EXHIBIT 4.12

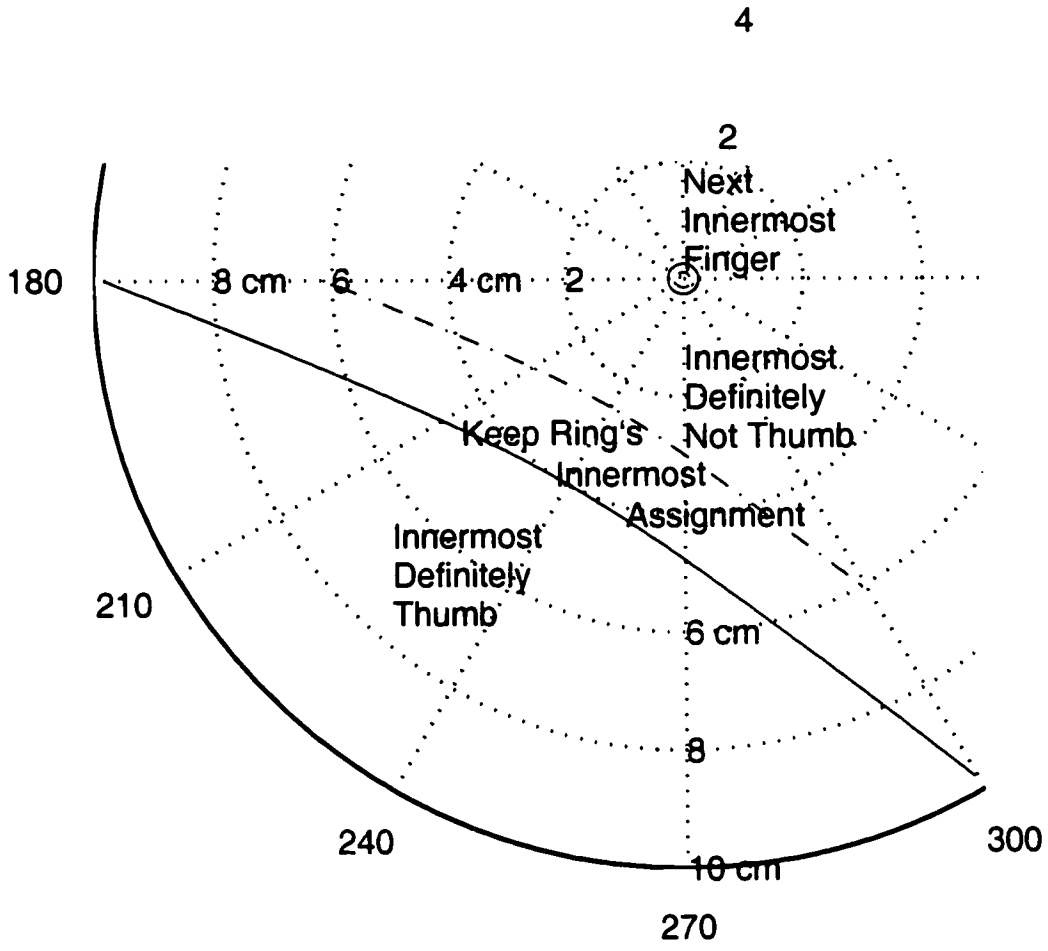

Figure 4.23: Thumb verification cutoffs for right hand versus inner separation and angle when other inter-contact features are not discriminating. This is a polar plot of the decision regions relative to the position of the next innermost finger contact (circle at center). If the innermost contact lies 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

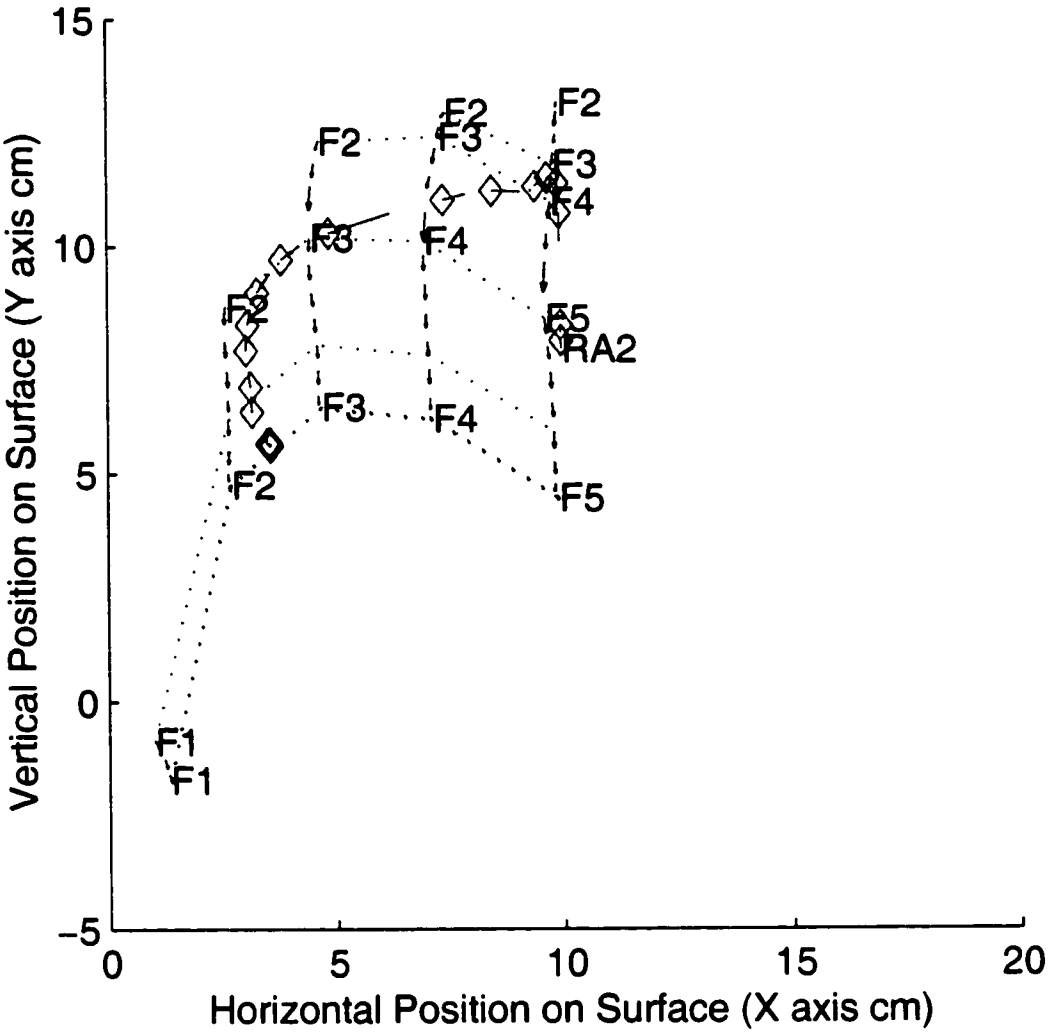

Figure 4.24: The hand starts at the top left with pinky touching down first and the other fingers rolling onto the surface at regular time intervals as the hand as a whole slides down. Since the original pinky touchdown occurs in the index finger Voronoi cell while there are no feature or inter-contact 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.

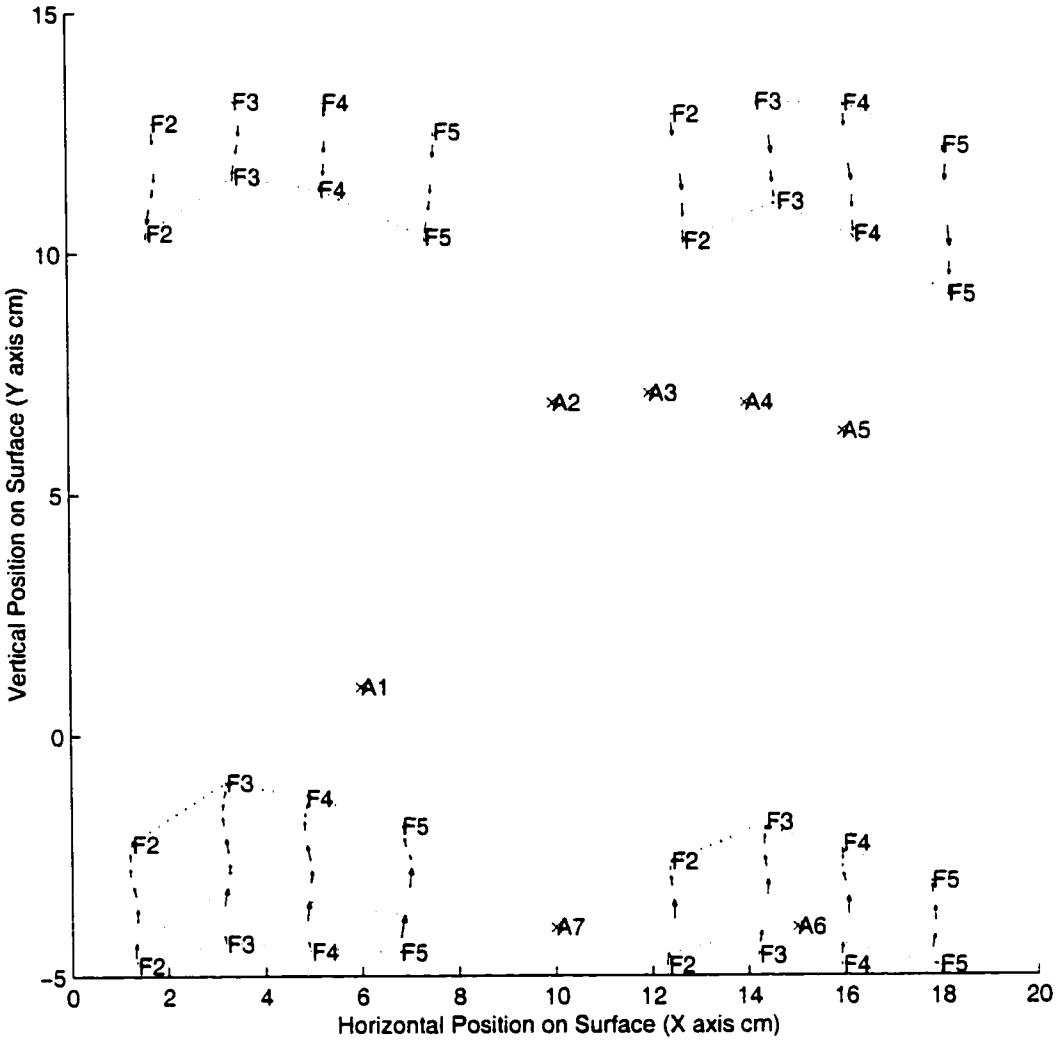

Figure 4.25: Row of four fingertips is placed successively at the four corners with attractors always starting at their defaults (red x's). Correct identification throughout every placement shows that four fingertips in a roughly horizontal row are sufficient for perfect, instantaneous identification 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.

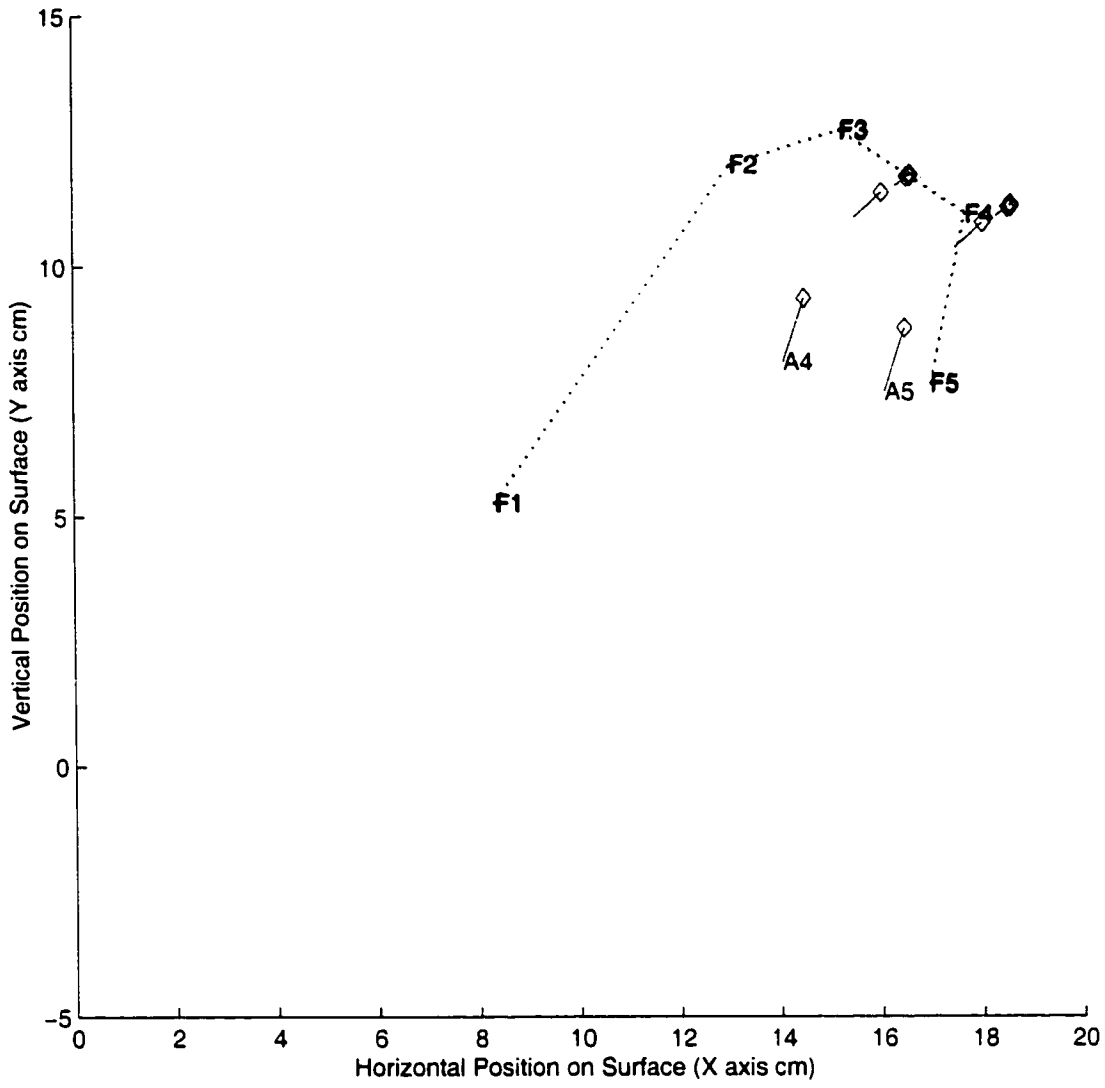

Figure 4.26: A claw hand with the pinky crossed under the ring finger verifies the finger rotation tolerances postulated for the attractor ring in Figure 4.11 on Page 160. The pinky contact (bottom right) is always identified correctly, regardless of attractor translation, since the angle between ring 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).

