Hand and Finger Registration for Multi-Touch Joysticks on Software-Based Operator Control Units

Mark Micire, Eric McCann, Munjal Desai, Katherine M. Tsui, Adam Norton, and Holly A. Yanco
University of Massachusetts Lowell, One University Avenue, Lowell, MA 01854
{mmicire, emccann, mdesai, ktsui, anorton, holly}@cs.uml.edu

***Abstract*—Robot control typically requires many physical joysticks, buttons, and switches. Taking inspiration from video game controllers, we have created a Dynamically Resizing, Ergonomic, and Multi-touch (DREAM) controller to allow for the development of a software-based operator control unit (SoftOCU). The DREAM Controller is created wherever a person places his or her hand; thus we needed to develop an algorithm for accurate hand and finger registration. Tested with a set of 405 hands from 62 users, our algorithm correctly identified 97% of the hands.**

***Index Terms*—Robot teleoperation, multi-touch joystick, finger registration, hand detection**

I. Introduction

For the purposes of robot teleoperation, there are several ways to control the movement of the platform and all of its various sensors, effectors, and lighting. The most common control method involves a mix of a large number of joysticks, switches, dials, buttons, and sliders that each manage some degree of freedom or functionality (e.g., Foster Miller's TALON shown in Fig. 1 (left)). These widgets are physical entities that provide tactile feedback to the operator. However, the operator control unit (OCU) layout is static and redesigning it to change its layout or add functionality can be quite costly.

A software-based OCU (SoftOCU) using multi-touch displays (shown in Fig. 1 (right)) will be a powerful replacement for traditional OCUs. Because the display is the interface, prototyping layouts is as simple as creating a paper prototype [**?**], and the cost to add a new button is just a few lines of code. Another benefit is dynamic adaptation for an individual user's hand size, range of motion, and hand placement.

Fig. 1. The Foster Miller TALON robot controller (left) is operated through three two degree of freedom joysticks and a large array of switches, selectors, and dials. (Photos courtesy of iRobot Corporation and the National Institute of Standards and Technology.) Our SoftOCU (right) is shown running a multi-touch interface for a flight simulation that includes the DREAM Controller. The SoftOCU uses 3M Multi-Touch Technology, based on 3M Projected Capacitive Touch Technology, using two 22-inch 3M Multi-Touch Display M2256PW monitors.

We have been developing an adaptive user interface for a SoftOCU using our **D**ynamic **R**esizing **E**rgonomic **A**nd **M**ulti-touch (DREAM) Controller. The DREAM Controller provides an intuitive method for analog input with four total degrees of freedom (two on each thumb) and a wide range of other inputs via two highly customizable control panels. Because it is dynamically generated wherever a person places his or her hand on the multi-touch screen, tactile feedback is not required. The input possibilities provided by DREAM Controllers and multi-touch technologies for a SoftOCU could revolutionize the future design of operator control units.

To spawn a DREAM Controller, the fingers touching the screen need to be reliably and efficiently identified as digits of either a left or right hand with each finger registered correctly. This paper describes our hand and finger detection algorithm for performing this task and its accuracy. The algorithm is versatile in terms of the platforms on which it can be implemented, as its input is a single Cartesian point for each finger tip. We have implemented our hand and finger detection algorithm on the Microsoft Surface (diffuse infrared illumination) and 22-inch 3M Multi-Touch Display M2256PW monitors (3M Projected Capacitive Touch Technology).

II. Related Work

The algorithms used in most multi-touch interfaces do not consider which finger is touching the contact surface. For most applications, this design is preferred since users have been shown to regularly use any finger or combination of fingers for interface operations like dragging, selection, and zooming [1] [2]. However, in our design of a multi-touch controller, registration of the individual fingers and the identification of the hand as a right or left hand is needed. The thumb, index, middle, ring, and little finger are vital to the position, orientation and sizing of the joystick.

Agarwal et al. [3] applied machine learning techniques to accurately find fingertips and detect touch on regular tablet displays using an overhead stereo camera. Researchers have also investigated machine learning for hand poses, which does not take fingertip detection into consideration [4]. While these approaches are robust and able to find fingers using overhead cameras, it is not clear that the fingers themselves were labeled in terms of thumb, index, etc. The Surface and 3M monitors are able to easily find the finger tips and return the centroid points, so the finger tip detection problem was solved for us by

the hardware and software provided with the device. Instead, our algorithm needed to specifically label the fingers relative to the anatomy of the hand.

