rubber dome keyswitches. Peak applied forces were 53% higher for rubber dome switches with .83 N make force. Finger flexor EMG's were independent of pace as long as typists worked at or below their comfortable pace, possibly because faster typists make more efficient movements [45].

Though these studies indicate which keyswitch designs will lessen typing force, the need to support resting hands and avoid accidental activation of keys ultimately limits the reduction of typing force [123] with mechanical keyswitches. Rose [127] suggests make forces less than .5 N may lead to accidental activation

and an increase in static extensor contraction for postural support, but .3 N seems tolerable on ergonomic keyboards such as the Kinesis which has firm wrist rests.

A.4 Relevance to the MTS

The approach taken in this dissertation has been to eliminate pressure switches altogether in favor of touchpad-like surface tapping and algorithmic discrimination between resting fingers and keypresses. There is no doubt subjectively that finger impact and finger flexion forces are greatly reduced when typing on the MTS. Future research should verify that this translates to reduction in pressure inside the carpal tunnel and avoidance of the critical cumulative trauma threshold. Also, future research should ensure that the MTS does not impose excessive postural demands since the hands are sometimes suspended above its surface.

Appendix B

VERTICAL INTERPOLATION BIASES ON PARALLELOGRAM ELECTRODE ARRAYS

Though the interspersed wedges of parallelogram electrode arrays introduced in Section 2.1.8 generally aid vertical interpolation when row spacings are larger than fingertip contact heights, they can cause misleading vertical measurements for small, light contacts not centered exactly on or between electrodes. The interpolated vertical position is always unbiased if contacts are centered on a row, but as Figure B.1 illustrates, under certain conditions it is biased when a contact is between rows. The actual positions of all contacts in Figure B.1 are halfway between the top (light gray) and bottom (dark gray) rows, and the interpolated positions of contacts a), c), d), and f) accurately reflect this because the contacts overlap equal portions of top and bottom electrodes. However, contact b) appears to overlap two to three times as much light gray as dark gray, which will cause a stronger signal on the top middle electrode and the interpolated position to be about 3/4 of the way to the top row. For an electrode row spacing of 1.2 cm this corresponds to a position error of 3-5 mm.

For the wider contact in Figure B.1e the contact overlaps of the middle and left columns are roughly equal, and only the right column is severely biased because the contact only touches the top electrode there. When the right column interpolation is averaged with the unbiased middle and left column interpolations the final reported position will be only 1-2 mm too high. For even larger contacts the

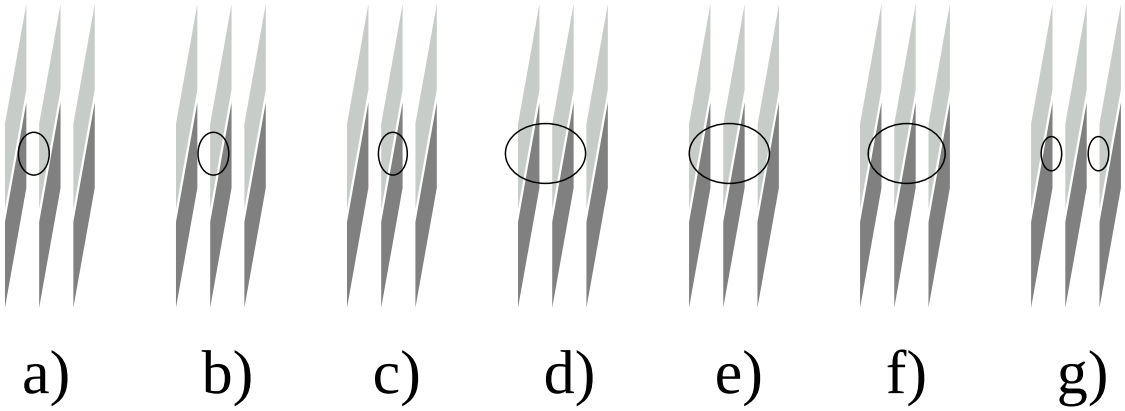


Figure B.1: Diagram illustrating the vertical interpolation biases which can arise when small-to-medium-sized contacts are halfway between parallelogram electrode rows but not centered on or between columns. In a) and d) small and medium contacts (ovals) are centered halfway between the left and middle columns and touch or overlap equal portions of top row electrodes (light gray) and bottom row electrodes (dark gray). In c) and f) the contacts are centered on the middle columns but still overlap equal portions of top and bottom rows, causing their vertical position to be correctly interpolated as halfway between the rows. But in b) and e) the contacts are just to the left of middle column center and overlap more of the top row electrodes than the bottom, causing their interpolated vertical position to be erroneously high. If they were just to the right of column center their interpolated vertical position would be erroneously low. The bias worsens for smaller contacts so that even though the tiny contacts in g) are centered between rows, the one on the left will be interpolated to be centered on the bottom row while the one on the right will be interpolated to be centered on the top row. Note that the parallelogram shadings in this figure are only meant to visually differentiate the top row electrodes from bottom row electrodes where the contacts overlap them, not to indicate the relative proximity measured in each row. The gaps between columns are necessary to route sensor wiring, but eliminating them would not eliminate the biases.

errors caused by biased overlap on peripheral columns become minuscule because the correct interpolations by columns toward the center of the contact dominate the centroid average.

Figure B.1g shows the worst case scenario, in which the position error can be as much as plus or minus half a row spacing. Here the contacts are so small that the left contact only overlaps the bottom left electrode, so its position will be reported as centered on the bottom row, a full 6 mm lower than the actual position, and the right contact will be reported a full 6 mm too high.

Note that the parallelograms cause no significant biases in horizontal centroid computation. This is easily proved by summing the proximity signals in each column and then interpolating horizontally using the column sums. As long as the vertical gaps between parallelograms are minuscule and the proximity sensors are linear, these computed column sums are equivalent to the proximity signal which would be obtained from a physically continuous rectangular electrode spanning the whole column, so they should produce exactly the same horizontal interpolation.