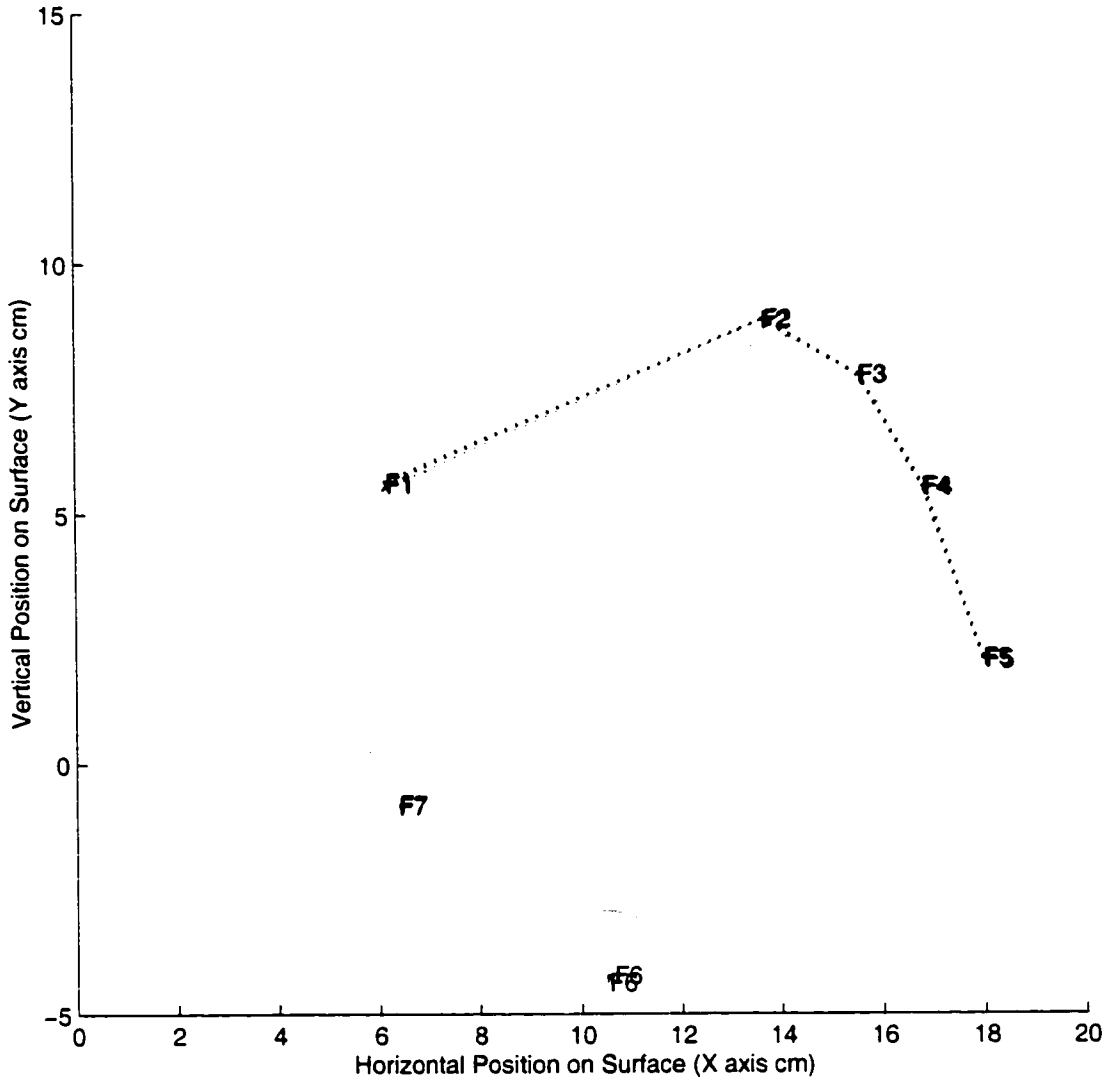

Figure 4.27: Fingers in a hand rotated fully clockwise to the limits of ulnar deviation at the wrist are always identified perfectly. Presence of the palm heels is not needed for this correct result. However, the fingertips are kept well spread to avoid parallelogram-electrode-induced segmentation merging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

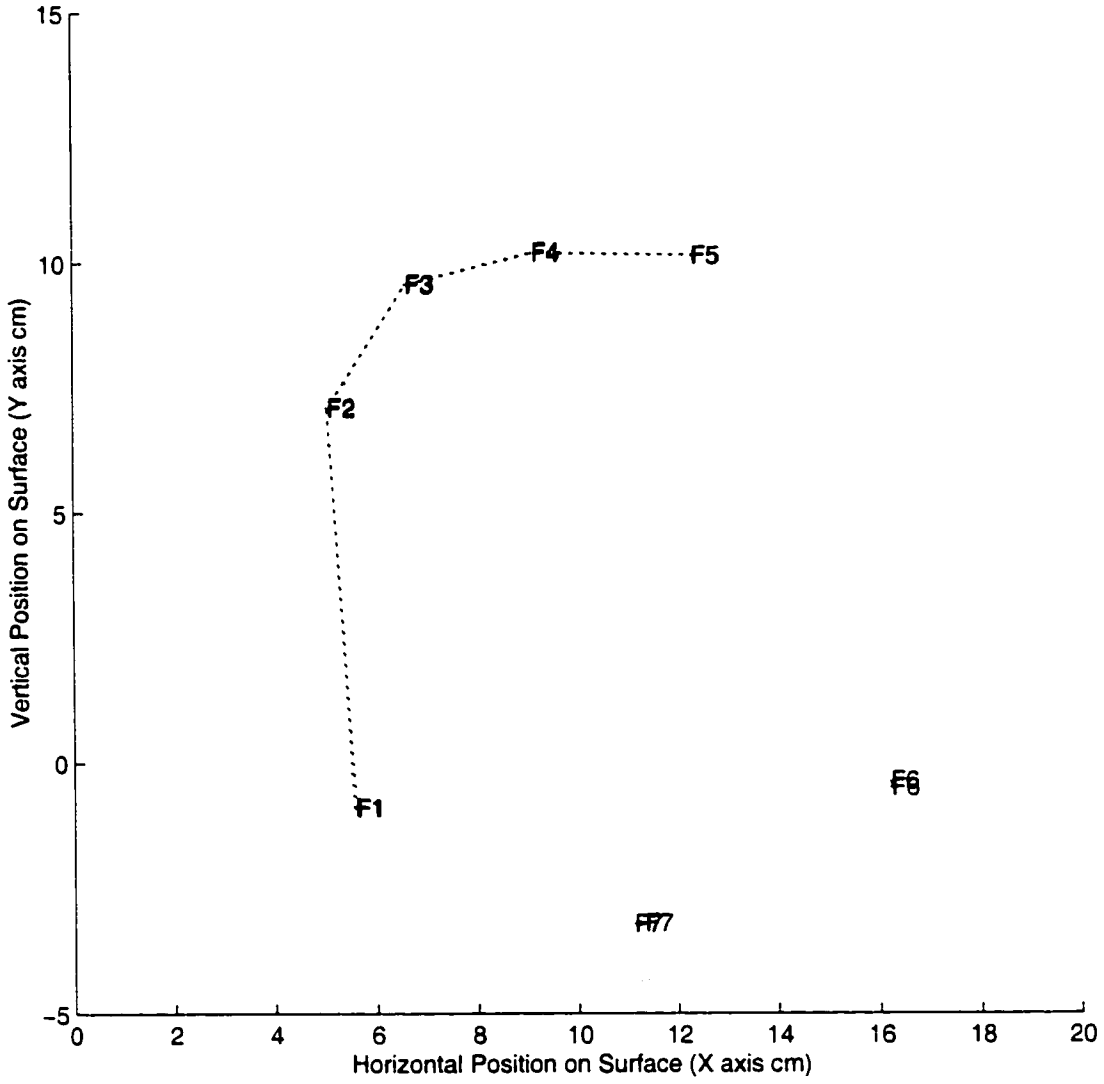

Figure 4.28: Fingers in a hand rotated fully counter-clockwise to the limits of radial deviation are identified correctly in this experiment, but not always. As will be seen in the next figure, absence or merging of both palm heels will open up the inner palm heel attractor (A7) which can then grab the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

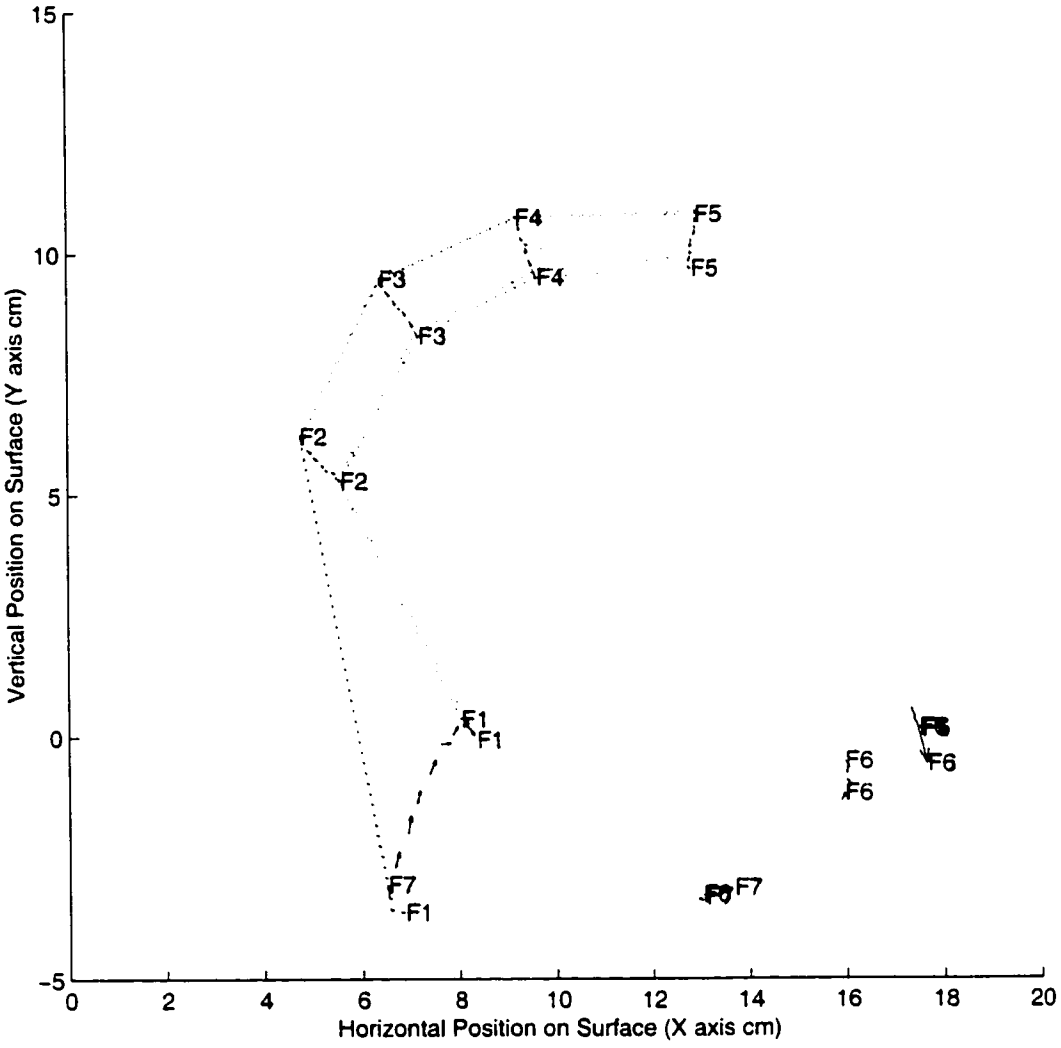

Figure 4.29: Absence or merging of both palm heels can cause thumb misidentification when right hand is rotated fully counter-clockwise. In this case, all identifications are initially correct, but as the palm heels press onto the surface they merge into one huge contact, leaving the inner palm heel 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.

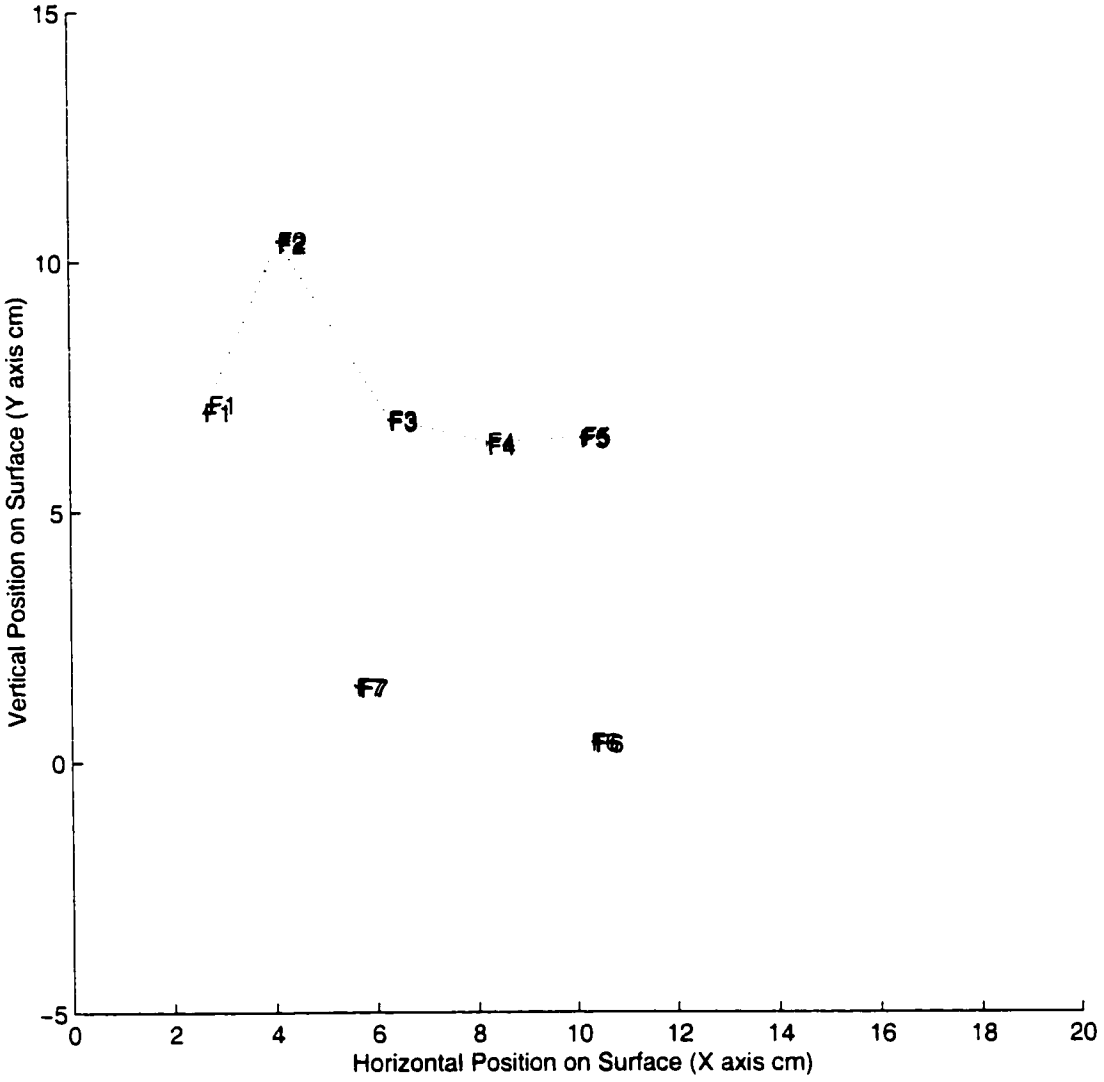

Figure 4.30: Fingers in a pen grip configuration are always identified correctly anywhere on the surface as long as they are segmented properly. In this experiment the fingers were kept a little looser than they would be in practice to prevent the thumb contact from merging with the index fingertip or 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.

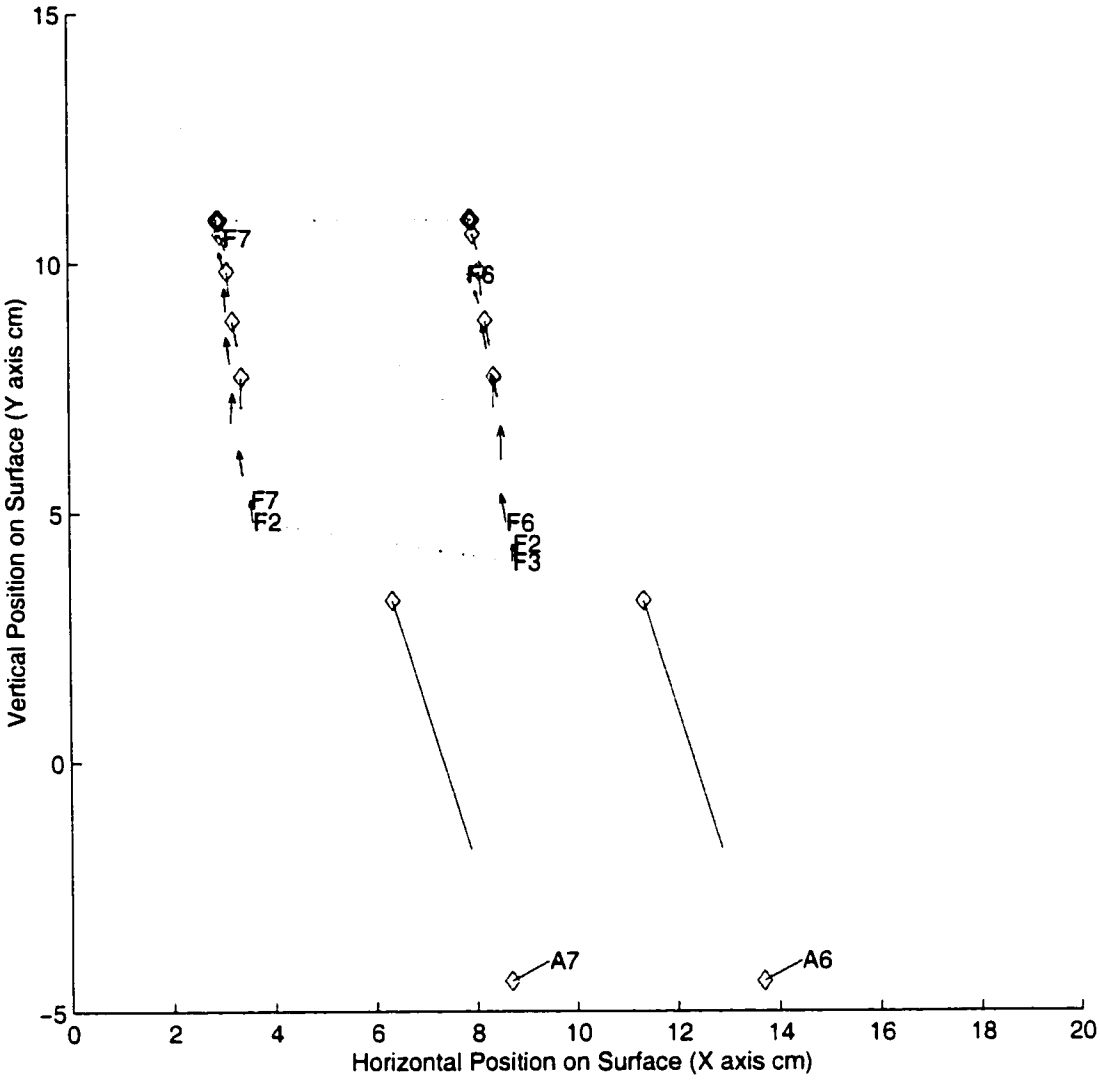

Figure 4.31: Palm heels alone are identified correctly anywhere on surface when they bottom out and reach full size. In this experiment the palms touch down gradually on the middle left and slide toward the top of the surface. Initially they are misidentified as fingertips and the palm attractors (A6,A7) 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.