Wang et al. [5] developed an algorithm for detecting the orientation of finger pads for camera-based multi-touch devices. The orientation is derived from the angle of the vector from the contact center in the previous frame through the current frame which points away from the user's palm. This algorithm has been shown to work well for individual fingers. However, we needed to be able to uniquely identify finger types for our controller. Finger orientation for multiple fingers can be used to help determine if the fingers are on the same hand, but this does not necessarily indicate handedness. Additionally, people have varying flexibility to spread their fingers (e.g., a pianist or a person with arthritis), and it is difficult to robustly identify fingers based directly on contact orientation when this surface area may change dramatically while on the tips of the fingers.

Dang et al. [6] developed an algorithm to map fingers to unique hands also based on ellipses created by fingertip contact regions. Like [5], the researchers first detect the individual fingers' orientations using the major and minor axes of the ellipse and the angle of orientation of the major axis vector. A combination of heuristics allows for accurate detection of fingers mapped to a unique hand even when not all of the fingers from a given hand are placed on the surface. While their approach is robust, the reliance on ellipses makes it a difficult choice when we expect a portion of our users to use the tips of their fingers while learning the system's behaviors.

Matejka et al. [7] successfully emulated a mouse on a multi-touch platform using simplistic and elegant heuristics for finger tracking. In their paper, they describe a method that uses four primary methods for interaction: Chording, Side, Distance, and Gesture. They constructed a state machine that used timing for registration of the initial tracking finger or fingers. For subsequent contacts they measured the distance in pixels to identify which fingers were in contact. While their method is not computationally expensive, it makes the base assumption that the user knows to begin with the index finger before any subsequent finger contact. It also assumes that the user's hands conform to these static offsets. Since explicit finger identification, handedness, and automatic sizing was needed for our application, the technique in this research was not appropriate despite other benefits in its design.

III. Design of a Multi-Touch Joystick

Leveraging video game paradigms is a powerful tool an interface designer can employ to improve ease of learning. Familiarity with a different interface that uses a similar input method is practically free experience, at least in terms of how long it will take a new user to perform passably and, eventually, to become proficient. Fortunately, the capabilities of real robots closely mirror those of virtual game characters: cameras can look left, right, up, and down, while the robot can drive forward, backward, turn left, and turn right. A popular controller paradigm established in the late 1990's by the Sony Playstation® and the Microsoft® XBox® for video

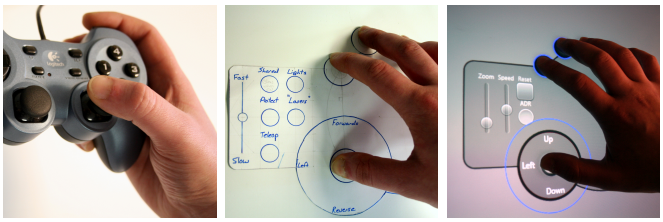

Fig. 2. A dual-thumb joystick modeled after the Sony Playstation controller (left) inspired the design of a paper prototype (center) that was selected for the design of the multi-touch DREAM Controller (right).

games used a dual-thumb joystick design that allowed each thumb to manipulate two degrees of freedom, with various buttons located in locations comfortably reachable by the thumb and the player's hand's other digits. First-person games often employ a mapping where the left thumb joystick controls lateral movement of the camera, with the rotation of the perspective of the camera, controlled by the right. As both robots and first-person games have such similar capabilities, employing this game mapping for robot control makes adapting to DREAM Controller-based interfaces even easier for video gamers, without sacrificing learnability for non-gamers [8].

The general stages of the design process we used are illustrated in Figure 2. As multi-touch screens are large and flat, we decided to make use of the real-estate such an input device provides in order to alleviate some of the ergonomic issues that a flattened-out video game controller could produce. Rather than forcing close left and right hand positions, as in the case of a physical game controller, we decoupled the two halves of the controller. This independence allows for the joysticks to be used on different areas of the screen, at different angles, and even allowing them to be different sizes. An unintentional, but powerful, side-effect is that the joysticks need not be the only functional part of the interface. As the user can choose to have both, either one, or neither of the joysticks on the screen, the window itself can still be a functional multi-touch interface in complete parallel with the use of the joysticks.