B.1 Nonlinear Vertical Centroid for Parallelogram Interpolation

The MTS employs a number of methods to ameliorate the effects of these vertical interpolation biases. First, sensed electrode capacitance is inversely proportional to flesh proximity, so using a substantially thick dielectric cover between the fingers and electrodes weakens the signal at the center of a contact where it is firmly touching the dielectric, preventing the center from totally dominating the signals from the contact periphery. For very small, light contacts such as in Figure B.1g, this causes the contact to appear wider or more diffuse to the sensors and exhibit more moderate biases between those of Figure B.1b and Figure B.1e. When a finger is moving horizontally across the surface, the biases manifest themselves as vertical oscillations in the finger trajectory. Therefore during medium-to-high speed motion they can be removed with a low-pass motion filter.

For medium-sized contacts as in Figure B.1e, a nonlinear centroid computation which de-emphasizes the potentially biased columns at the periphery of the contact in favor of those columns at the center which are overlapped evenly can further diminish the errors by at least a factor of two. Note that in the biased right column the contact only overlaps the left side of the column at the tip of the top electrode, so the total signal from the right column is much less than that from the left or middle columns. This is the most straightforward indication that the right column is on the contact periphery and may contribute a biased vertical interpolation.

Instead of linearly weighting the contribution of each column's vertical interpolation by each column's total proximity as in a standard centroid (Equation 3.11), the following nonlinear centroid formulas square each column's total proximity to emphasize the stronger, unbiased interpolations of the central columns. Let G_{E_j} be the electrodes in the intersection of group G and electrode column j, G_{z_j} be the total proximity of G_{E_j}, and G_{y_j} be the vertical centroid as measured within the electrodes of column j:

$$G_{E_j} = G_E \cap \{e \in j\} \tag{B.1}$$

$$G_{z_j} = \sum_{e \in G_{E_j}} e_z \tag{B.2}$$

$$G_{y_j} = \sum_{e \in G_{E_j}} \frac{e_z e_y}{G_{z_j}} \tag{B.3}$$

Then the improved vertical centroid G_{ypar} for parallelogram arrays is given by:

$$G_{zpar} = \sum_j G_{z_j}^2 \tag{B.4}$$

$$G_{ypar} = \sum_j \frac{G_{z_j}^2 G_{y_j}}{G_{zpar}} \tag{B.5}$$

Though these steps ameliorate the interpolation biases to less than a millimeter, ensuring that they are not a nuisance during evaluation of the MTS, they still

cause noticeable oscillations in on-screen cursor motion given high cursor motion sensitivity and light fingertip contacts. The only known way to eliminate the biases entirely is to decrease electrode row spacing so much that parallelograms are no longer needed.

Appendix C

CONVERGENCE TRAPS FOR LOCALIZED COMBINATORIAL SEARCH ON AN ATTRACTOR RING.

Figures C.1 and C.2 demonstrate unfortunate cases in which a pairwise exchange sequence like bubble sort can get trapped in assignments whose cost is only a local minimum. In Figure C.1, the attractors do not lie on a perfect circle, and a single contact (circle) is poorly initialized to an attractor x on the opposite side of the ring from its closest attractor c. For the contact's assignment to propagate around the ring to the closest attractor c, there must be a subsequence of the pairwise exchange sequence such that swapping in each pair of the subsequence decreases the contact's assignment distance. Moreover, this subsequence must connect the far attractor x to the contact's closest attractor c through overlapping attractor pairs.

In the case of Figure C.1a, where the attractors are arranged around a perfect circle, this convergence condition is met because the attractors a and d adjacent to the far attractor x are closer to the contact than x. This can be verified visually by noticing that they lie within the circle of points (heavy arc) the same distance from the contact as attractor x. It can also be verified by noticing that these attractors are on the same side of the relevant perpendicular bisector as the contact o, which will cause the contact to be swapped out of the far attractor and propagate toward the near attractor c.

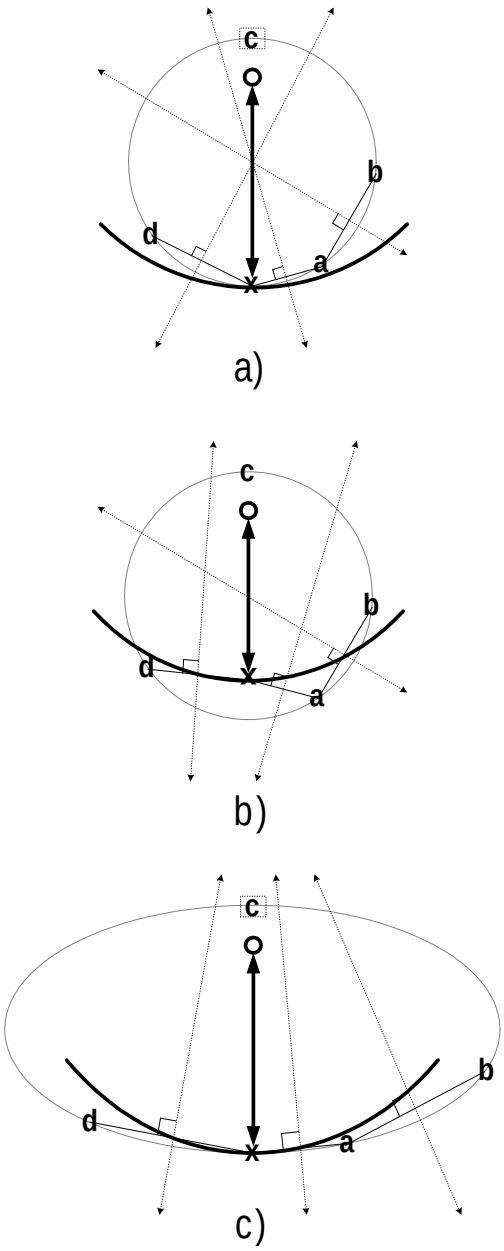


Figure C.1: Diagrams showing convergence failures for attractor rings which are not perfectly circular. In a) the attractor ring is perfectly circular so pairwise swapping of adjacent attractors can propagate a contact's (o) assignment from an attractor (x) on the opposite side of the ring to the closest attractor (c) on the near side. But the concavity in the ring of b) and the warping of the ring in c) trap the contact (o) in the far attractor (x) since attractors (a) and (d) adjacent to the far attractor are farther from the contact than (x).