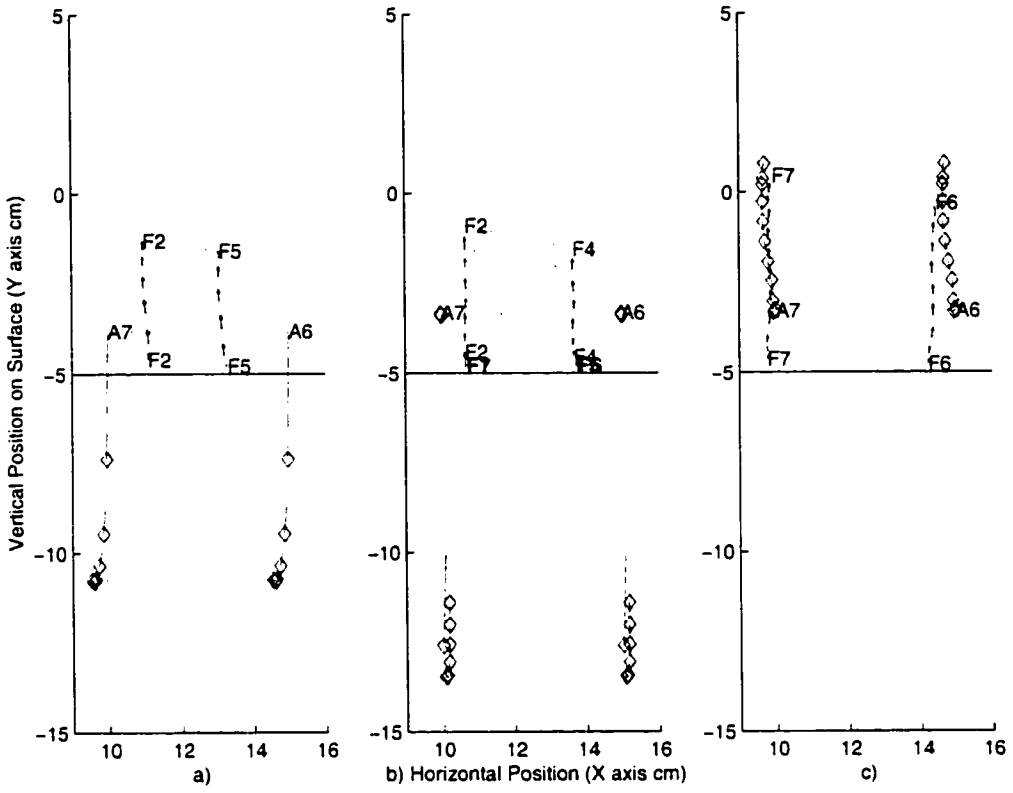

Figure 4.32: Dependency of fingertip pair identification in palm regions on fingertip separation. In each case, middle and ring fingers touch down and slide upward between the palm heel attractors near the bottom edge (horizontal black line) of the sensing area. In a) the adjacent fingertips are about 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.

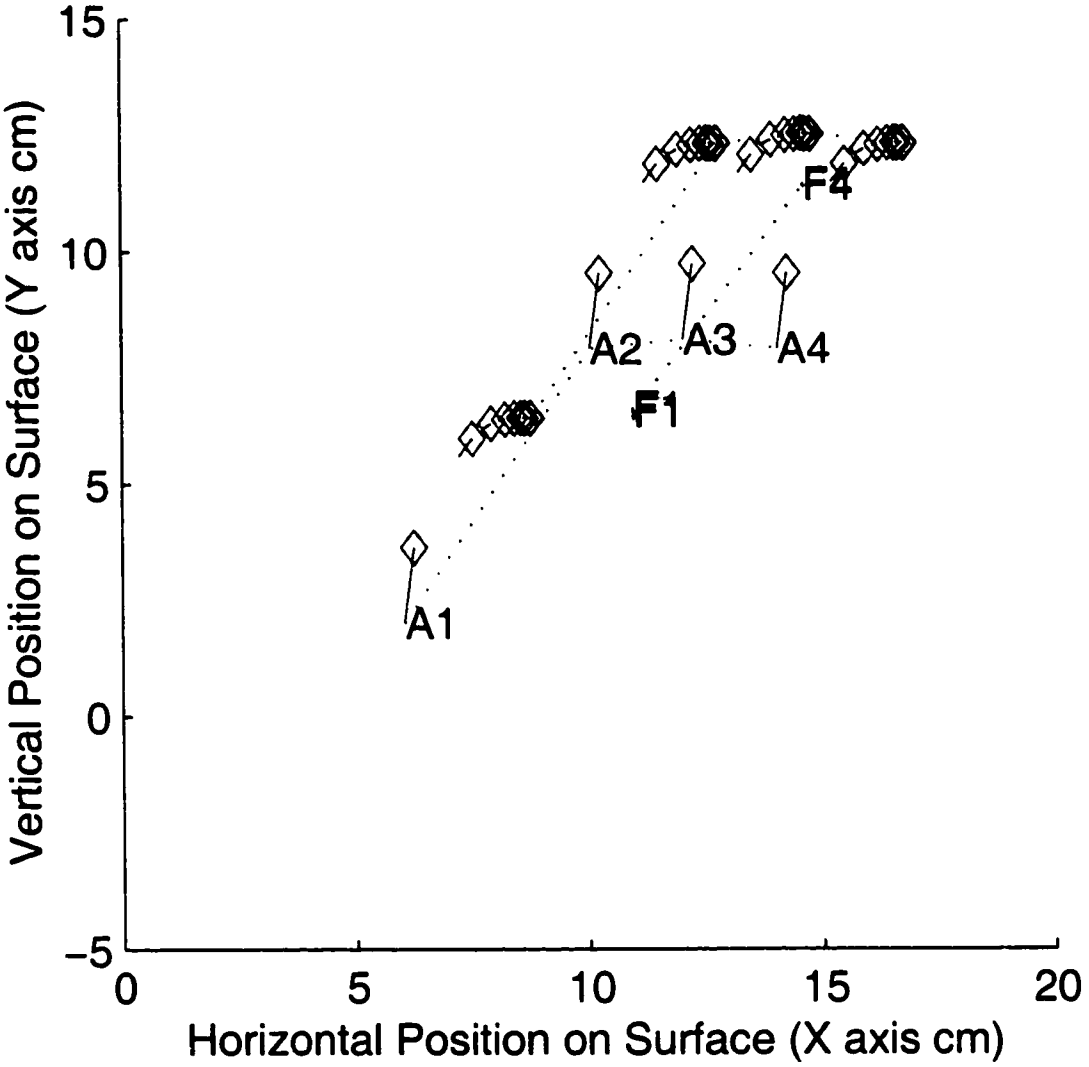

Figure 4.33: Correct thumb identification for a thumb-middle fingertip chord in the fingertip regions. Note how the thumb and fingertip attractors jump to the upper right to align with the finger contacts. The thumb verification module is able to distinguish a thumb and a fingertip from two fingertips 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.

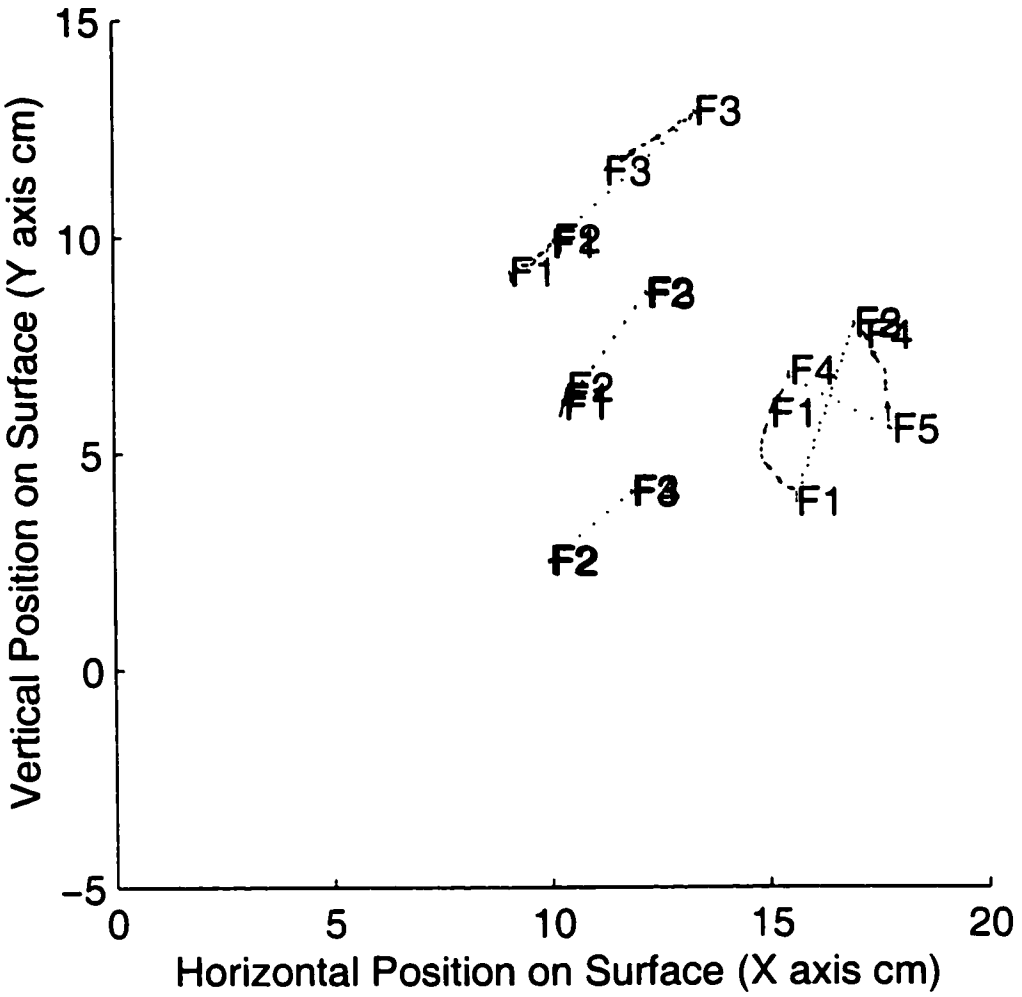

Figure 4.34: Identifications of a thumb and middle finger which do not start uniquely separated but perform unique motions. In a) at the top, the thumb is initially misidentified as F2, but the expansion factor (Section 4.4.7.3) quickly responds to the anti-pinch scaling motion between the thumb and 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

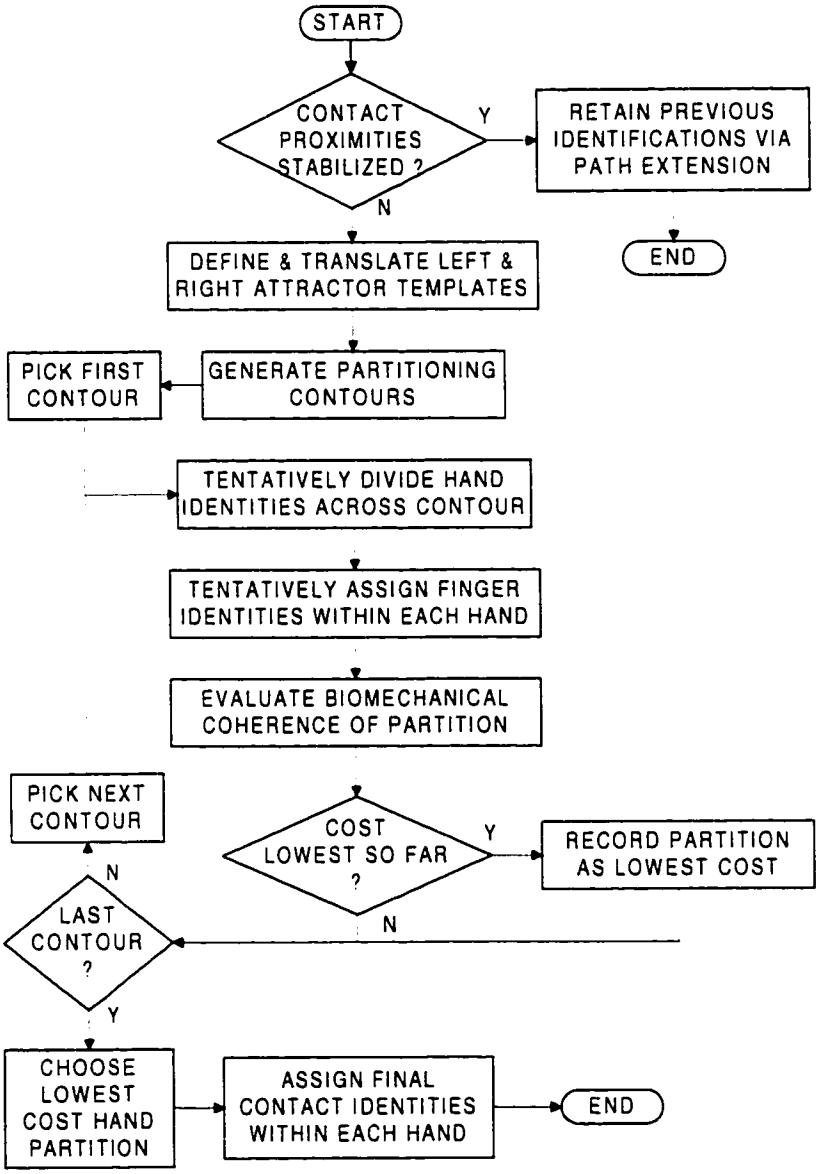

Figure 4.35: Flow chart of the hand identification algorithm.

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.

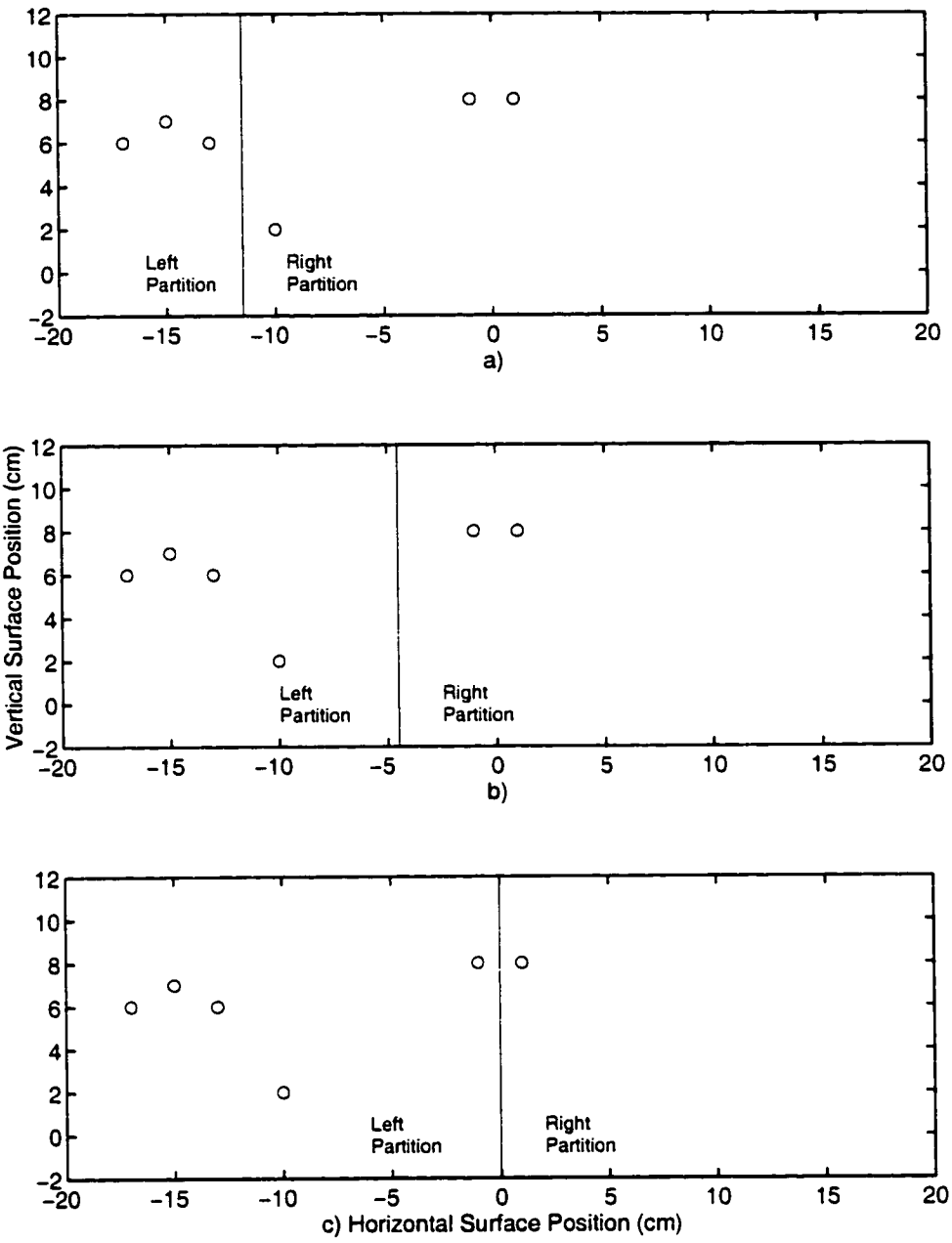

Figure 4.36: Vertical contours (dividing lines) creating three different partitioning hypotheses. Contours are always placed halfway between horizontally adjacent contacts (circles). The partitioning of b) is probably the correct partitioning for this arrangement of contacts, meaning that the two contacts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