In designing the DREAM Controller, we took into account both human-computer interaction (HCI) criteria derived from Nielsen's five usability attributes (learnability, efficiency, memorability, errors, and satisfaction) [9] and ergonomic criteria based on the anatomy of the human hand. (Details about the HCI criteria are discussed in [10].) We engineered the controller with a significant respect for the resting poses for the human arm, wrist, and hand. The paper prototypes in the early design process helped minimize flexion (decrease of angle), extension (increase of angle), and pronation (downward rotation) of the muscles in the wrists and fingers [11]. We also considered that the movements of the thumb, index finger, and little finger have been shown to have much more individualized movement characteristics than the middle or ring fingers [12]. In particular, the movement of the thumb for managing the two degrees of freedom for robot movement and camera positioning must be appropriate for accurate and long-term use.

Two sources of information were important in establishing the ergonomic requirements of the DREAM Controller. A

wealth of ergonomic information related to gestural interfaces can be found in [13]. The six key principles of ergonomics that were used in the design of the controller were: avoid outer positions, avoid repetition, relax muscles, relaxed neutral position is in the middle between outer positions, avoid staying in static position, and avoid internal and external force on joints and stopping body fluids. Each one of these principles was evaluated during the prototyping phase and influenced our design. Another source of anatomical information (and inspiration) was found in the 1955 book by Henry Dreyfuss titled Designing for People [14]. The book contains composite figures of human anatomy gathered from years of research and data collection. It was with the aid of his technical drawings that we began decomposing the anatomy of the hand and recognized that the fingers were limited in their lateral deflection angles. Even in extreme stretching, there were key characteristics that we could exploit to identify the components of the hand.

IV. Hand Detection Algorithm

The hand detection algorithm needs to have a low error rate in order for the DREAM Controller to be usable. Our original algorithm (Version 1), described in [8], [10], correctly identified 91.6% hands (371 of 405, collected from 62 people). In this algorithm, when a user puts five fingers down, we check to see if the distance between point pairs is less than a threshold based on Dreyfuss' measurements [14] and empirically refined to 8 inches. If so, we create a bounding box around the five points with its sides parallel to the major axes. The centroid of this box roughly represents the center of the palm. A sorted list of angles between adjacent points and the centroid is calculated. The largest angle is between the thumb and little finger; the second largest angle is between the thumb and index finger. The remaining points can be enumerated, and handedness can be determined based on the sign of the angle between the thumb and index finger.

In validating Version 1 of the algorithm, we discovered a disproportionate number of the incorrect identifications occurred when the user was oriented at the screen from an off-axis direction (positions 1, 3, 5, and 7 in Figure 4 (left)). As shown in Table I, these placements were identified correctly 85.2% (167 of 196) in comparison to hand placements where the user was directly facing one side of the screen (97.6%; 204 of 209). An analysis of our original algorithm revealed that when the user was along the side of the screen (positions 0, 2, 4, of 6), the bounding box around their five fingers was approximately a square, and the centroid was approximately at the center of the physical hand. However, when the user rotated his or her hand about 45 degrees (i.e., standing at a corner), the resulting bounding box was an elongated rectangle, thereby causing the centroid of the bounding box to deviate from its intended location relative to the physical hand. This small deviation didn't affect the thumb identification, but the handedness and other finger identifications would be incorrect. In our improved hand detection algorithm (Version 2), the bounding box rotates with the hand to resolve this.

In Version 2, when a finger touches the device, the contact's center point is added to a list of candidates. When there are five candidates in the list, the candidate list is passed through a heuristic to determine if those points pass a preliminary hand test (Fig. 3a), based upon the 8 inch maximum distance threshold. If the candidates fail this heuristic, the handedness is declared to be unknown, which is a terminal state for this algorithm. Otherwise, the rotated bounding box around all five of the fingers is calculated.

The bounding box is rotated so that two sides are parallel to and two are perpendicular to the best-fit line through the non-thumb fingers. We use two levels of distance-based sorting to make an educated guess about which four points could be the non-thumb fingers. In the first pass, the "thumb" is found by finding the point that has the highest total distance from the other four points (Fig. 3b). We then order the remaining four points using a breadth-first strategy, beginning with the finger closest to the thumb.