The case of Figure C.1b contains a concavity in the attractor ring at x. This causes attractor x to be closer to the contact o than its adjacent attractors a and d. Thus pairwise swaps with these adjacent attractors will not be accepted and the contact cannot escape from attractor x. This is verified visually by noting that the adjacent attractors a and d lie outside the equidistant arc which passes through x or noting that attractor x is on the same side of the adjacent perpendicular bisectors as the contact o.

The elliptical attractor ring of Figure C.1c shows that a concavity is not necessary to prevent convergence. Again the attractors adjacent to the far attractor are outside of the equidistant arc and perpendicular bisectors, trapping the contact's assignment on the far attractor.

Since the finger attractor ring is not perfectly circular, convergence failures similar to those of Figure C.1b and Figure C.1c have actually been observed in the identification system. Typically a fingertip contact will get trapped in a palm attractor and misidentified as a palm. This has been addressed successfully by increasing the size of the exchange neighborhood for palm attractors to include all attractors instead of just the next attractor in the ring. Thus when the exchange sequence reaches a palm attractor, pairwise swaps are considered between the palm attractor and all other attractors, and the swap which produces the minimum assignment cost over all such pairs is taken. This increases the worst case number of comparisons to $O(M^3)$, the same as algorithms for general assignment problems.

Figure C.2 shows that for attractor rings which are perfectly circular, symmetric and full of contacts, pairwise exchange can get stuck in rotational local minima. The assignment of Figure C.2a clearly has the globally lowest total assignment cost, but the rotated assignments of Figure C.2b are also a local minimum. Under the distance-squared metric, the assignments of Figure C.2b all meet the local pairwise ordering relation. All pairwise swaps such as the swap on the bottom attractors

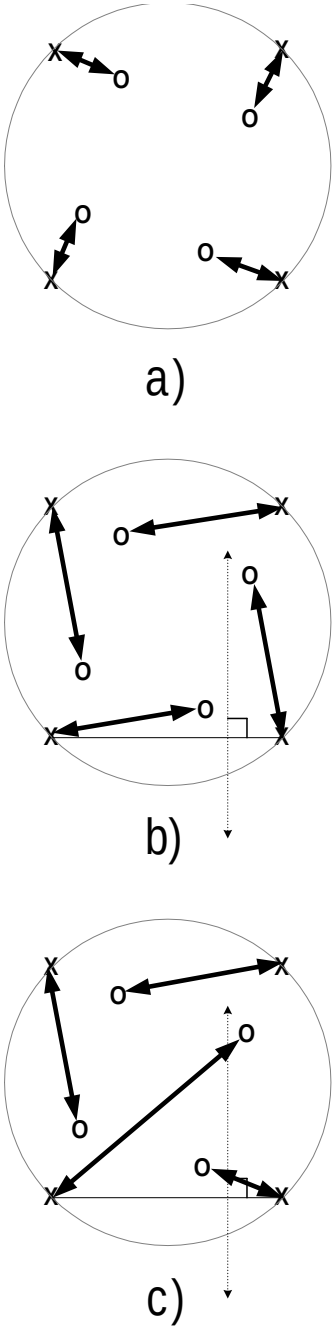


Figure C.2: Rotational local minimum for a perfectly symmetric attractor ring full of contacts. The global minimum assignment is shown in a). However, the rotated assignments in b) are also a stable local minimum because no pairwise swap such as that in c) will be accepted under the distance-squared metric.

in Figure C.2c would increase the assignment costs and will not be taken, trapping the assignments in this rotated state. However, this type of convergence failure has not been noticed in practice, presumably because the asymmetries in attractor spacings around the actual finger attractor ring discourage this type of convergence failure.

BIBLIOGRAPHY

[1] Emile Aarts and Jan Karel Lenstra. *Local Search In Combinational Optimization*. John Wiley and Sons, Chichester, 1997.

[2] Johnny Accot and Shumin Zhai. Beyond Fitts' Law: Models for trajectory-based hci tasks. In *CHI '97*, pages 295–302. ACM, March 1997.

[3] Subutai Ahmad. A usable real-time 3D hand tracker. In *Proceedings of the 28th Asilomar Conference on Signals, Systems, and Computers - Part 2 (of 2)*, volume 2, Pacific Grove, CA, October 1994. IEEE.

[4] Samuel Audet. Hot Scroll 1.0 for OS/2. http://www.cam.org/~guardia, 1998.

[5] Ravin Balakrishnan, Thomas Baudel, Gordon Kurtenbach, and George Fitzmaurice. The Rockin' Mouse: Integral 3D manipulation on a plane. In *Proceedings of CHI '97*, pages 311–318, Atlanta, GA USA, March 1997.

[6] Ravin Balakrishnan and I. Scott MacKenzie. Performance differences in the fingers, wrist, and forearm in computer output control. In *Proceedings of the CHI '97 Conference on Human Factors in Computing Systems*, pages 303–310, New York, 1997. ACM.

[7] Egon Balas, Donald Miller, Joseph Pekny, and Paolo Toth. A parallel shortest path algorithm for the assignment problem. Technical Report EDRC 05-38-89, Carnegie Mellon University, April 1989.

[8] M. Balinski. A competitive (dual) simplex method for the assignment problem. *Mathematical Programming*, 34:125–141, 1986.