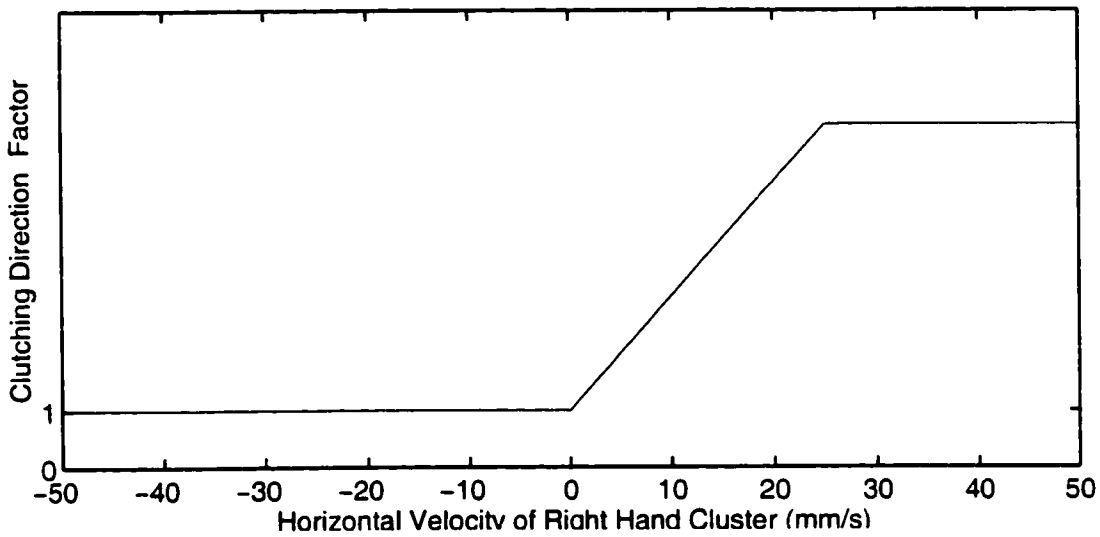

Figure 4.37: Hand clutching direction factor versus the average of the right hand's horizontal contact velocities.

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

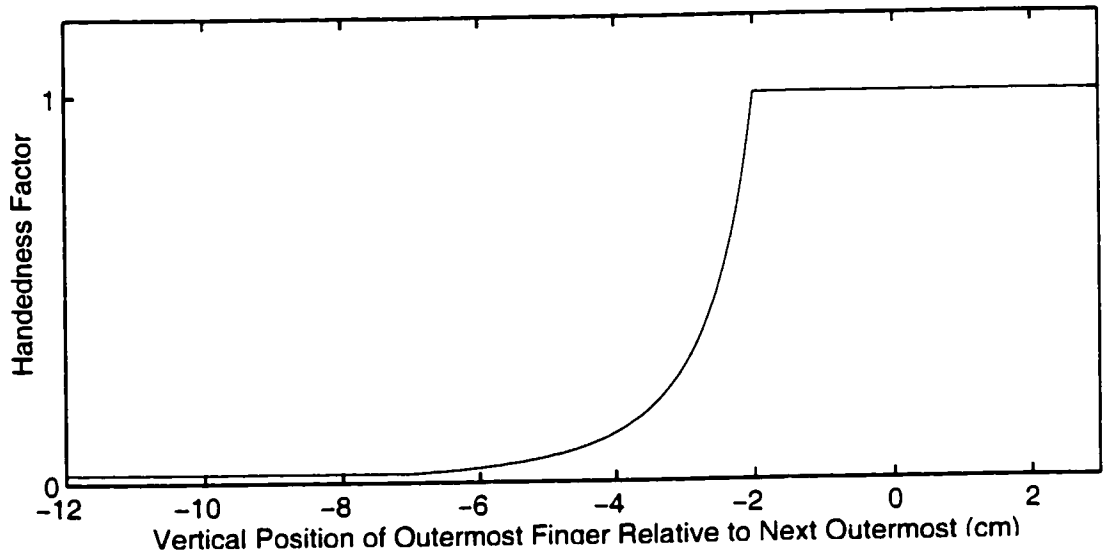

Figure 4.38: Handedness factor versus the vertical separation between outermost and next outermost finger contacts.

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.

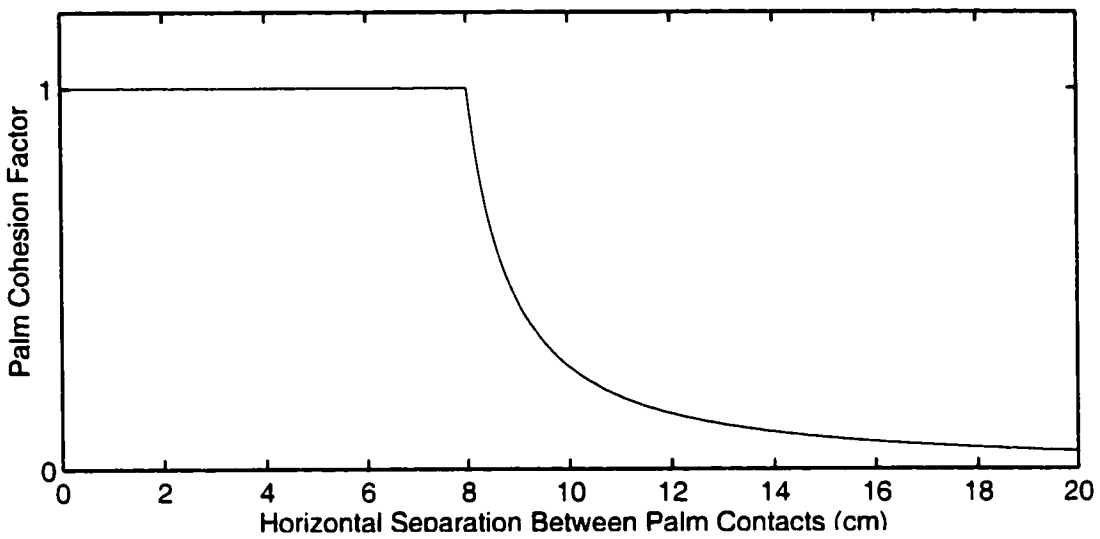

Figure 4.39: Palm cohesion factor versus the horizontal separation between the innermost and outermost contacts identified as palms.

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

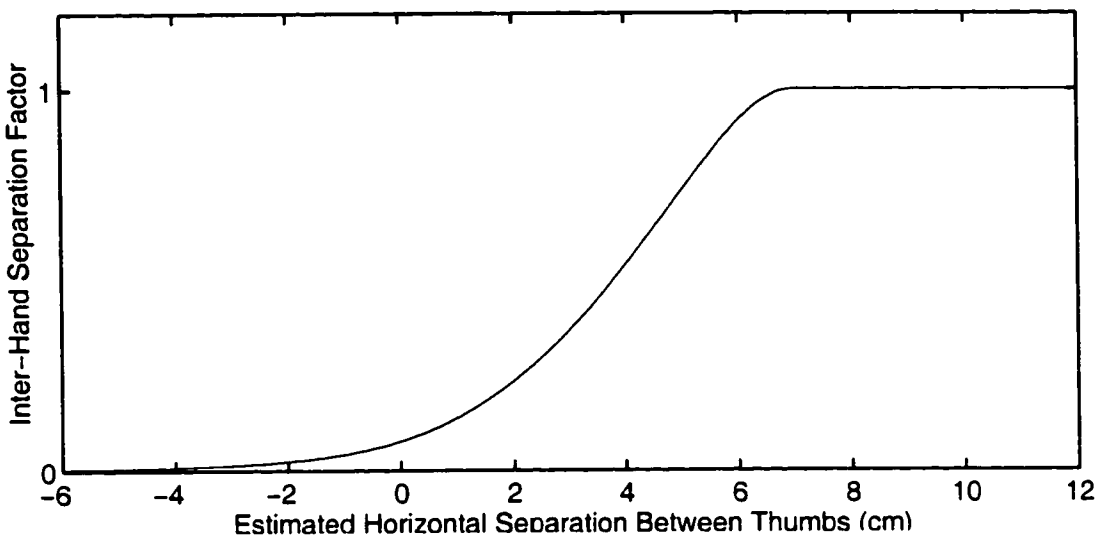

Figure 4.40: Inter-hand separation factor versus the estimated distance between the left and right thumbs.

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

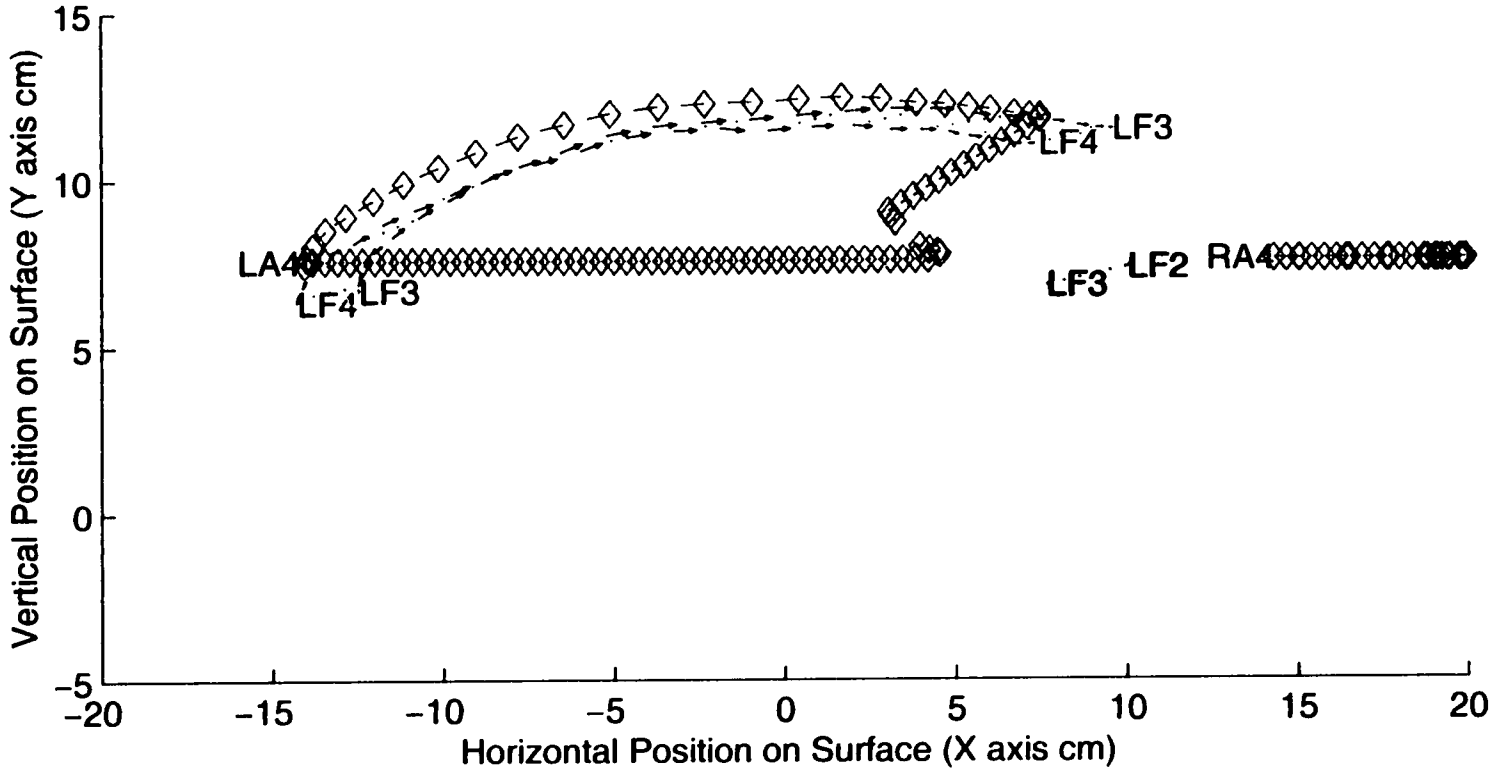

Figure 4.41: Short-term memory of hand identity as maintained by the hand position estimate. In this experiment, a left hand finger pair starts in left default position and slides well into the right side of the surface, then lifts off for a second and taps the surface a few cm lower. Because the left 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.

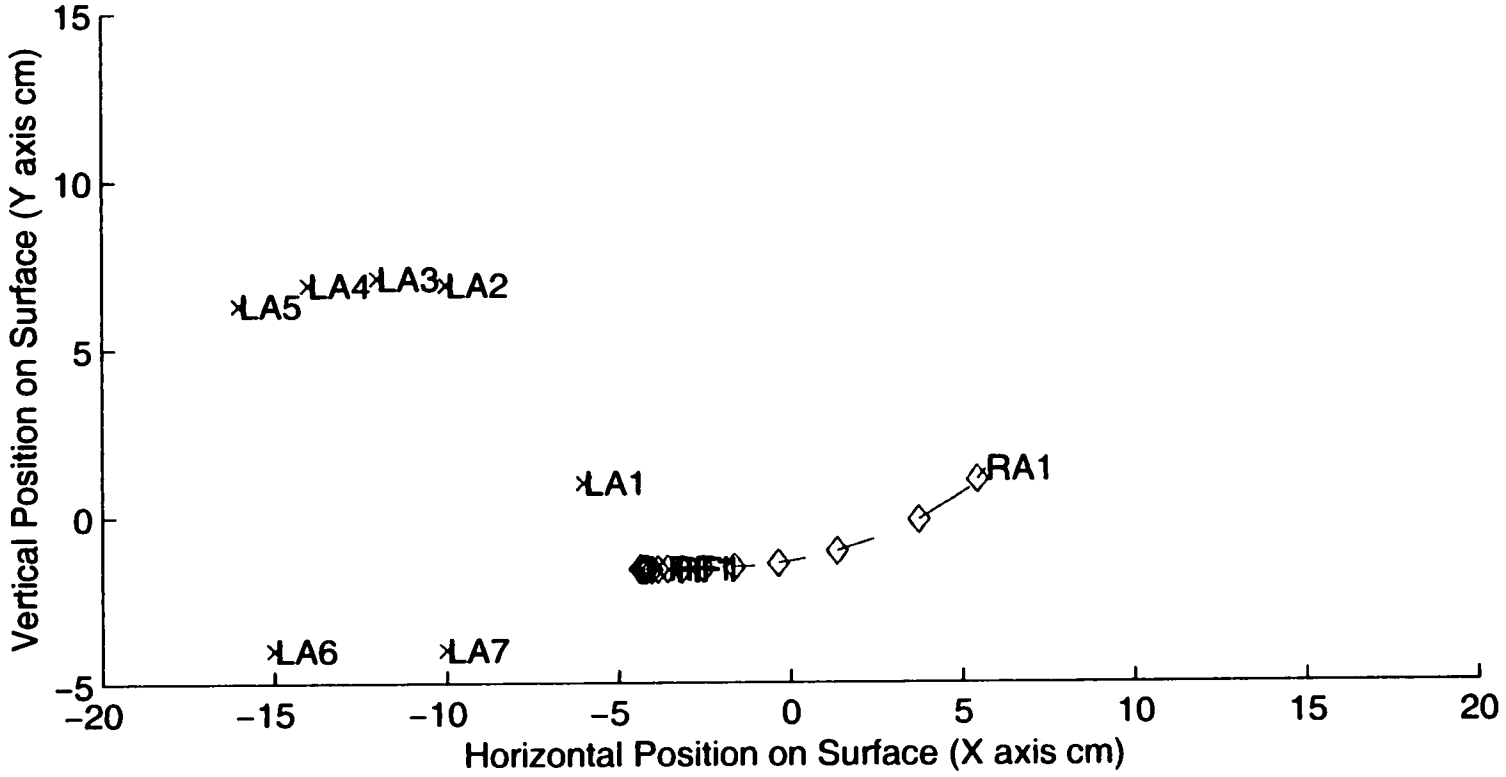

Figure 4.42: A right thumb (RF1) placed in the left middle of the surface is correctly identified solely by virtue of its orientation. Note that the contact ellipse's major angle is about 120°. Even though this right thumb is placed quite near the left thumb attractor (LA1), the right thumb orientation 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.