We then calculate the average slope of the three lines that join adjacent pairs of points (Fig. 3c). If the slope is 0 or undefined, then the centroid is found using the coordinates of the fingers themselves instead of the coordinates of the corners of the rotated bounding box (as was done in the Version 1 algorithm). If the slope is defined, we compute the equation of the line parallel to the average non-thumb slope for the each of the fingers (Fig. 3d). Similarly, we compute the equation of the line perpendicular to the average non-thumb slope for the each of the fingers (Fig. 3e) The two lines with the maximum and minimum y-intercept from each group form the sides of the bounding box around the five fingers. To find the four corners of the bounding box, we calculate all of the intersections of those four lines. The corners are then used to compute the centroid (Fig. 3f).

After the centroid is located, we calculate the maximum and minimum distance between it and any candidate (Fig 3g). If the ratio of the maximum over the minimum is more than 2, we determine that the handedness is unknown, as we found empirically that valid hand placements usually have a ratio between 1 and 1.5. Otherwise, the finger registration algorithm attempts to figure out which of the five points correspond to specific fingers. First, a sorted list of angles between adjacent points and the centroid of the bounding box is populated (Fig. 3h). The largest angle in this list represents the angle between the thumb and the little finger. The second largest angle represents the angle between the thumb and the index finger. The thumb is the only point that occurs in both the largest and second largest angle. The complimentary point of the largest angle is the little finger and the complimentary point of the second largest is the index finger. The middle and ring finger can then be identified logically.

Once the specific fingers are identified, two additional tests to ensure the candidates correspond to a hand are performed. The first calculates the average distance between any two adjacent fingers. If there are more than two inter-finger distances above that average, then the hand is declared to be unknown (Fig. 3i). The second calculates the distance between the fingers

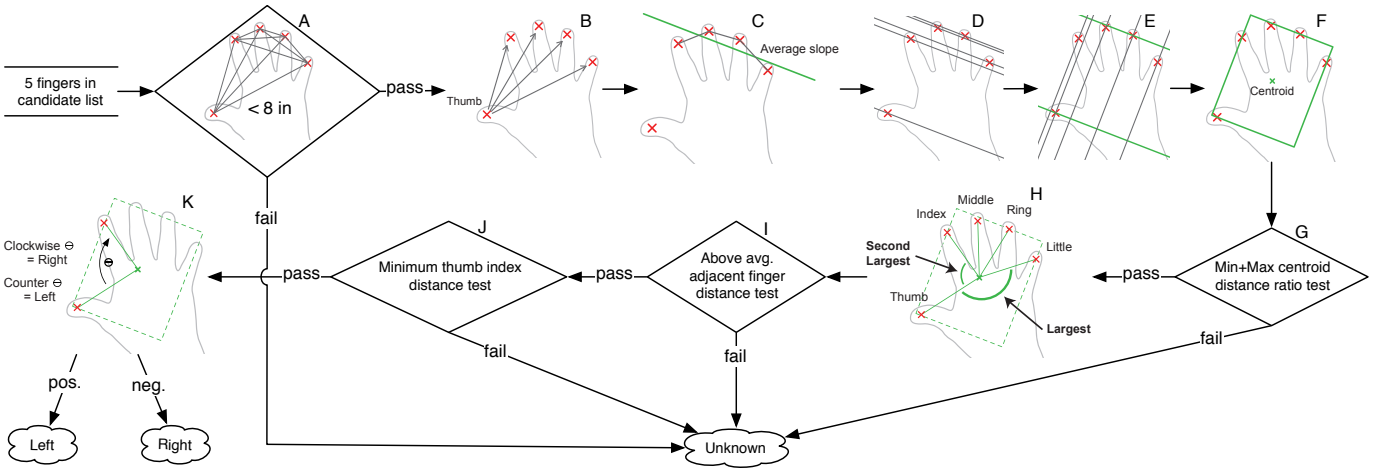

Fig. 3. Improved hand detection algorithm. **a.** Maximum hand size threshold verification. **b-e.** Calculating the rotated bounding box. **f.** Finding the centroid. **g.** The minimum/maximum centroid-finger distance ratio test. **h.** Determining the two largest angles. **i.** The above average adjacent finger distance test. **j.** The minimum thumb-index distance test. **k.** Determining whether the five points are from a left or right hand.