[9] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[10] R. S. Barr, F. Glover, and D. Klingman. The alternating basis algorithm for assignment problems. *Mathematical Programming*, 13:1–13, 1977.

[11] D. P. Bertsekas. A new algorithm for the assignment problem. *Mathematical Programming*, 21:152–171, 1981.

[12] D. P. Bertsekas. A distributed asynchronous relaxation algorithm for the assignment problem. In *Proceedings of the 24th IEEE Conference on Decision and Control*, 1987.

[13] Paul J. Besl and Ramesh C. Jain. Range image segmentation. In Herbert Freeman, editor, *Machine Vision: Algorithms, Architectures, and Systems*, pages 221–256. Academic Press, Inc., New York, 1988.

[14] Eric A. Bier, Maureen C. Stone, Ken Fishkin, William Buxton, and Thomas Baudel. A taxonomy of see-through tools. In *Proceedings of CHI '94*, pages 358–364, New York, 1994.

[15] Sephen J. Bisset and Bernard Kasser. *Multiple Fingers Contact Sensing Method for Emulating Mouse Buttons and Mouse Operations on a Touch Sensor Pad.* U.S. Patent 5825352, October 20 1998.

[16] Klaus Boehm, Wolfgang Broll, and Michael Sokolewicz. Dynamic gesture recognition using neural networks; a fundament for advanced interaction construction. In *SPIE Conference on Electronic Imaging Science and Technology*, San Jose, California, February 1994.

[17] Robert A. Boie, Laurence W. Ruedisueli, and Eric R. Wagner. *Computer Mouse or Keyboard Input Device Utilizing Capacitive Sensors.* U.S. Patent 5463388, October 31 1995.

[18] J. E. Bolton and R. C. Wilkerson. Responsiveness of pain scales: a comparison of three pain intensity measures in chiropractic patients. *Journal of Manipulative and Physiological Therapeutics*, 21(1):1–7, January 1998.

[19] Grigore Burdea and Philippe Coiffet. *Virtual Reality Technology.* John Wiley and Sons, Inc., New York, 1994.

[20] Bill Buxton. Smoke and mirrors. *BYTE*, pages 205–210, July 1990.

[21] William Buxton. Chunking and phrasing and the design of human-computer dialogues. In *Information Processing 86*, pages 475–480, North Holland, 1986. Elsevier Science Publishers B.V.

[22] William Buxton, Eugene Flume, Ralph Hill, Alison Lee, and Carson Woo. Continuous hand-gesture driven input. In *Graphics Interface*, pages 191–195, Toronto, Ontario, 1983. Computer Systems Research Group, Department of Computer Science, University of Toronto.

[23] William Buxton and Brad A. Myers. A study in two-handed input. In *Proceedings of CHI '86*, pages 321–326, Toronto, Ontario, 1986.

[24] Ahmet E. Çakir, Gisela Çakir, Thomas Müller, and Pieter Unema. The Trackpad - a study on user comfort and performance. In *Proceedings of CHI '95*, http://www.acm.org/sigchi/chi95/Electronic/documnts/shortppr/aec1bdy.htm, 1995. ACM.

[25] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Erlbaum, Hillsdale, NJ, 1983.

[26] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. The design space of input devices. In M.M. Blattner and R.B. Dannenberg, editors, *Multimedia Interface Design*, pages 217–232. ACM Press, New York, 1992.

[27] G. Carpaneto and P. Toth. Primal-dual algorithms for the assignment problem. *Discrete Applied Mathematics*, 18:137–153, 1987.

[28] K. C. Chung, M. S. Pillsbury, M. R. Walters, and R. A. Hayward. Reliability and validity testing of the Michigan Hand Outcomes Questionnaire. *Journal of Hand Surgery*, 23(4):575–587, July 1998.

[29] Kinesis Corporation. Revolutionary ergonomic keyboards & high performance input accessories for PC, Macintosh, and Sun. http://www.kinesis-ergo.com/, 1999.

[30] J. L. Crowley and J. Coutaz. Vision for man machine interaction. In Leonard J. Bass and Claus Unger, editors, *Engineering for Human-Computer Interaction*, pages 28–43. Chapman and Hall, New York, 1995.

[31] U. Derigs. The shortest augmenting path method for solving assignment problems – motivation and computational experience. *Annals of Operations Research*, 4:57–102, 1985.

[32] Sarah A. Douglas and Anant Kartik Mithal. *The Ergonomics of Computer Pointing Devices*. Springer, London, 1997.

[33] J. Edmonds and R. M. Karp. Theoretical improvements on algorithmic efficiency of network flow problems. *Journal of the ACM*, 19:248–261, 1972.

[34] J. Egerváry. Matrixok kombinatorius tulajdonsagairol. *Mathematikai es Fizikai Lapok*, 38:16–26, 1931.

[35] Horst A. Eiselt, Giorgio Pederzoli, and Carl-Louis Sandblom. *Continuous Optimization Models*. Walter de Gruyter, New York, 1987.

[36] FakeSpace, Inc. Pinch Gloves. http://www.fakespace.com/prod-pnch.html.

[37] Donald L. Fisher, Robert O. Andres, David Airth, and Stephen S. Smith. Repetitive motion disorders: The design of optimal rate-rest profiles. *Human Factors*, 35(2):283–304, June 1993.

[38] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47:381–391, 1954.

[39] George W. Fitzmaurice and William Buxton. An empirical evaluation of graspable user interfaces: Towards specialized, space-multiplexed input. In *Proceedings of CHI '97*, pages 43–50, Toronto, Ontario, March 1997.

[40] Patrick J. Franz and David H. Straayer. *Integrated Keyboard and Pointing Device System with Automatic Mode Change*. U.S. Patent 5189403, February 1993.

[41] Masaaki Fukumoto and Yoshinobu Tonomura. Body coupled fingering: Wireless wearable keyboard. In *CHI 97*, pages 147–154, Atlanta, GA, March 1997. ACM.