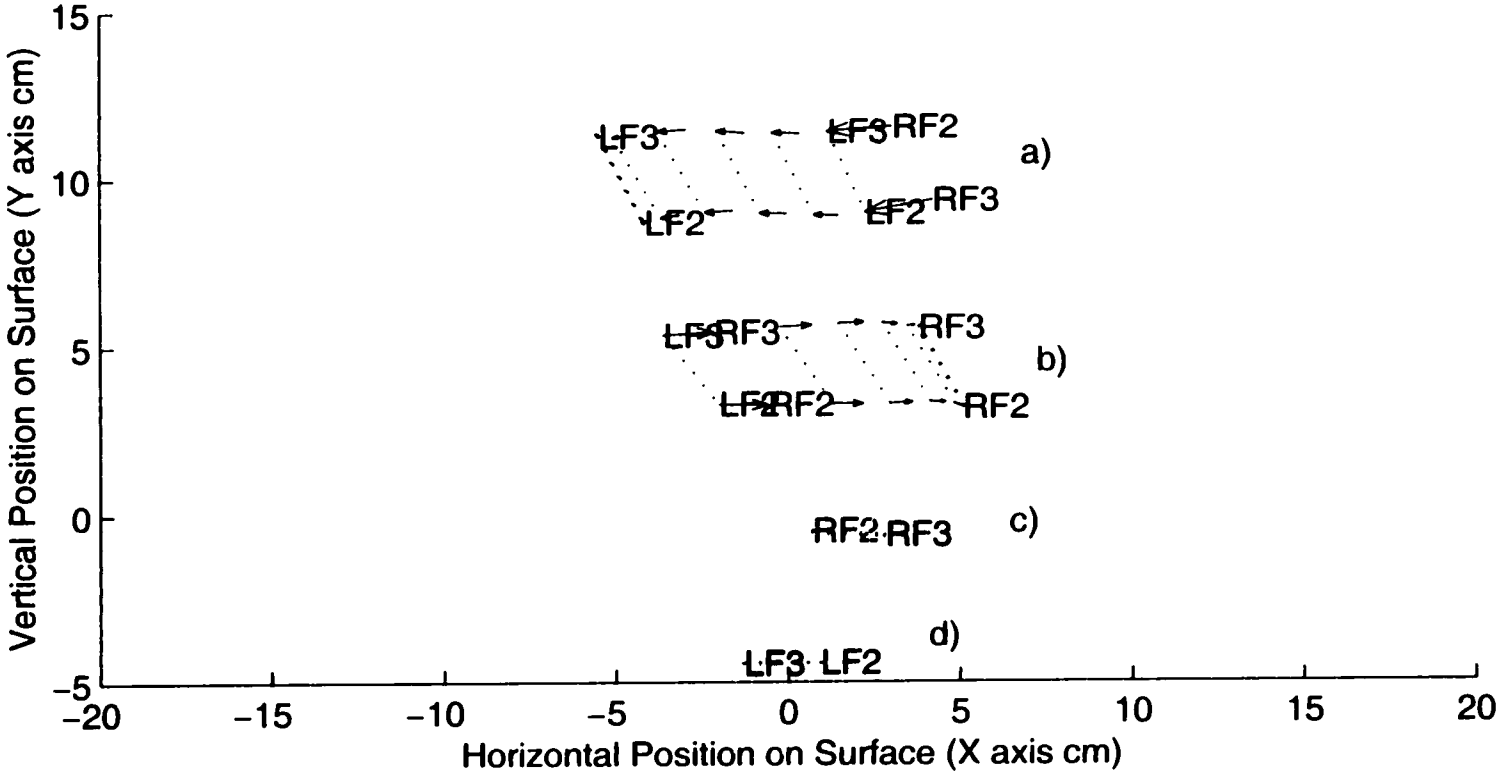

Figure 4.43: Effect of clutching velocity factor on hand identification near middle of surface. For the stationary fingertip pairs in c) and d), both fingertips get the same hand identity but the pair identity depends on whether the pair touches down just to the left d) or right c) of board center. Without 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.

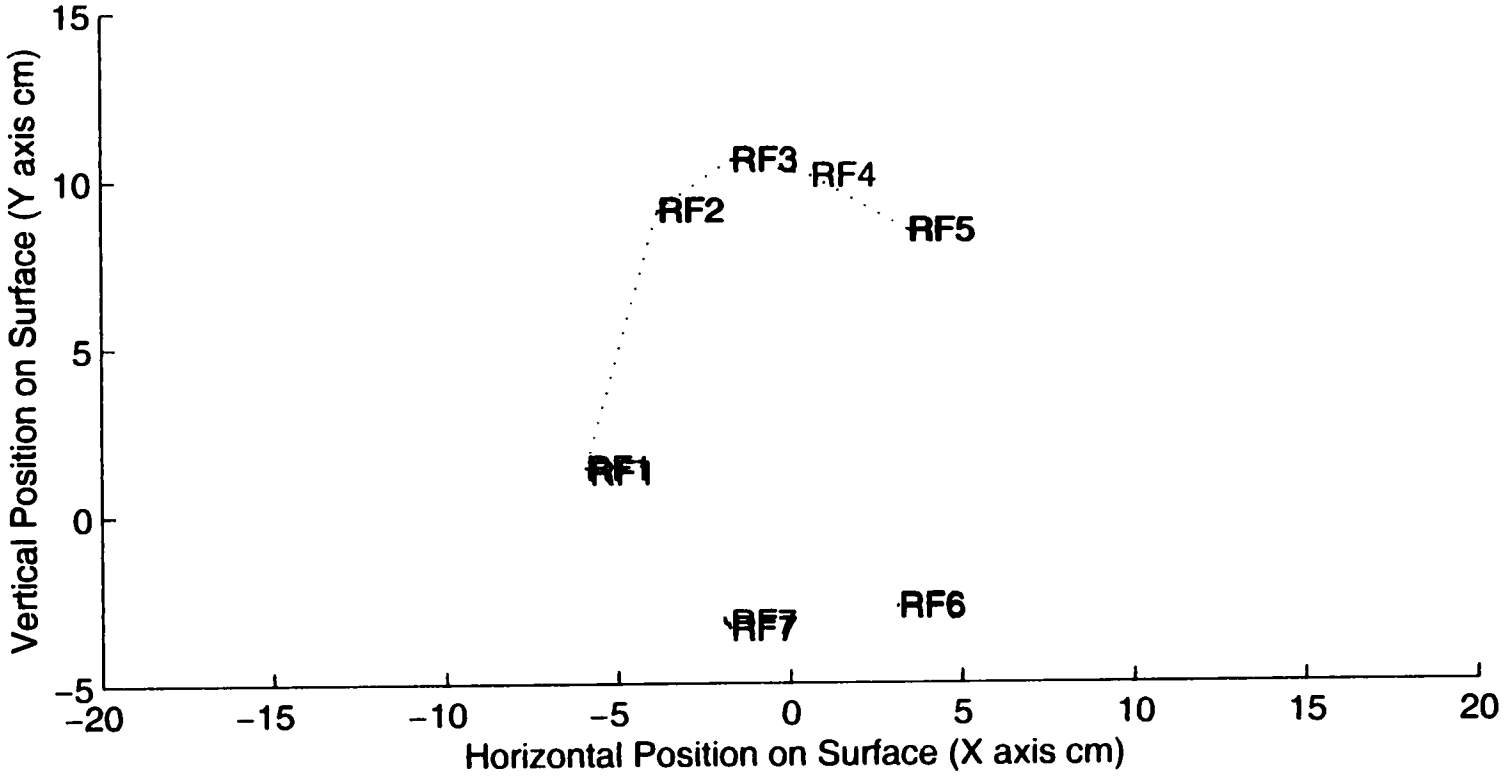

Figure 4.44: A right hand touches down straddling the middle of the surface. The inter-hand separation factor ensures all contacts are attributed to the same hand, and since the thumb is present, the thumb orientation and handedness factors ensure the contacts are correctly identified as a right hand 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.

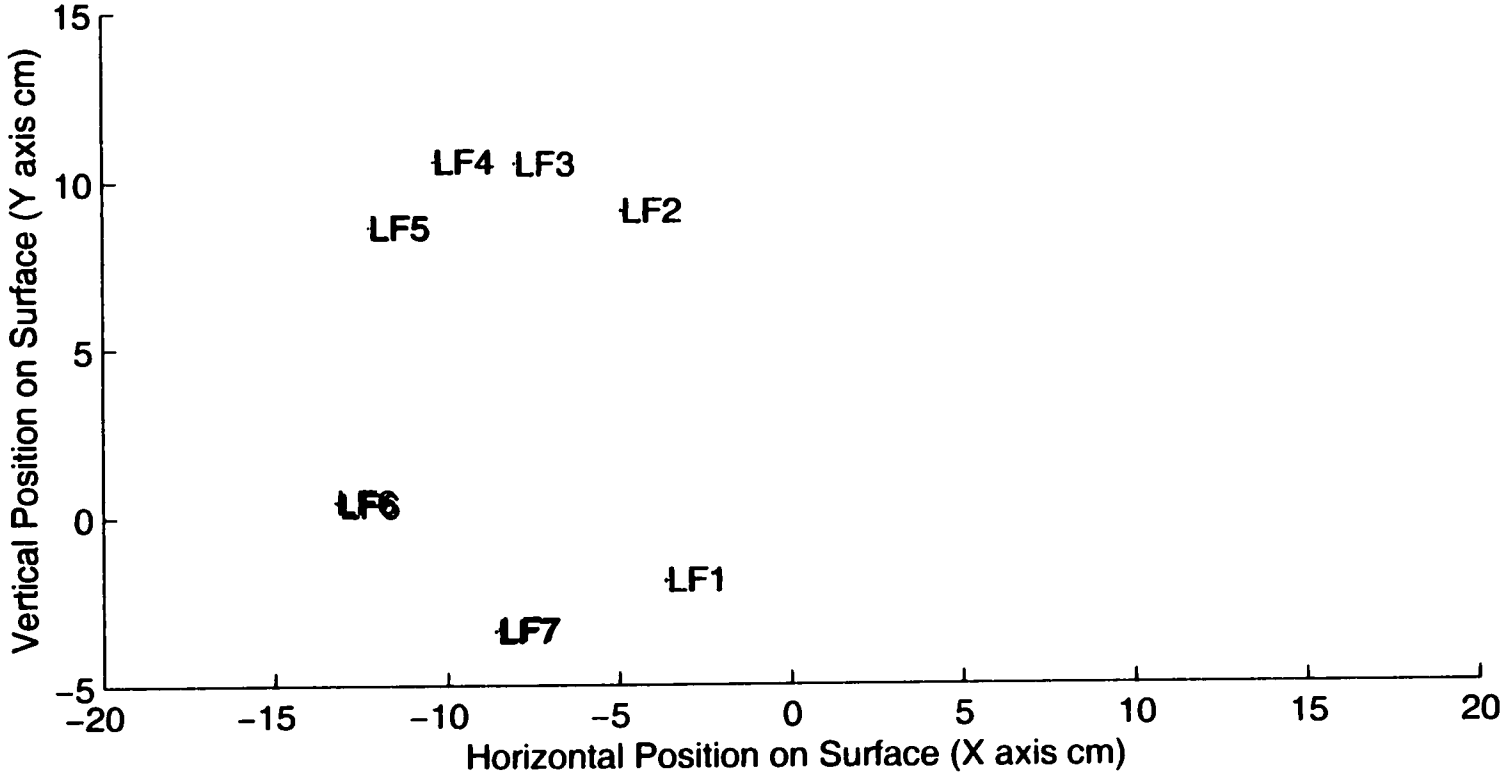

Figure 4.45: A right hand touches down well into the left half of the surface and is misidentified as the left hand. To avoid potentially destabilizing ambiguities, the strength of a hand's thumb orientation and handedness factors is intentionally limited at the far side, and in this case they are unable 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.

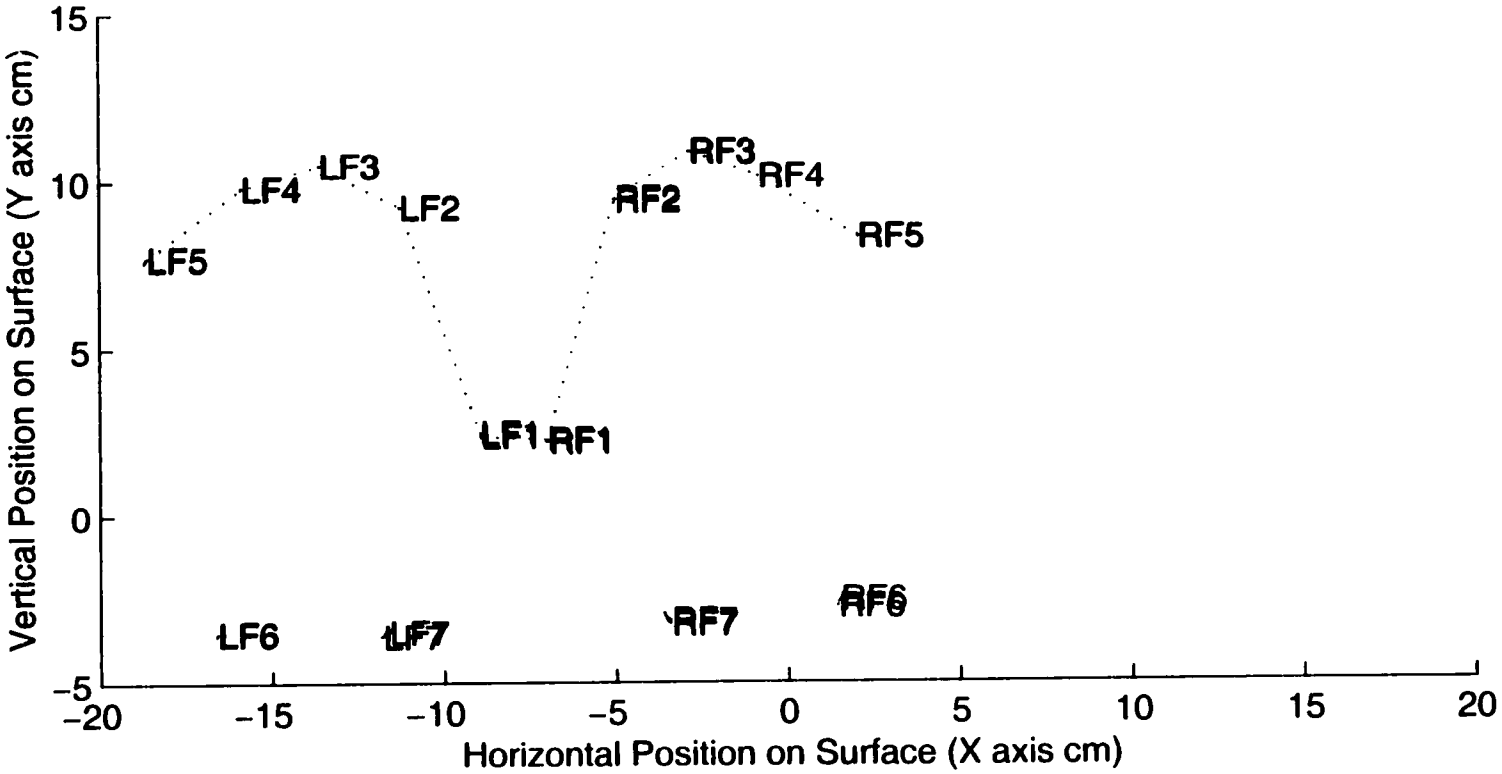

Figure 4.46: Two full hands placed side by side on the left half of the surface are partitioned correctly, the clusters being split right between the thumbs. The inter-hand separation factor, handedness factors, thumb orientation factors, and palm cohesion factors all compete to provide this correct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