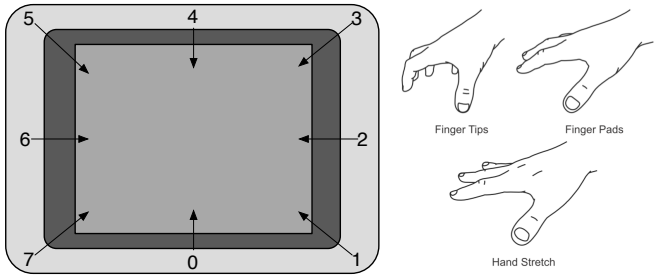

Fig. 4. **Left:** The eight screen positions users stood at in the experiment used to validate the success of our original algorithm. **Right:** The three hand positions used for each of their hands.

identified as the index finger and thumb. If the distance is less than 1 inch, then the hand's handedness is unknown (Fig. 3j). Otherwise, the hand can be identified as a left or right hand. If the rotation around the centroid from the thumb to the index finger is clockwise, then it is the right hand. If that rotation is counterclockwise, then it is the left hand (Fig. 3k). To determine the direction of rotation, we take a property of the cross product of two vectors. For the purpose of derivation, we define the vector from the centroid to the thumb as $\vec{a}$ and the vector from the centroid to the index finger as $\vec{b}$. Assuming that $\vec{a}$ and $\vec{b}$ are in the XZ plane, the cross product vector will be positive (point upwards) if the rotation from $\vec{a}$ to $\vec{b}$ is counterclockwise, or negative (point downwards) if the rotation is clockwise. Since these are simply two dimensional vectors, the cross product derivation expands to Equation 1 where subscripts c, t, and i, refer to the coordinates of the centroid, thumb's center point, and index finger's center point, respectively.

$$(x_c - x_t) \cdot (y_i - y_c) - (y_c - y_t) \cdot (x_i - x_c) \quad (1)$$

If the hand identification and finger registration process reaches this point, then all of the information necessary to begin building the DREAM Controller underneath the user's hand has been determined.

V. Algorithm Validation

We previously collected a data set of hand placements on our Microsoft Surface. Sixty-five people were each asked to stand at one of eight positions around the screen (Fig. 4 (left) and to place each hand on the screen once in each of three different hand positions (Fig. 4 (right)). We used the Microsoft Surface Simulator to record Surface Contact Scripts, which are key frame animation-like XML files that allow every touch that occurred on the screen to be played back in real time. Due to recording issues, the data set only contains the contact scripts for 62 participants.[1]

We tested both algorithms with the 62 Surface Contact Scripts to see how accurate they were at identifying hands (Table I). We first manually coded the 62 Surface Contact Scripts to determine at what time each hand detection in our original manual logs occurred and noted the handedness of any touch combinations that appeared to be a left or right hand. The raw data set contains 405 hand placements. We ran the raw data set through the Version 1 and Version 2 hand and finger detection algorithms. Each detected hand was logged with the playback time at which the detection had occurred. The two logs were then compared with the human-determined identifications of that very same data. "Correct" is defined as the algorithm returning the same handedness as was manually coded within the same second of playback as the manually coded hand.

Table I shows the percentage of correctly identified hand placements (*# correct/total placements*). The Version 1 hand detection algorithm correctly detected 91.6% (371 of 405) of the hands for those 62 users' contact scripts, and the Version 2 correctly detected 97.0% (393 of 405) of the hands.