[42] John M. Gauch. Image segmentation and analysis via multiscale gradient watershed hierarchies. *IEEE Transactions on Image Processing*, 8(1):69–79, January 1999.

[43] Michael J. Gerard, T. J. Armstrong, J.A. Foulke, and B. J. Martin. Effects of key stiffness on force and the development of fatigue while typing. *American Industrial Hygiene Association Journal*, 57:849–854, 1996.

[44] Michael J. Gerard, Stephen K. Jones, Leo A. Smith, Robert E. Thomas, and Tai Wang. An ergonomic evaluation of the kinesis ergonomic computer keyboard. *Ergonomics*, 37(10):1661–1668, 1994.

[45] Michael James Gerard. *Effects of Keyswitch Stiffness, Typing Pace, and Auditory Feedback on Typing Force, Muscle Activity, and Subjective Discomfort.* Ph.D. dissertation, University of Michigan, 1997.

[46] George E. Gerpheide. *Electrical Charge Transfer Apparatus.* U.S. Patent 5349303, September 20 1994.

[47] George E. Gerpheide. *Methods and Apparatus for Data Input.* U.S. Patent 5305017, April 19 1994.

[48] George E. Gerpheide and Michael D. Layton. *Capacitance-Based Proximity with Interference Rejection Apparatus and Methods.* U.S. Patent 5565658, October 15 1994.

[49] Douglas J. Gillan, Kritina Holden, Susan Adam, Marianne Rudisill, and Laura Magee. How does Fitts' Law fit pointing and dragging? In *CHI '90 Proceedings*, pages 227–234. ACM, April 1990.

[50] David Gillespie, Timothy P. Allen, Robert J. Miller, and Federico Faggin. *Object Position Detector With Edge Motion Feature.* U.S. Patent 5543590, August 6 1996.

[51] David Gillespie, Timothy P. Allen, and Ralph Wolf. *Object Position Detector With Edge Motion Feature and Gesture Recognition.* U.S. Patent 5543591, August 6 1996.

[52] William G. Gillick and Clement C. Lam. *Roller Mouse for Implementing Scrolling in Windows Applications.* U.S. Patent 5530455, June 25 1996.

[53] S. A. Goldstein. *Biomechanical aspects of cumulative trauma to tendons and tendon sheaths.* Ph.D. dissertation, University of Michigan, 1981.

[54] Etienne Grandjean. *Ergonomic Design of VDT Workstations.* Taylor and Francis, Philadelphia, 1987.

[55] Yves Guiard. Asymmetric division of labor in hamn skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 19(4):486–517, 1987.

[56] Handkey Corporation. Twiddler User's Manual. http://www.handykey.com/, 1997.

[57] Kostas Haris, Serafim N. Efstratiadis, Nicos Maglaveras, and Aggelos K. Katsaggelos. Hybrid image segmentation using watersheds and fast region merging. *IEEE Transactions on Image Processing*, 7(12):1684–1699, December 1998.

[58] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research.* Holden-Day, Inc., Oakland, California, 1986.

[59] W. Daniel Hillis. A high-resolution imaging touch sensor. *The International Journal of Robotics*, 1(2):33–44, 1982.

[60] Ken Hinckley. *Haptic Issues for Virtual Manipulation.* Ph.D. dissertation, University of Virginia, 1996.

[61] Ken Hinckley, Mary Czerwinski, and Mike Sinclair. Interaction and modeling techniques for desktop two-handed input. In *UIST '98 Symposium on User Interface Software and Technology*, pages 49–58, 1998.

[62] Ken Hinckley, Randy Pausch, Dennis Proffitt, James Patten, and Neal Kassell. Cooperative bimanual action. In *Proceedings of CHI '97*, pages 27–34, Atlanta, GA, March 1997.

[63] Ken Hinckley, Joe Tullio, Randy Pausch, Dennis Proffitt, and Neal Kassell. Usability analysis of 3D rotation techniques. In *Proceedings of UIST '97*, pages 1–10, Banff, Alberta, Canada, 1997.

[64] E. R. Hoffman. Fitts' Law with transmission delay. *Ergonomics*, 35:37–48, 1992.

[65] M. Honan, M. Jacobson, R. Tal, and D. Rempel. Changes in wrist postures during a prolonged typing task. In *Proceedings of the Human Factors and Ergonomics Society - 40th Annual Meeting*, pages 629–631, 1996.

[66] M. Honan, E. Serina, R. Tal, and D. Rempel. Wrist postures while typing on a standard and split keyboard. In *Proceedings of the Human Factors and Ergonomics Society - 39th Annual Meeting*, volume 1, pages 366–368, 1995.

[67] Don Hopkins. The design and implementation of pie menus. *Dr. Dobb's Journal*, December 1991.

[68] S. L. Horowitz and T. Pavlidis. A graph-theoretic approach to picture processing. *Computer Graphics and Image Processing*, 7:282–291, 1978.

[69] Infogrip, Inc. The BAT Personal Keyboard. http://www.infogrip.com/bat.htm, 1993.

[70] S. Inokuchi and R. Nevatia. Boundary detection in range pictures. In *Proceedings of the 5th International Conference Pattern Recognition*, pages 1301–1303, Miami, Florida, November 1980.

[71] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and M. Preston Mullen Jr. Integrality and separability of input devices. *ACM Transactions on Computer-Human Interaction*, 1:3–26, March 1994.

[72] R. A. Jarvis and Edward A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, C-22(11):1025–1034, November 1973.

[73] Herbert D. Jellinek and Stuart K. Card. Powermice and user performance. In *CHI '90 Proceedings*, pages 213–220. ACM, April 1990.

[74] Peter W. Johnson and Steven Lehman David M. Rempel. Measuring muscle fatigue during computer mouse use. In *Proceedings of the 1996 18th Annual Conference of the IEEE Engineering in Medicine Biology Society*, volume 4, pages 1454–1455, 1996.