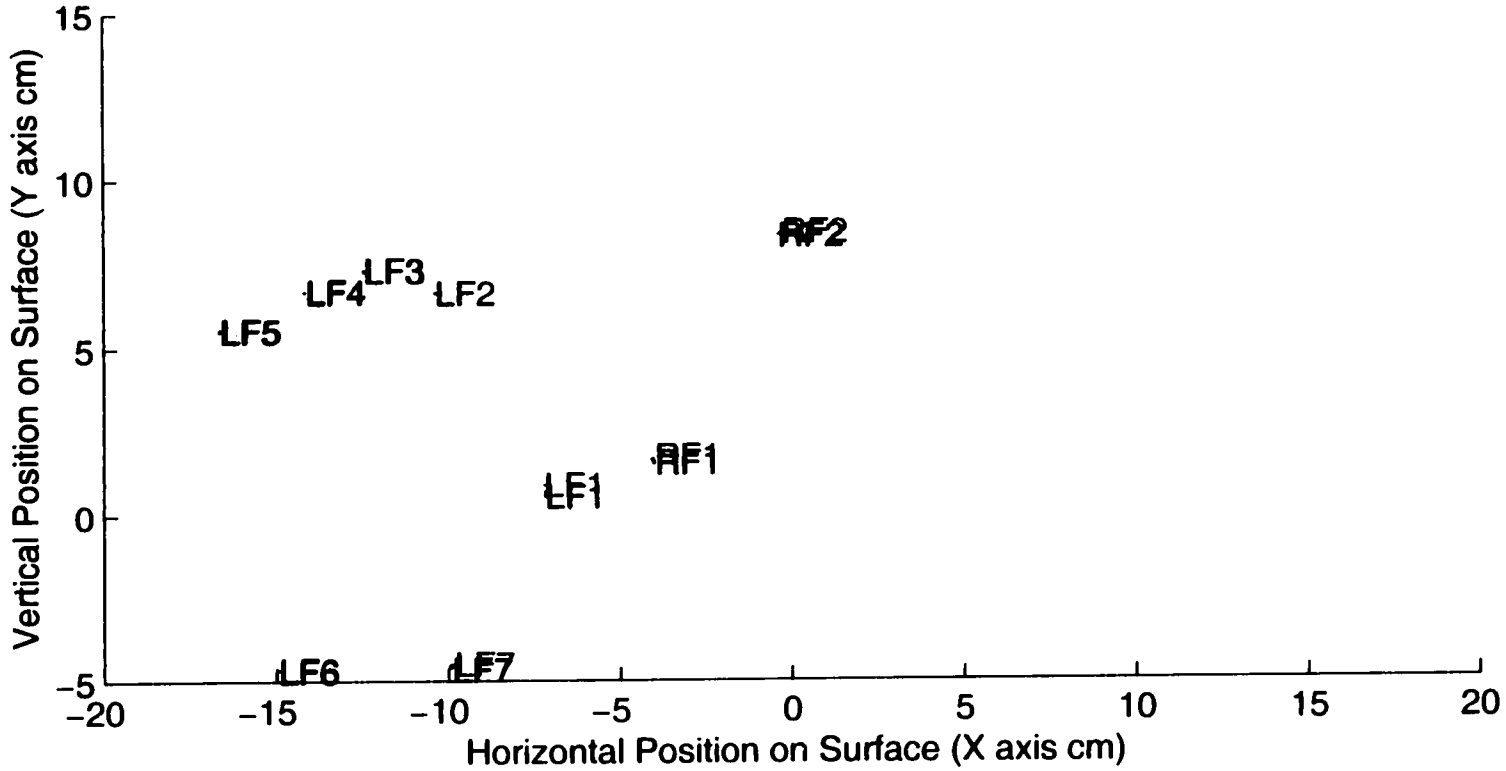

Figure 4.47: The entire left hand plus the right thumb and middle finger placed side by side on the left half of the surface are partitioned correctly, the clusters being split right between the thumbs. The inter-hand separation factor, handedness factors, thumb orientation factors, and palm cohesion 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.

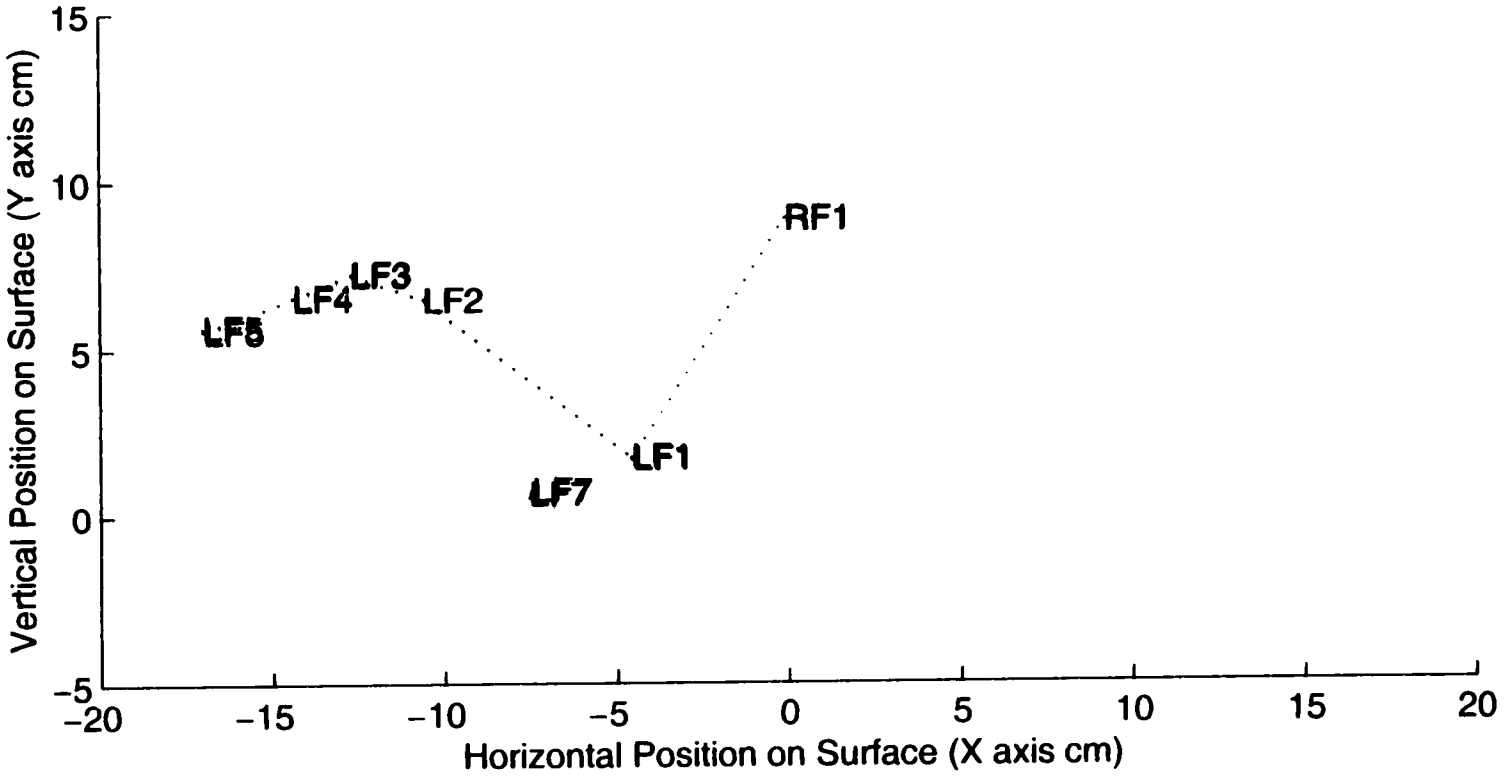

Figure 4.48: The five left hand fingers plus the right thumb and middle finger placed on the left half of the surface are not partitioned correctly. Since the left palm heels do not actually touch and therefore leave their attractors open, the system finds a bad partition in which the left thumb is misidentified 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.

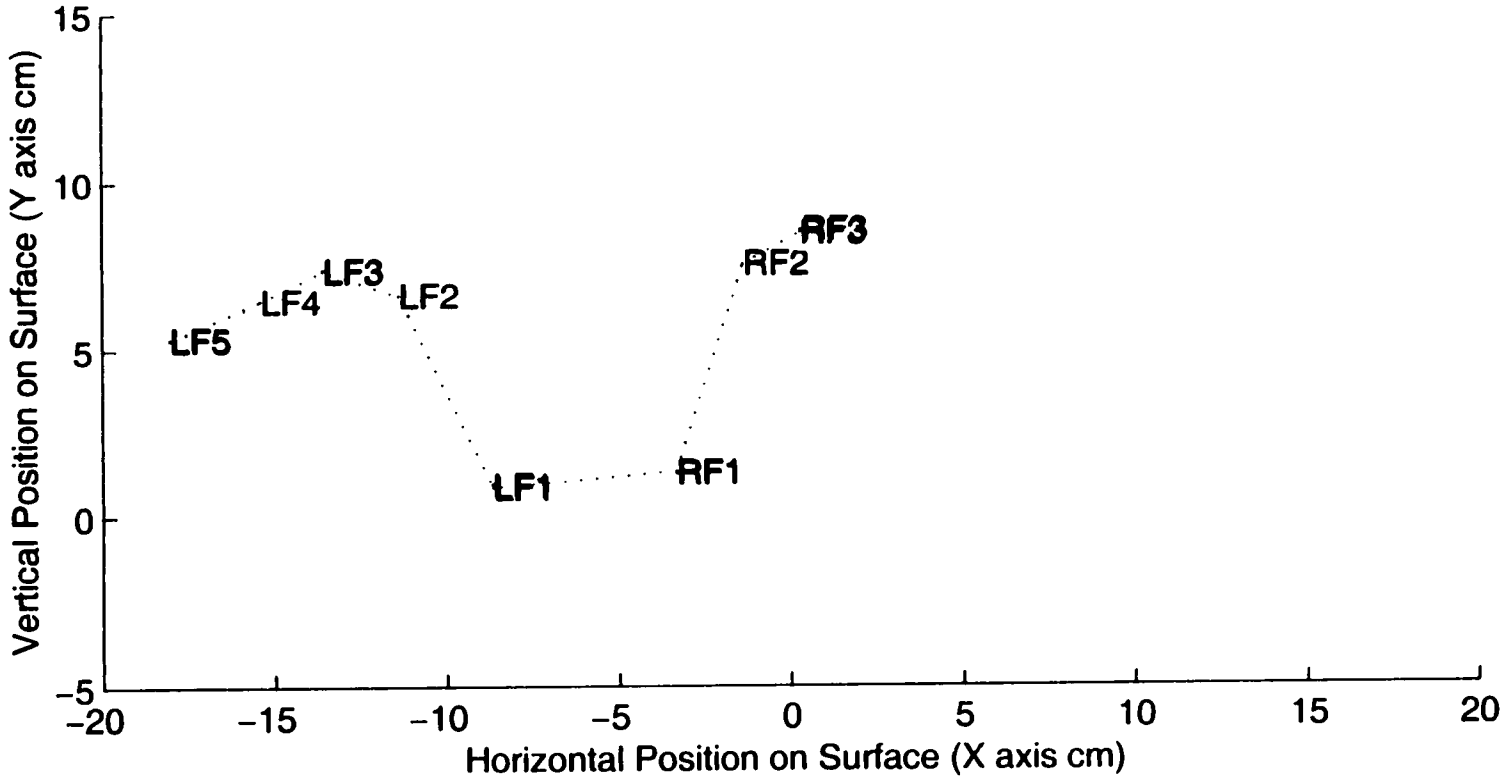

Figure 4.49: The five left hand fingers plus the right thumb, index, and middle fingers placed side by side on the left half of the surface are partitioned correctly. The only difference between this and Figure 4.49 is that the right index finger has been added and the thumbs are spaced apart a bit more,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

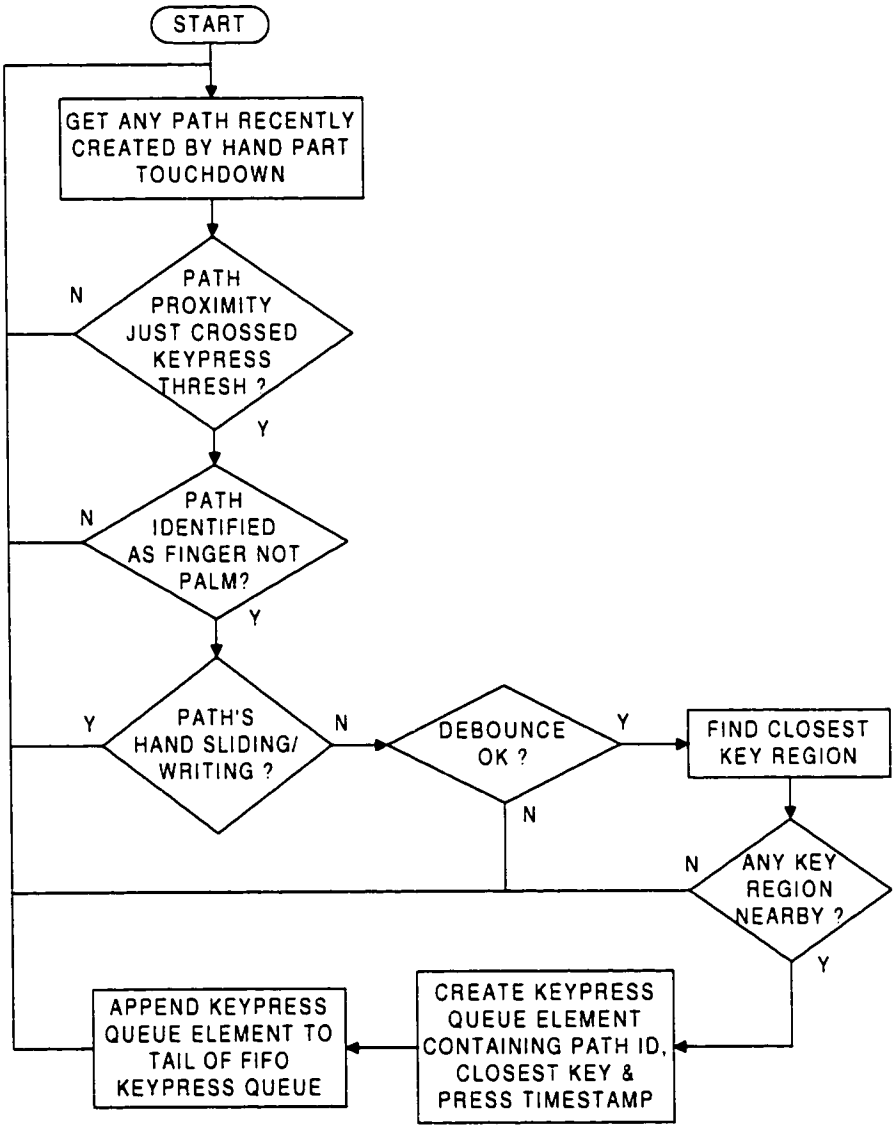

Figure 5.1: Flow chart of the keypress registration process.

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

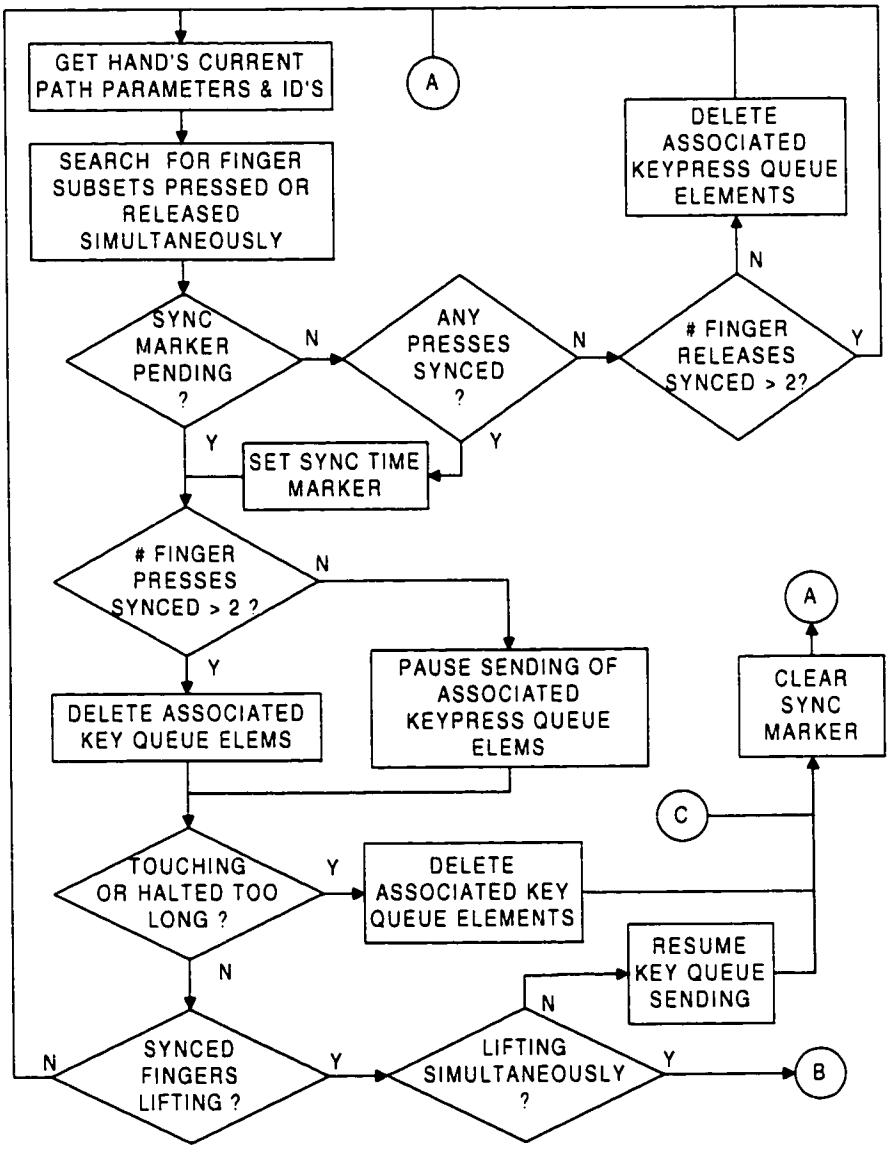

Figure 5.2: Flow chart of the finger synchronization detection process.