[1]The 62 Surface Contact Scripts are available at http://robotics.cs.uml.edu/handdetection/HandSurfaceData.zip

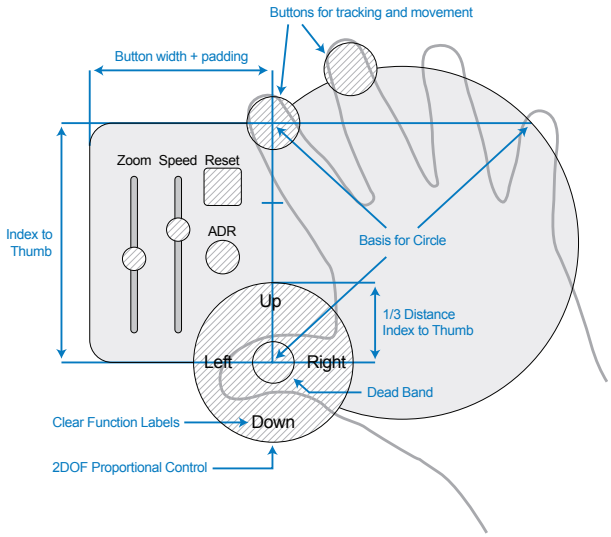

Fig. 5. Users have a wide variety of hand characteristics. The DREAM Controller adjusts several parameters to tailor the size, orientation, and position of the controller to maximize comfort and performance.

VI. DREAM Controller

The hand geometry and handedness is used to adjust the size, orientation, and arrangement of the DREAM Controller elements. As shown in Figure 5, there are some key measurements that determine these features. First, the angle from the thumb to the index finger determines the orientation of the controller and the control panel. The top right corner of the control panel is placed at the index finger and the lower right corner is placed at the thumb. The width of the grid is determined by the size of the buttons and sliders with the addition of padding for layout and visual balance. The height is simply the distance between the thumb and the index finger.

A circumscribed circle is calculated around the triangle formed by the points corresponding to the thumb, index finger, and little finger. Its purpose is both aesthetic and functional, in that it provides a feedback to the user that the controller is tailored to their specific hand size, but also plays an important role in hit detection in the application. It protects user elements in lower panels from detecting the touches from fingers of the hand that the joystick is under.

Two circles are placed under the center of the index and middle fingers. These are analogous to the "shoulder" buttons used on dual-thumb joystick controllers. The circles can receive their own events and thus can easily be used as momentary or toggle buttons if needed. Additionally, these have the potential to be used as pressure sensitive buttons since the relative size of the finger blob detected can be determined as the user changes the amount of pressure placed on the surface.

A circle is placed under the thumb representing analog control for two fully proportional degrees of freedom (DOF). Much like the Playstation controller, the thumb is then moved up, down, left, and right corresponding to the desired movement. The circles have their function clearly labeled within the boundaries of the circle. The user has full proportional control and can mix the analog ranges for both degrees of freedom. Lifting the thumb from the analog control pad is analagous to letting go of a physicial joystick, in that the virtual joystick is returned to its origin when that occurs.

When the thumb is in contact with the analog pad, the joystick widget's angle and position are locked. When the thumb is not in contact with the thumb circle, the index and middle finger can be moved to adjust the angle and position of the controller. The shunting of movement is important for safety and continuity of control. Many secondary movements are transferred to the index and middle finger when the thumb muscles are actuated. The thumb position relative to the analog control pad changes if this movement of the index finger and middle finger then rotate or translate the position of the controller. This, in turn, causes the user to move their thumb, which causes the index and middle finger to again move. This feedback loop quickly becomes very frustrating to the user and results in the user pushing down on the screen surface to limit secondary movement. Muscle fatigue will quickly occur and the user experience will be severely negative. By simply stopping movement on thumb contact, the controller maintains position and allows the user to relax his or her hand muscles without consequence. At any time during the use of the controller, the user can lift or lower their ring and little finger with no effect on operation, added for the comfort of the user after we noticed this relaxation behavior with the early paper prototypes.

VII. Discussion and Future Work

The hand and finger identification and registration algorithm described above has successfully been used to control real and simulated robots with DREAM Controllers. In a user study of a search and rescue task with first responders (n=6), we observed people who had never interacted with a multi-touch device or a video game controlling an ATRV-JR's movement and the robot's camera's pan tilt without looking at their hands during their 25 minute runs [8], [10]. In a multi-robot command and control interface, the DREAM Controller provided a means of taking control of one of multiple simulated robots [10].

We are currently fine-tuning our heuristics and investigating others that could be added, as we strive for 100% accuracy, all while allowing the detection of hands when there are more than 5 touches on the screen. However, with its current 97% accuracy rate, the algorithm has shown itself to be effective for controlling robots using the DREAM Controller. Our next steps are to continue development of the SoftOCU to allow for its deployment in the field with real robots.