[75] R. Jonkers and T. Volgenant. Shortest augmenting path method for dense and sparse linear assignment problems. *Computing*, 38:325–390, 1987.

[76] Paul Kabbash and William Buxton. The 'Prince' Technique: Fitts' Law and selection using area cursors. In *CHI '95 Mosaic of Creativity*, pages 273–279. ACM, May 1995.

[77] Paul Kabbash, I. Scott MacKenzie, and William Buxton. Human performance using computer input devices in the preferred and non-preferred hands. In *Proceedings of the InterCHI '93*, pages 474–481, 1993.

[78] Bernard Kasser, Bernhard Joss, and Sephen J. Bisset. *Touch Sensing Method and Apparatus.* U.S. Patent 5790107, August 4 1998.

[79] Donald E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1973.

[80] D. König. *Theorie der endlichen und unendlichen Graphen.* Akademie-Verlag, Leipzig, 1936.

[81] Myron W. Krueger. *Artificial Reality II.* Addison-Wesley, New York, 1991.

[82] Valerie Kucharewski. Technology evaluation: One-handed typing devices. http://www.stanford.edu/~valya/hci.html, 1997.

[83] H. W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Quarterly*, 2:83–97, 1955.

[84] Shrawan Kumar. A conceptual model of overexertion, safety and risk of injury in occupational settings. *Human Factors*, 36(2):197–209, June 1994.

[85] Gordon Kurtenbach and William Buxton. The limits of expert performance using hierarchic marking menus. In *Proceedings of INTERCHI '93*, pages 482–487, April 1993.

[86] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. The design of a gui paradigm based on tablets, two-hands and transparency. In *Proceedings of CHI '97*, pages 35–42, Atlanta, GA, March 1997.

[87] Wendi Ann Latko. *Development and Evaluation of an Observational Method for Quantifying Exposure to Hand Activity and Other Physical Stressors in Manual Work*. Ph.D. dissertation, University of Michigan, 1997.

[88] Seonkyoo Lee. A Fast Multiple-Touch-Sensitive Input Device. Master's thesis, University of Toronto, 1984.

[89] SK. Lee, W. Buxton, and K.C. Smith. A multi-touch three dimensional touch-sensitive tablet. In *Proceedings of CHI '85*, Toronto, Ontario, April 1985. ACM.

[90] Andrea Leganchuk, Shumin Zhai, and William Buxton. Manual and cognitive benefits of two-handed input: An experimental study. *ACM Transactions on Computer Human Interaction*, http://www.dgp.utoronto.ca/people/andrea/bimanual.html, 1999. in press.

[91] Neil A. Levine. *Finger worn graphic interface device*. U.S. Patent 4954817, September 4 1990.

[92] James D. Logan. *System for Using A Touchpad Input Device for Cursor Control and Keyboard Emulation*. U.S. Patent 5666113, September 9 1997.

[93] I. Scott Mackenzie. A note on the information-theoretical basis for Fitts' Law. *Journal of Motor Behavior*, 21:323–330, 1989.

[94] I. Scott MacKenzie. Input devices and interaction techniques for advanced computing. In W. Barfield and T.A. Furness III, editors, *Virtual Environments and Advanced Interface Design*, pages 437–470. Oxford University Press, Oxford, UK, 1995.

[95] I. Scott MacKenzie and William Buxton. Extending Fitts' Law to two-dimensional tasks. In *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems*, pages 219–226, New York, 1992.

[96] I. Scott MacKenzie and William Buxton. A tool for the rapid evaluation of input devices using Fitts' Law models. *SIGCHI Bulletin*, 25(3):58–63, 1993.

[97] I. Scott MacKenzie, Abigail Sellen, and William Buxton. A comparison of input devices in elemental pointing and dragging tasks. In *Proceedings of the CHI '91 Conference on Human Factors*, pages 161–166, New York, 1991. ACM.

[98] I. Scott MacKenzie, R. William Soukoreff, and Chris Pal. A two-ball mouse affords three degrees of freedom. In *Extended Abstracts of the CHI '97 Conference on Human Factors in Computing Systems*, pages 303–304, New York, 1997.

[99] I. Scott MacKenzie and Colin Ware. Lag as a determinant of human performance in interactive systems. In *INTERCHI '93*, pages 488–493. ACM, April 1993.

[100] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. A semantic analysis of the design space of input devices. *Human-Computer Interaction*, 5:145–190, 1990.

[101] Lillian G. Malt. Keyboard design in the electronic era. In *PIRA Eurotype Forum*, London Hilton Hotel, Park Lane, London http://www.teleprint.com/keyboard/history.html, September 14–15 1977. Malt Applied Systems, UK, Printing Industry Research Association.

[102] S. Martello and P. Toth. Linear assignment problems. *Annals of Discrete Mathematics*, 31:259–282, 1987.

[103] Edgar Matias, I. Scott MacKenzie, and William Buxton. Half-qwerty: A one-handed keyboard facilitating skill transfer from qwerty. In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, pages 88–94, New York, 1994. ACM.

[104] Edgar Matias, I. Scott MacKenzie, and William Buxton. Half-QWERTY: Typing with one hand using your two-handed skills. In *Companion of the CHI '94 Conference on Human Factors in Computing Systems*, pages 51–52, New York, 1994. ACM.

[105] Edgar Matias, I. Scott MacKenzie, and William Buxton. One-handed touch-typing on a qwerty keyboard. *Human-Computer Interaction*, 11:1–27, 1996.

[106] Edgar Matias, I. Scott MacKenzie, and William Buxton. A wearable computer for use in microgravity space and other non-desktop environments. In *Companion of the CHI '96 Conference on Human Factors in Computing Systems*, pages 69–70, New York, 1996. ACM.