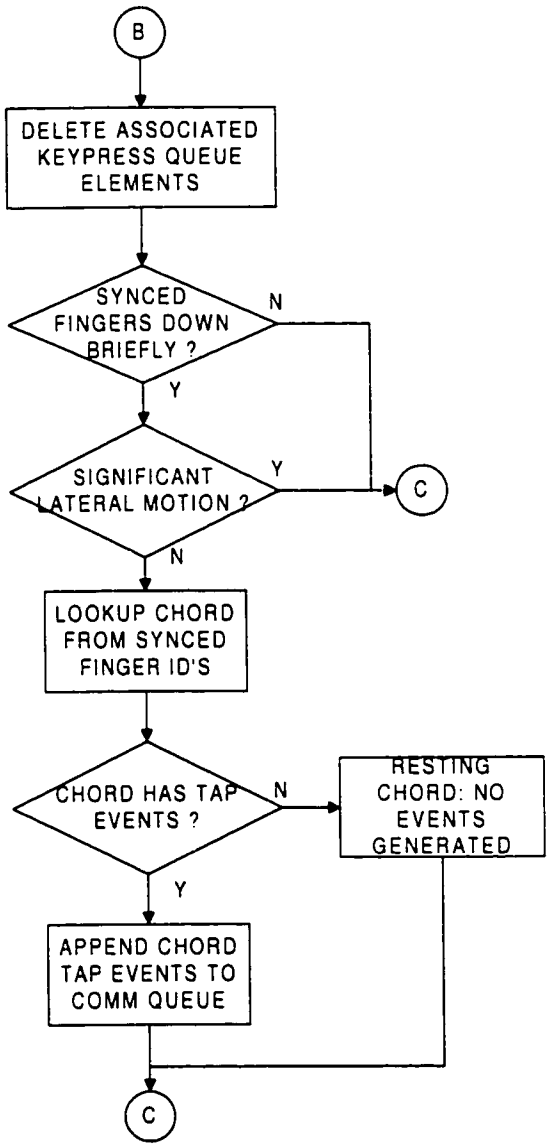

Figure 5.3: Continuation of Figure 5.2 showing chord tap detection and transmission.

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.

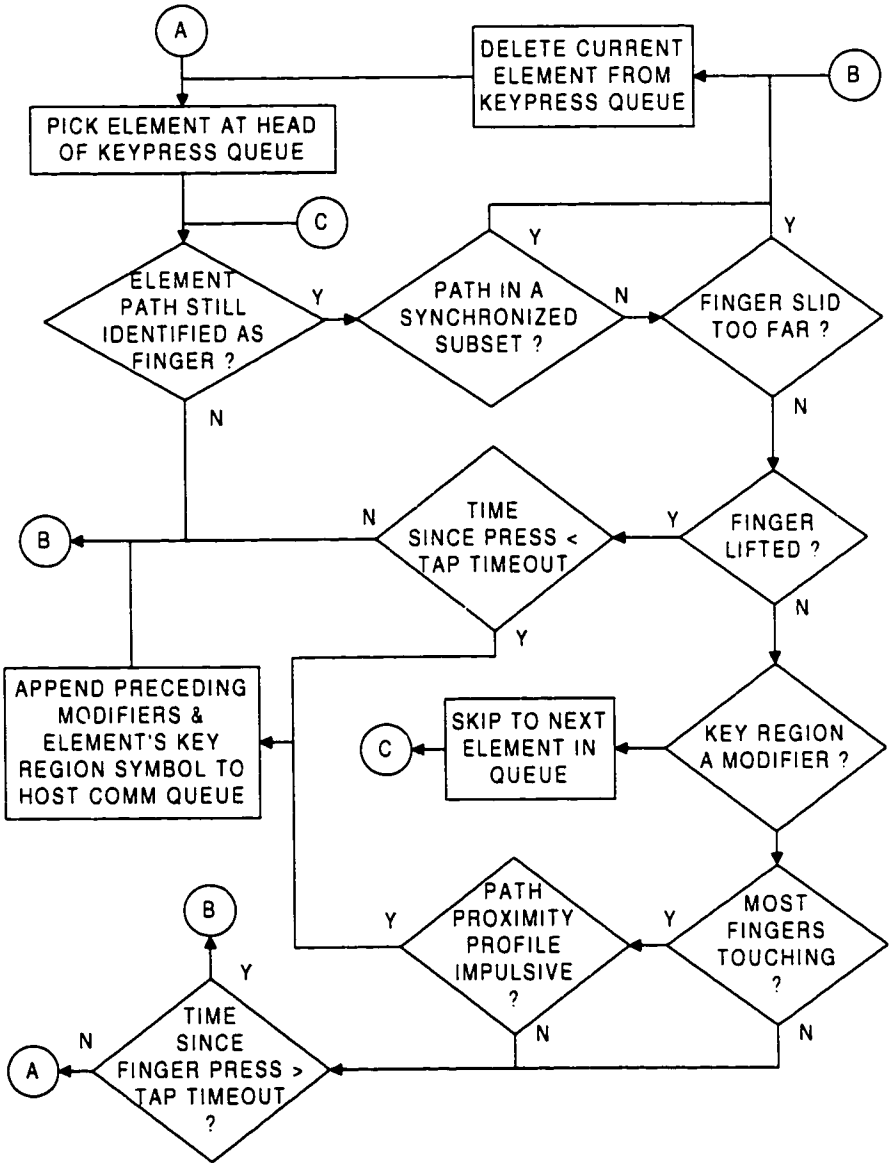

Figure 5.4: Flow chart of the keypress acceptance and transmission process.

- 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

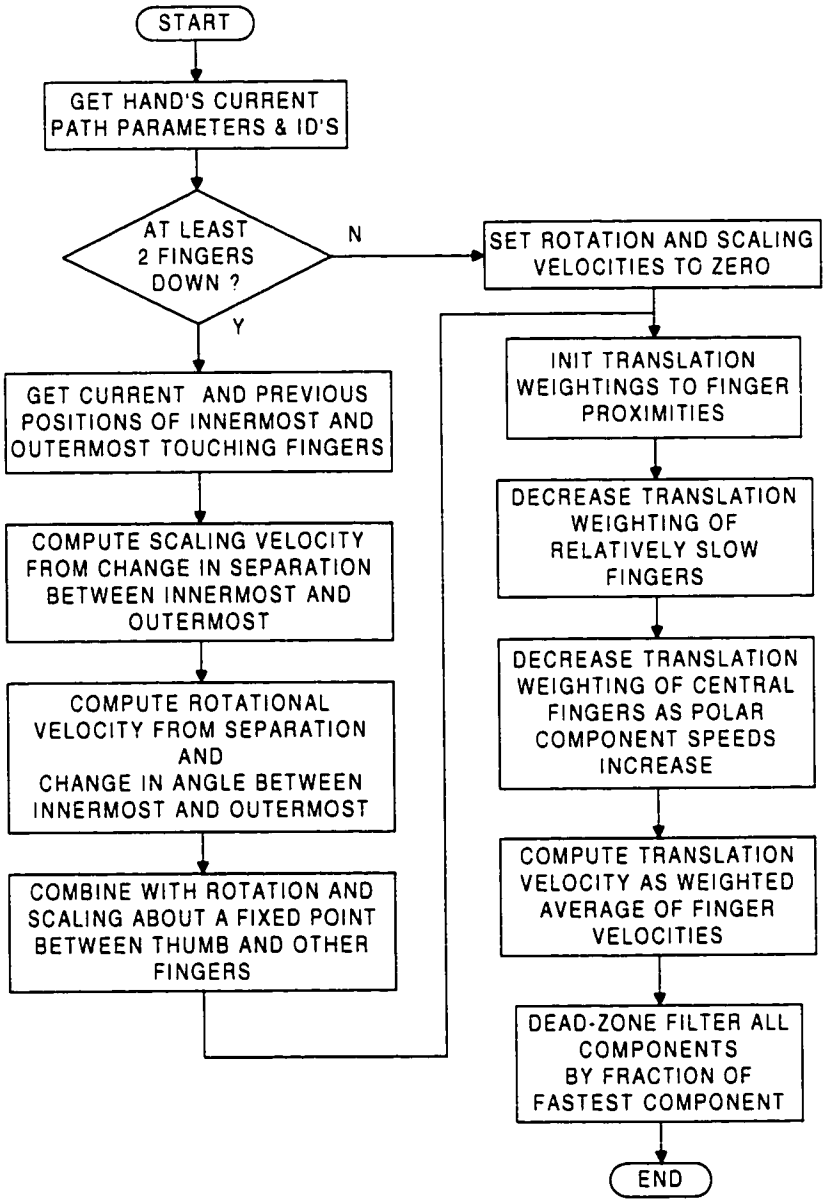

Figure 5.5: Flow chart of the algorithm for extracting hand scaling, rotation, and translation velocities from individual finger velocities.

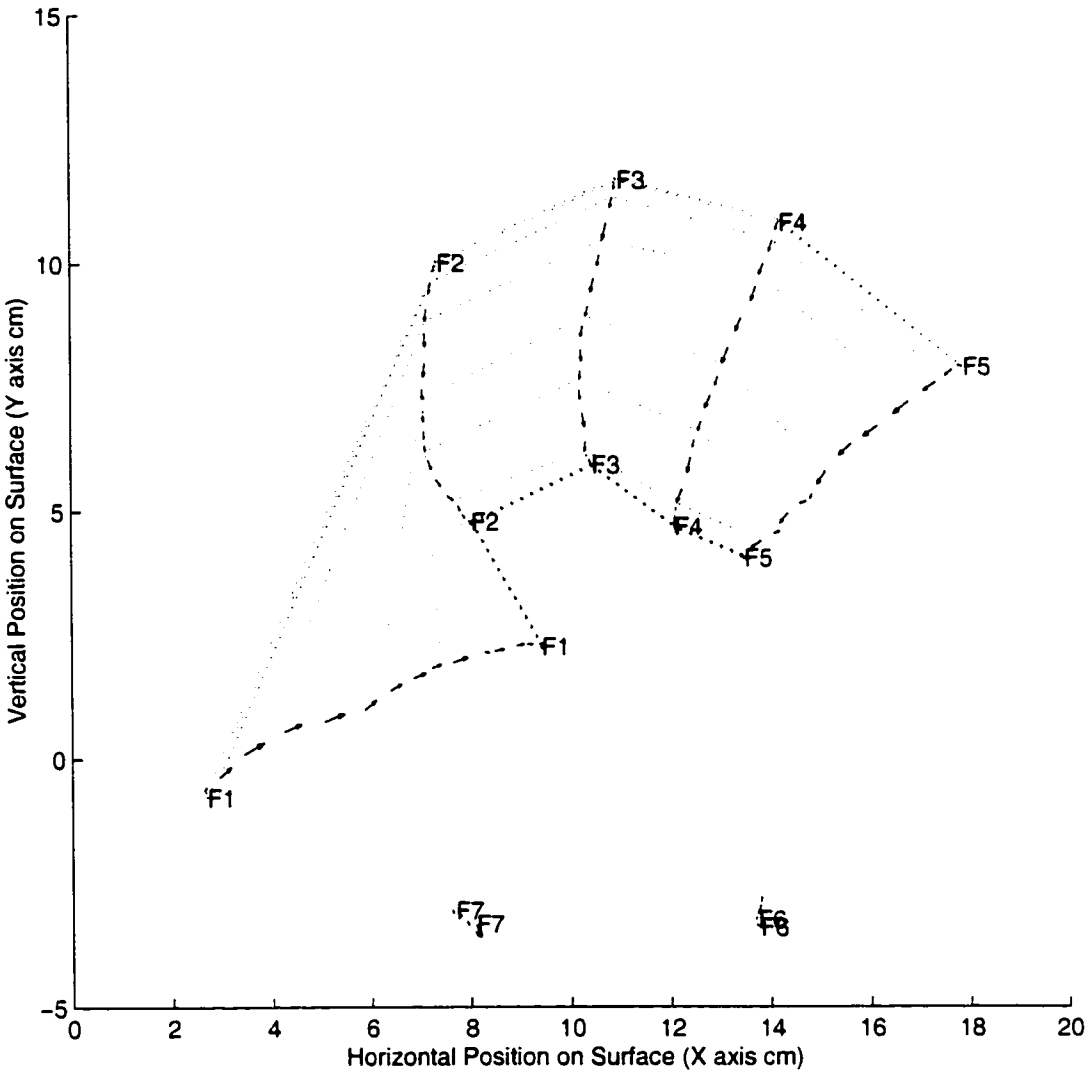

Figure 5.6: Typical flexing finger trajectories when performing a hand scaling. Note that if added as translations the thumb and pinky motions cancel, but since the palms are stationary, the motions of the central (index, middle, and ring) fingers go uncanceled and result in a net downward translation.

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:

$$H_{vs}[n] = \frac{d(FI[n], FO[n]) - d(FI[n-1], FO[n-1])}{\Delta t} \tag{5.4}$$

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:

$$H_{vr}[n] = \left(\frac{\angle(FI[n], FO[n]) - \angle(FI[n-1], FO[n-1])}{\Delta t}\right) \times \left(\frac{d(FI[n], FO[n])}{\pi}\right) \tag{5.5}$$

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

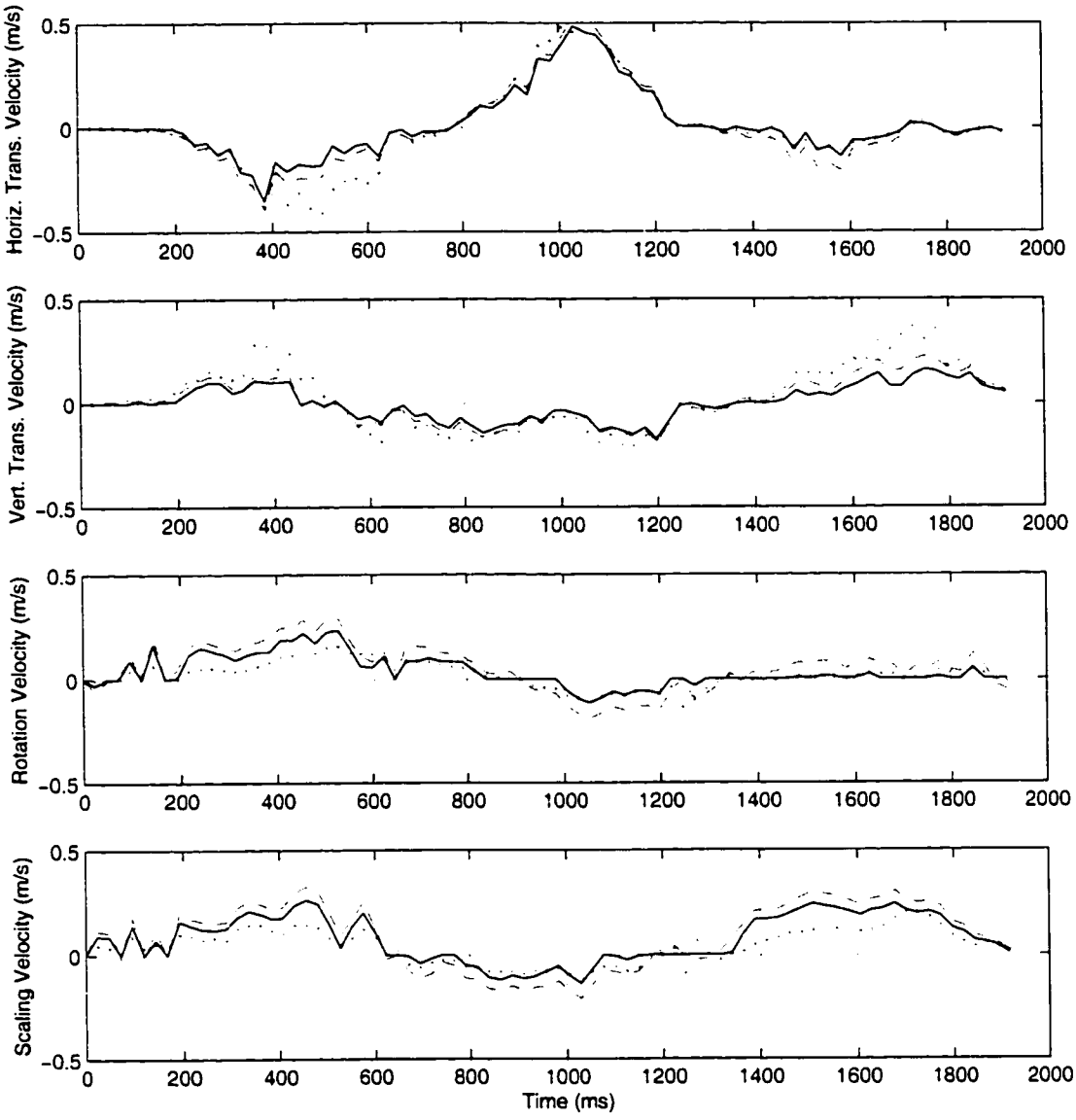

Figure 5.7: Velocity components extracted from simultaneous hand translation, rotation, and scaling. Up to 700 ms, a right hand slides in a circle with fingers expanding and rotating counter-clockwise. From 700–1300 ms, the hand continues in a circle while fingers flex and the wrist rotates back clockwise. 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.