VIII. Acknowledgments

This research has been funded in part by the National Science Foundation (IIS-0546309) and Microsoft Research. The 22-inch 3M Multi-Touch Display M2256PW monitors were provided courtesy of 3M Touch Systems. The hand detection algorithm and DREAM Controller are patent pending.

TABLE I

HAND AND FINGER RECOGNITION RATES ($\#_{correct}/total$) BY BOARD POSITION FOR 405 HAND PLACEMENTS FROM 62 PARTICIPANTS

		Position 0	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7
Overall	V1	65 of 65 (100%)	42 of 52 (80.8%)	51 of 51 (100%)	49 of 52 (94.2%)	47 of 49 (95.9%)	33 of 39 (84.6%)	41 of 44 (93.2%)	43 of 53 (81.1%)
	V2	64 of 65 (98.5%)	49 of 52 (94.2%)	50 of 51 (98%)	52 of 52 (100%)	47 of 49 (95.9%)	37 of 39 (94.9%)	43 of 44 (97.7%)	51 of 53 (92.2%)
Right overall	V1	35 of 35 (100%)	21 of 25 (84%)	27 of 27 (100%)	26 of 27 (96.3%)	22 of 23 (95.7%)	14 of 19 (73.7%)	22 of 23 (95.7%)	24 of 29 (82.8%)
	V2	35 of 35 (100%)	24 of 25 (96%)	27 of 27 (100%)	27 of 27 (100%)	22 of 23 (95.7%)	18 of 19 (94.7%)	22 of 23 (95.7%)	27 of 29 (93.1%)
Right tips	V1	13 of 13 (100%)	6 of 8 (75%)	9 of 9 (100%)	7 of 7 (100%)	7 of 8 (87.5%)	4 of 6 (66.7%)	7 of 7 (100%)	7 of 10 (70%)
	V2	13 of 13 (100%)	7 of 8 (87.5%)	9 of 9 (100%)	7 of 7 (100%)	7 of 8 (87.5%)	5 of 6 (83.3%)	7 of 7 (100%)	8 of 10 (80%)
Right pads	V1	12 of 12 (100%)	7 of 8 (87.5%)	9 of 9 (100%)	8 of 9 (88.9%)	7 of 7 (100%)	5 of 6 (83.3%)	7 of 7 (100%)	6 of 8 (75%)
	V2	12 of 12 (100%)	8 of 8 (100%)	9 of 9 (100%)	9 of 9 (100%)	7 of 7 (100%)	6 of 6 (100%)	7 of 7 (100%)	8 of 8 (100%)
Right stretch	V1	10 of 10 (100%)	8 of 9 (88.9%)	9 of 9 (100%)	11 of 11 (100%)	8 of 8 (100%)	5 of 7 (71.4%)	8 of 9 (88.9%)	11 of 11 (100%)
	V2	10 of 10 (100%)	9 of 9 (100%)	9 of 9 (100%)	11 of 11 (100%)	8 of 8 (100%)	7 of 7 (100%)	8 of 9 (88.9%)	11 of 11 (100%)
Left overall	V1	30 of 30 (100%)	21 of 25 (84%)	24 of 24 (100%)	23 of 25 (92%)	25 of 26 (96.2%)	19 of 20 (95%)	19 of 21 (90.5%)	19 of 24 (79.2%)
	V2	29 of 30 (96.7%)	25 of 25 (100%)	23 of 24 (95.8%)	25 of 25 (100%)	25 of 26 (96.2%)	19 of 20 (95%)	21 of 21 (100%)	24 of 24 (100%)
Left tips	V1	10 of 10 (100%)	6 of 8 (75%)	8 of 8 (100%)	7 of 9 (77.8%)	8 of 8 (100%)	6 of 6 (100%)	5 of 7 (71.4%)	5 of 8 (62.5%)
	V2	10 of 10 (100%)	7 of 8 (87.5%)	7 of 8 (87.5%)	9 of 9 (100%)	8 of 8 (100%)	6 of 6 (100%)	7 of 7 (100%)	8 of 8 (100%)
Left pads	V1	10 of 10 (100%)	6 of 8 (84%)	8 of 8 (100%)	7 of 7 (100%)	8 of 9 (88.9%)	6 of 6 (100%)	7 of 7 (100%)	7 of 8 (87.5%)
	V2	10 of 10 (100%)	8 of 8 (100%)	8 of 8 (100%)	7 of 7 (100%)	8 of 9 (88.9%)	6 of 6 (100%)	7 of 7 (100%)	8 of 8 (100%)
Left stretch	V1	10 of 10 (100%)	9 of 11 (81.8%)	8 of 8 (100%)	9 of 9 (100%)	9 of 9 (100%)	7 of 8 (87.5%)	7 of 7 (100%)	7 of 8 (87.5%)
	V2	9 of 10 (90%)	10 of 11 (90.9%)	8 of 8 (100%)	9 of 9 (100%)	9 of 9 (100%)	7 of 8 (87.5%)	7 of 7 (100%)	8 of 8 (100%)