[107] Paul McAvinney. The Sensor Frame-a gesture-based device for the manipulation of graphic objects. Available from SensorFrame, Inc., Pittsburgh, Pa., December 1986.

[108] Paul McAvinney. Telltale gestures. *BYTE*, 15(7):237–240, July 1990.

[109] L. F. McGinnes. Implementation and testing of a primal-dual algorithm for the assignment problem. *Operations Research*, 31:277–291, 1983.

[110] Nimish Metha. A flexible human machine interface. Master's thesis, University of Toronto, 1982.

[111] Robert J. Miller, Stephen Bisset, Timothy P. Allen, and Gunter Steinbach. *Object Position and Proximity Detector.* U.S. Patent 5495077, February 27 1996.

[112] K.W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations.* Cambridge University Press, Great Britain, 1994.

[113] Gita Murthy, Norman J. Kahan, Alan R. Hargens, and David M. Rempel. Forearm muscle oxygenation decreases with low levels of voluntary contraction. *Journal of Orthopaedic Research*, 15(4):507–511, 1997.

[114] M. Nakaseko, E. Grandjean, W. Hunting, and R. Gierer. Studies on ergonomically designed alphanumeric keyboards. *Human Factors*, 27(2):175–187, 1985.

[115] Claudia Nölker and Helge Ritter. Detection of fingertips in human hand movement sequences. In Ipke Wachsmuth and Martin Fröhlich, editors, *Gesture and Sign Language in Human-Computer Interaction, Proceedings of the International Gesture Workshop*, pages 209–218. Springer, New York, 1997.

[116] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations - Concepts and Applications of Voronoi Diagrams.* John Wiley and Sons, Chichester, 1992.

[117] Emil Pascarelli and Deborah Quilter. *Repetitive Strain Injury: A Computer User's Guide.* John Wiley and Sons, Inc., New York, 1994.

[118] Mark Patrick and George Sachs. *X11 Input Extension Protocol Specification.* The X Consortium, ftp://ftp.x.org/pub/R6.4/xc/doc/hardcopy/Xi/proto_ps.gz, version 1.0 edition, 1992.

[119] P.C.D. Maltron Ltd. Keyboards for people with special needs. http://www.maltron.com, 1997.

[120] Marko M. Pecina, Jelena Krmpotic-Nemanic, and Andrew D. Markiewitz. *Tunnel Syndromes: Peripheral Nerve Compression Syndromes.* CRC Press, New York, 2nd edition, 1997.

[121] Francis K.H. Quek. Unencumbered gestural interaction. *IEEE Multimedia*, 3:36–47, Winter 1996.

[122] MD R. Werner, PhD T. J. Armstrong, MS C. Bir, and M.K. Aylard. Intracarpal canal pressures: The role of finger, hand, wrist and forearm position. *Clinical Biomechanics*, 12(1):44–51, 1997.

[123] Robert G. Radwin. Activation force and travel effects on overexertion in repetitive key tapping. *Human Factors*, 39(1):130–140, March 1997.

[124] Graham J. Reid, Cheryl A. Gilbert, and Patrick J. McGrath. The Pain Coping Questionnaire: preliminary validation. *Pain*, 76:83–96, 1998.

[125] David Rempel, Peter J. Keir, W. Paul Smutz, and Alan Hargens. Effects of static fingertip loading on carpal tunnel pressure. *Journal of Orthopaedic Research*, 15(3):422–426, 1997.

[126] David Rempel, Elaine Serina, and Edward Klinenberg. The effect of keyboard keyswitch make force on applied force and finger flexor muscle activity. *Ergonomics*, 40(8):800–808, 1997.

[127] M. J. Rose. Keyboard operating posture and activation force: Implications for muscle over-use. *Applied Ergonomics*, 2:198–203, 1991.

[128] David A. Rosenbaum. *Human Motor Control.* Academic Press, Inc., New York, 1991.

[129] Dean Rubine and Paul McAvinney. Programmable finger-tracking instrument controllers. *Computer Music Journal*, 14(1):26–41, 1990.

[130] Dean Harris Rubine. The automatic recognition of gestures. Master's thesis, Carnegie Mellon University, 1991.

[131] Joseph D. Rutledge and Ted Selker. Force-to-motion functions for pointing. In D. Diaper et al., editor, *Human-Computer Interaction - INTERACT '90*, pages 701–706, Yorktown N.Y., 1990. IBM T.J. Watson Research Center.

[132] Jacqueline L. Salmon. For students, a painful lesson on computers. *The Washington Post*, Sunday:A01, May 17 1998.

[133] Richard A. Schmidt. *Motor Control and Learning.* Human Kinetics Publishers, Inc., Champaign, Illinois, 1988.

[134] Elaine R. Serina, C. D. Mote Jr., and David Rempel. Force response of the fingertip pulp to repeated compression - effects of loading rate, loading angle and anthropometry. *J. Biomechanics*, 30(10):1035–1040, 1997.

[135] B. Shneiderman. An empirical comparison of pie vs. linear menus. In *Sparks of Innovation in Human-Computer Interaction*, pages 79–88. Ablex Publishers, Norwood, N.J., 1993.

[136] John L. Sibert and Mehmet Gokturk. A finger-mounted, direct pointing device for mobile computing. In *Proceedings of UIST '97 Banff*, pages 41–42, Alberta, Canada, 1997. ACM.

[137] B. A. Silverstein, L. J. Fine, and T. J. Armstrong. Hand, wrist cumulative trauma disorders in industry. *British Journal of Industrial Medicine*, 43:779–784, 1986.

[138] B. A. Silverstein, L. J. Fine, and T. J. Armstrong. Occupational factors and carpal tunnel syndrome. *American Journal of Industrial Medicine*, 11:343–358, 1987.

[139] Paula M. Sind. The effects of structual and overlay design parameters of membrane switches on the force exerted by users. In *Proceedings of the Human Factors Society - 34th Annual Meeting*, pages 380–384. Department of Industrial Engineering and Operations Research, 1990.