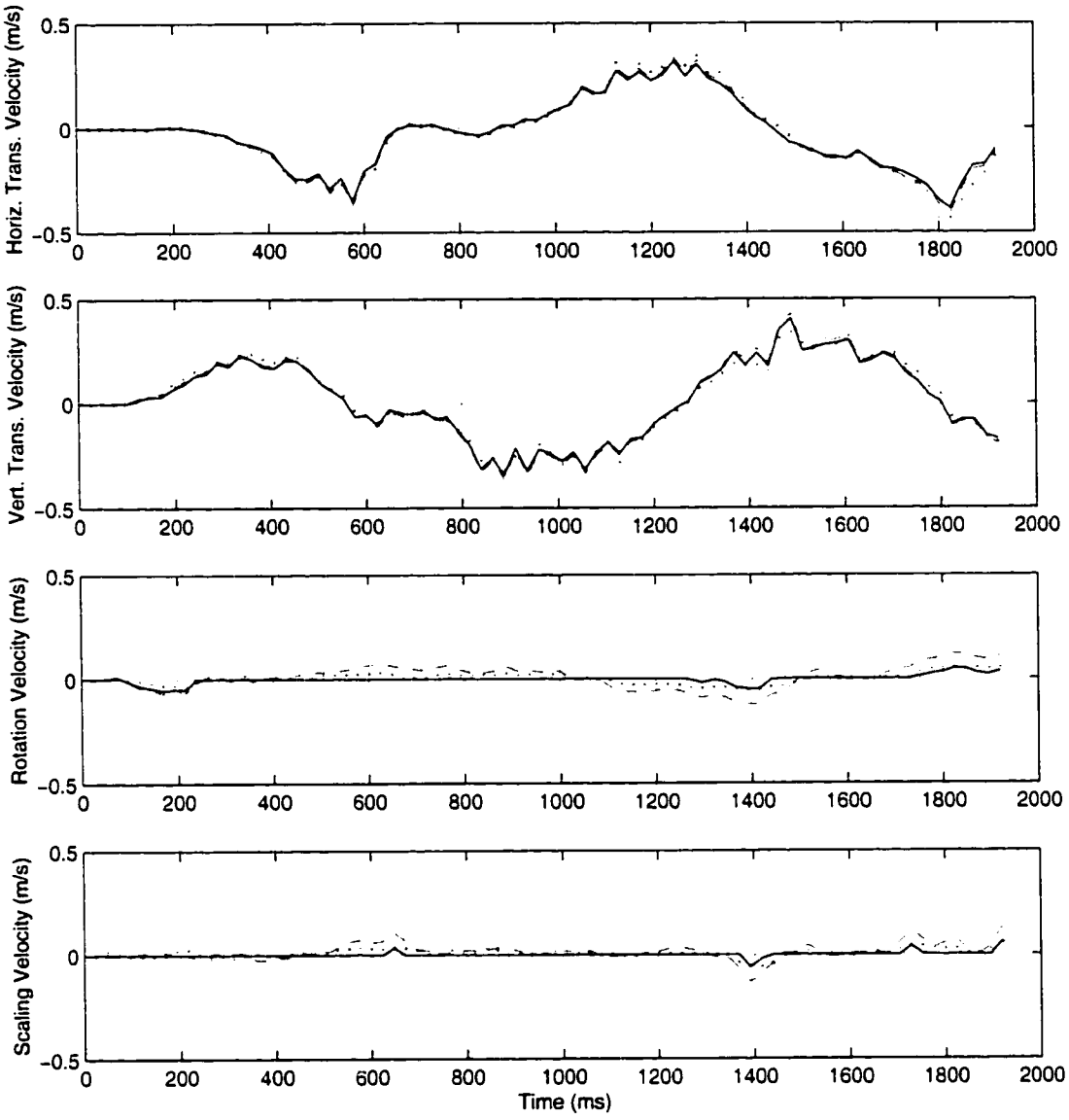

Figure 5.8: Velocity components extracted from whole-hand translation. The hand moves in a rough circle, causing the horizontal and vertical components to resemble sine and cosine waves. The fingers are not flexed nor the wrist rotated actively, but undoubtedly slight passive shifts occur in their posture 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.

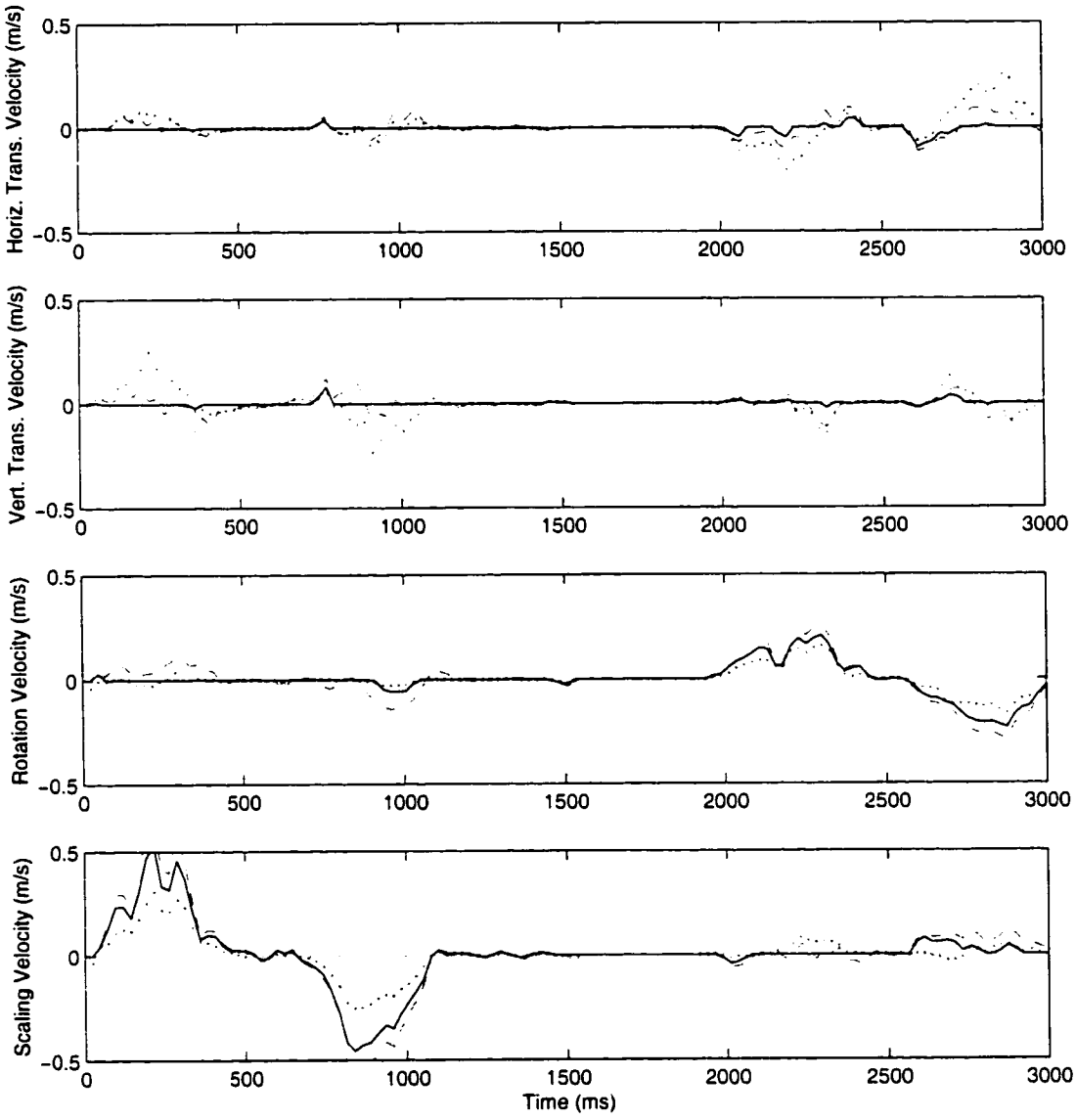

Figure 5.9: Velocity components extracted from separate hand rotation and scaling motions. Up to 1000 ms, the fingers extend and flex back smoothly. From 1000–3000 ms, the whole hand rotates counter-clockwise and then back clockwise. Dotted lines are uniform averages of finger motions, dashed are finger-weighted 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-

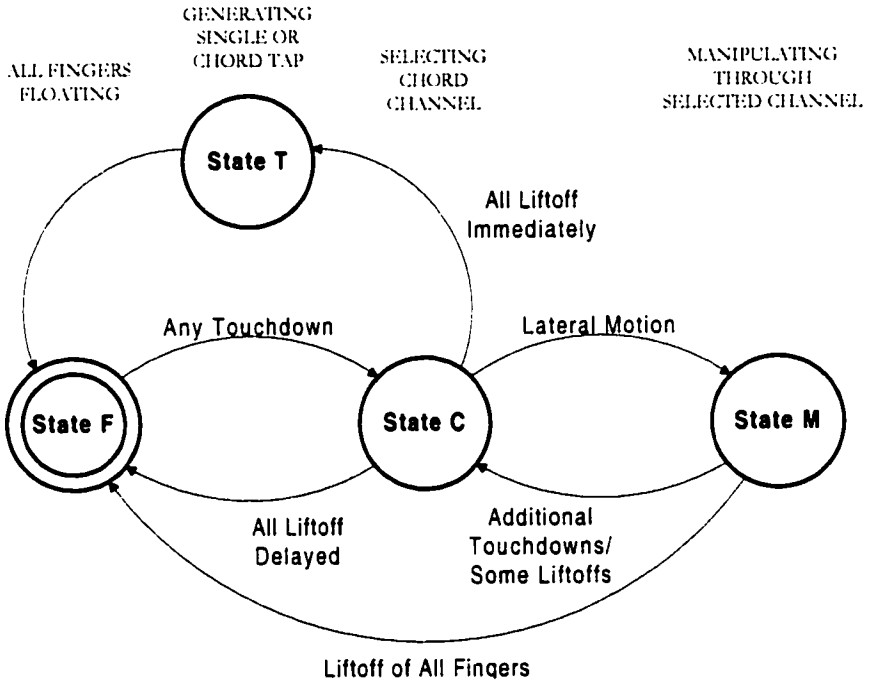

Figure 5.10: State diagram for 3-finger touchpad tapping and sliding.

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

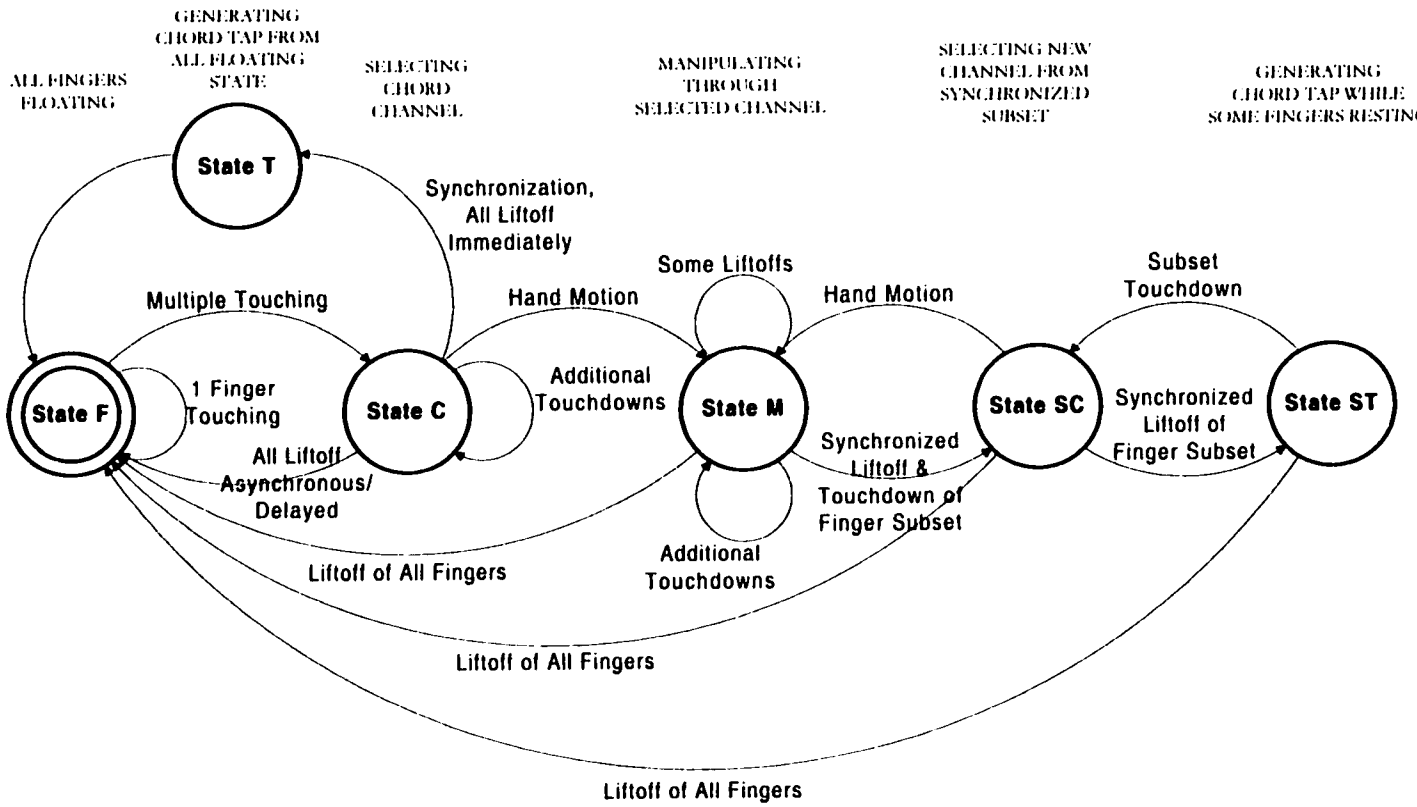

Figure 5.11: State diagram for the MTS chord motion recognizer. States C and T allow channel selection and chord tapping when the hand starts in state F floating above the surface. State M issues commands and manipulation events in response to hand slides, and states SC and ST allow a new chord channel 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

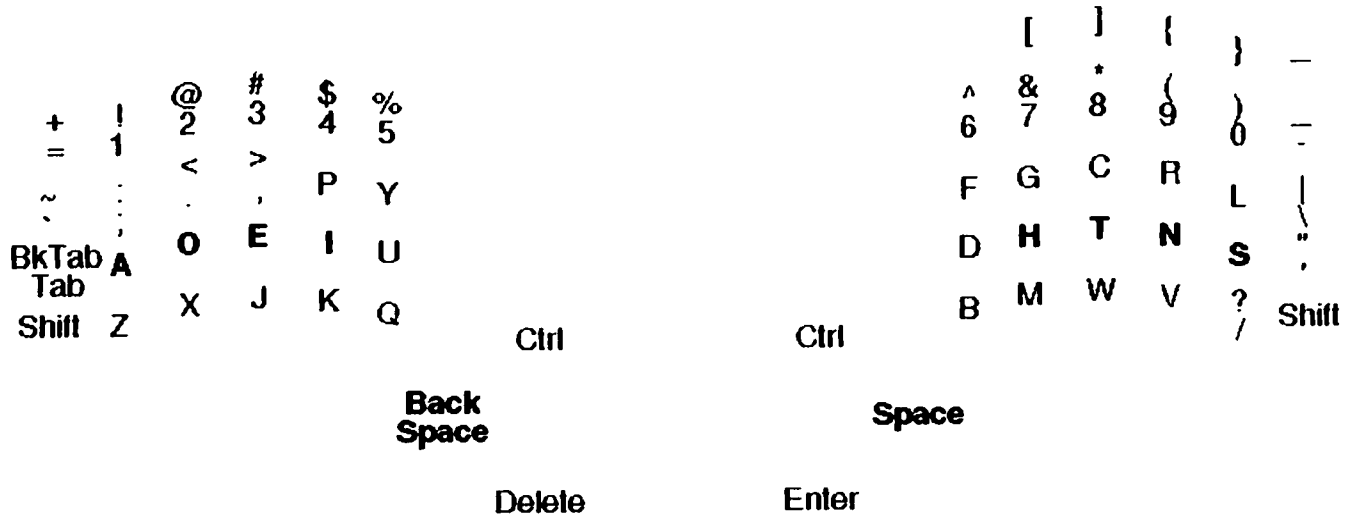

Figure 6.1: Modified Dvorak key layout adopted by the author starting in the fifth week of the trial. Note 'i' and 'u' are swapped from normal Dvorak to decrease frequency of reach by left index finger. Also, 'x' and 'q', '.' and ',', 'z' and '/' and ' and ';' are swapped to decrease differences from 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 × 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