REFERENCES

[1] C. Snyder, *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Morgan Kaufmann Publishers, 2003.

[2] J. Wobbrock, M. Morris, and A. Wilson, "User-defined Gestures for Surface Computing," in *Conf. on Human Factors in Computing Sys.*, 2009.

[3] M. Micire, J. Drury, B. Keyes, and H. Yanco, "Multi-Touch Interaction for Robot Control," in *Proceedings of the Intl. Conf. on Intelligent User Interfaces*, 2009.

[4] A. Agarwal, S. Izadi, M. Chandraker, and A. Blake, "High Precision Multi-touch Sensing on Surfaces Using Overhead Cameras," in *2^{nd} IEEE Intl. Workshop on Horizontal Interactive Human-Computer Sys. (Tabletop)*, 2007, pp. 197–200.

[5] S. Izadi, A. Agarwal, A. Criminisi, J. Winn, A. Blake, and Fitzgibbon, "C-Slate: A Multi-Touch and Object Recognition System for Remote Collaboration using Horizontal Surfaces," *2^{nd} IEEE Intl. Workshop on Horizontal Interactive Human-Computing Sys. (Tabletop)*, vol. 7, 2007.

[6] F. Wang, X. Cao, X. Ren, and P. Irani, "Detecting and Leveraging Finger Orientation for Interaction with Direct-touch Surfaces," in *Proc. of the 22^{nd} Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2009, pp. 23–32.

[7] C. Dang, M. Straub, and E. André, "Hand Distinction for Multi-touch Tabletop Interaction," in *Proc. of the ACM Intl. Conf. on Interactive Tabletops and Surfaces*, 2009, pp. 101–108.

[8] J. Matejka, T. Grossman, J. Lo, and G. Fitzmaurice, "The Design and Evaluation of Multi-finger Mouse Emulation Techniques," in *Proc. of the 27^{th} Intl. Conf. on Human Factors in Computing Sys.*, 2009, pp. 1073–1082.

[9] M. Micire, M. Desai, J. Drury, E. McCann, A. Norton, K. Tsui, and H. Yanco, "Design and Validation of Two-Handed Multi-Touch Tabletop Controllers for Robot Teleoperation," in *Proc. of the Intl. Conf. on Intelligent User Interfaces*, 2011.

[10] J. Nielsen, *Usability Engineering*. San Diego, CA: Academic Press, 1993.

[11] M. J. Micire, "Multi-Touch Interaction for Robot Command and Control," Ph.D. dissertation, Computer Science Department, University of Massachusetts Lowell, December 2010.

[12] D. Saffer, *Designing Gestural Interfaces: Touchscreens and Interactive Devices*. O'Reilly Media, Nov 26, 2008.

[13] C. Hager-Ross and M. Schieber, "Quantifying the Independence of Human Finger Movements: Comparisons of Digits, Hands, and Movement Frequencies," *J. of Neuroscience*, vol. 20, no. 22, p. 8542, 2000.

[14] M. Nielsen, M. Störring, T. Moeslund, and E. Granum, "A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for Man-machine Interaction," Aalborg University, Tech. Rep., 2003, report no. CVMT 03-01. http://www.cvmt.dk/~fgnet/docs/fgnet_techreport.pdf.

[15] H. Dreyfuss, *Designing for People*. Allworth Press, 2003.