[140] R. William Soukoreff and I. Scott MacKenzie. Generalized Fitts' Law Model Builder. In *Companion Proceedings of the CHI '95 Conference on Human Factors in Computing Systems*, pages 113–114, New York, 1995.

[141] David Joel Sturman. *Whole-Hand Input*. Ph.D. dissertation, Massachusetts Institute of Technology, 1992.

[142] BJ Sumner. *User's Guide to Pen for OS/2*. IBM Corporation, version 1.03 edition, 1995.

[143] Synaptics, Inc. Synaptics touchpad software drivers and utilities. http://www.synaptics.com, 1998.

[144] Esa-Pekka Takala, Sirna Lammi, Hannu Nieminin, and Eira Viikari-Juntura. Electromyographic changes in the static holding test of the arm. *International Journal of Industrial Ergonomics*, 12(1–2):85–90, 1993.

[145] Mark A. Tapia and Gordon Kurtenbach. Some design refinements and principles on the appearance and behavior of marking menus. In *Proceedings of UIST '95*, pages 189–195, Pittsburgh, PA. USA, November 1995.

[146] Jochen Triesch and Christoph von der Malsburg. Robust classification of hand postures against complex backgrounds. *Proceedings of the 1996 2nd International Conference on Automatic Face and Gesture Recognition*, pages 170–175, October 14–16 1996.

[147] John Viega, Matthew Conway, George Williams, and Randy Pausch. 3D Magic Lenses. In *Proceedings of UIST '96*, pages 51–58, Seattle, Washington, 1996.

[148] Inc. VPL Research. Data Glove Model 2 User's Manual. Redwood City, CA, 1987.

[149] Wacom Technology Co. Intuos: the first intelligent graphics tablet system. http://www.wacom.com/productinfo/intuos.html, 1998.

[150] CH. Wagner. The pianist's hand: anthropometry and biomechanics. *Ergonomics*, 31(1):97–129, 1988.

[151] Demin Wang. A multiscale gradient algorithm for image segmentation using watersheds. *Pattern Recognition*, 30(12):2043–2052, 1997.

[152] Colin Ware. Using hand position for virtual object placement. *The Visual Computer*, 6:245–253, 1990.

[153] Colin Ware and Danny R. Jessome. Using the Bat: A six-dimensional mouse for object placement. *IEEE Computer Graphics and Applications*, pages 65–70, November 1988.

[154] Pierre Wellner. The DigitalDesk Calculator: Tactile manipulation on a desk top display. In *Proceedings of UIST '91*, pages 27–33. ACM, November 1991.

[155] Pierre Wellner. Interacting with paper on the digitaldesk. *Communications of the ACM*, pages 1–17, July 1993.

[156] Pierre D. Wellner. Adaptive thresholding for the digitaldesk. Technical Report EPC-93-110, Rank Xerox EuroPARC, 1993.

[157] Pierre D. Wellner. Self calibration for the digitaldesk. Technical Report EPC-93-109, Rank Xerox EuroPARC, 1993.

[158] Wayne Westerman. Design and evaluation of a touch typing recognizer for a multi-touch surface. *in preparation*, 1999.

[159] Wayne Westerman and John Elias. *A method and apparatus for capacitive imaging of multiple finger contacts*. U. S. Patent Provisional Application Serial #60/072,509, January 26 1998.

[160] Wayne Westerman and John Elias. Graphical manipulation via finger chords on a multi-touch surface. *in preparation*, 1999.

[161] Wayne Westerman and John Elias. Hand tracking and finger identification on a multi-touch surface. *in preparation*, 1999.

[162] Wayne Westerman and John Elias. *A method and apparatus for integrated manual input.* U. S. Patent Application Serial #09/236,513, January 25 1999.

[163] Alan Daniel Wexelblat. A feature-based approach to continuous-gesture analysis. Master's thesis, University of Pennsylvania, 1994.

[164] Ph.D. William Hargreaves, M.D. David Rempel, Nachman (Manny) Halpern, M.D. Robert Markison, Dr. Ing. Karl Kroemer, and Jack Litewka. Toward a more humane keyboard. In *CHI '92*, pages 365–368, May 1992.

[165] Shaojun Xiao and Samy CS Leung. Muscle fatigue monitoring using wavelet decomposition of surface EMG. *Biomedical Sciences Instrumentation*, 34:147–152, 1998.

[166] Wayne Yacco. The BAT: Infogrip's keyboard good pick for alternative input device. *MacWEEK*, 6(27):71, July 27 1993.

[167] Shumin Zhai. *Human performance in six degree of freedom input control.* Ph.D. dissertation, University of Toronto, 1995.

[168] Shumin Zhai, Eser Kandogan, Barton A. Smith, and Ted Selker. In search of the "Magic Carpet": Design and experimentation of a bimanual 3d navigation interface. *Journal of Visual Languages and Computing*, February 1999.

[169] Shumin Zhai, Paul Milgram, and Anu Rastogi. Anisotropic performance in six degree-of-freedom tracking: An evaluation of 3D display and control interfaces. *IEEE Transactions on Systems, Man, and Cybernetics – part A: Systems and Humans.*, 27(4):518–528, July 1997.

[170] Shumin Zhai, Barton A. Smith, and Ted Selker. Dual stream input for pointing and scrolling. In *Proceedings of CHI '97 Extended Abstracts*, 1997.

[171] Shumin Zhai, Barton A. Smith, and Ted Selker. Improving browsing performance: A study of four input devices for scrolling and pointing tasks. In *Proceedings of INTERACT '97: The Sixth IFIP Conference on Human-Computer Interaction*, pages 286–292, July 1997.

[172] S. W. Zucker. Region growing: childhood and adolescence. *Computer Graphics and Image Processing*, 5:382–399, 1976.