A Multisensor Technique for Gesture Recognition Through Intelligent Skeletal Pose Analysis

Nathaniel Rossol, Irene Cheng, *Senior Member, IEEE*, and Anup Basu, *Senior Member, IEEE*

***Abstract*—Recent advances in smart sensor technology and computer vision techniques have made the tracking of unmarked human hand and finger movements possible with high accuracy and at sampling rates of over 120 Hz. However, these new sensors also present challenges for real-time gesture recognition due to the frequent occlusion of fingers by other parts of the hand. We present a novel multisensor technique that improves the pose estimation accuracy during real-time computer vision gesture recognition. A classifier is trained offline, using a premeasured artificial hand, to learn which hand positions and orientations are likely to be associated with higher pose estimation error. During run-time, our algorithm uses the prebuilt classifier to select the best sensor-generated skeletal pose at each time step, which leads to a fused sequence of optimal poses over time. The artificial hand used to establish the ground truth is configured in a number of commonly used hand poses such as pinches and taps. Experimental results demonstrate that this new technique can reduce total pose estimation error by over 30% compared with using a single sensor, while still maintaining real-time performance. Our evaluations also demonstrate that our approach significantly outperforms many other alternative approaches such as weighted averaging of hand poses. An analysis of our classifier performance shows that the offline training time is insignificant, and our configuration achieves about 90.8% optimality for the dataset used. Our method effectively increases the robustness of touchless display interactions, especially in high-occlusion situations by analyzing skeletal poses from multiple views.**

***Index Terms*—Depth sensors, gesture recognition, multisensor, occlusion, pose estimation, user evaluation.**

I. Introduction

INTERACTING with computer interfaces through mid-air hand gestures is emerging as an intuitive and effective alternative to traditional touch-screen interfaces. For example, interacting with medical displays via touch screens or a traditional mouse and keyboard can present a major problem for medical professionals wishing to keep equipment sterile. Even in small clinical settings, examinations may often use fluids, such as ultrasound conductive gel, which users do not want to spread onto a physical interface through touch [1].

Mid-air hand gestures allow individuals to interact freely with computer interfaces without keyboard, mouse, or screen contact. Instead, hand movements are tracked and interpreted through computer vision (CV) techniques [2]–[4]. This paper is focused on the domain of markerless mid-air hand gestures (i.e., gestures that are tracked with CV techniques alone and do not require the placement of markers or tracking devices on the user's hands). Markerless approaches are much more desirable among users because they allow for immediate interaction with a computer interface. Time and resources are not wasted on additional setup or calibration steps [5].

A. Problem

Previous work in markerless CV hand tracking has made use of color and/or depth cameras (such as the Microsoft Kinect [6]) in order to analyze either static or dynamic hand gestures in real time. Using raw depth/color data, features such as hand and finger positions/orientations can be extracted, from which an estimate of the hand's pose can be determined. The main drawback of these past approaches is a large amount of noise associated with the computed 3-D fingertip positions. In addition, low sampling rates make it difficult to track quick hand movements due to motion blur [7]. With the recent hardware advances in new CV hand-tracking sensor systems (such as the Leap Motion Sensor), millimeter-level precision can be achieved for tracking a fully articulated hand skeleton at sampling rates of over 120 frames/s [8]. However, these latest sensors still have low pose estimation accuracy due to occlusion. These situations frequently occur when the palm is not directly facing the camera, or when performing certain gestures, such as pinches, where one finger can be blocked by another. These problems can disrupt accurate gesture interpretation and lead to unintended computer operations.

B. Proposed Solution

We introduce a novel technique to improve hand pose estimation accuracy when using smart depth sensor technology for tracking hand poses. In particular, our technique addresses the issue of occlusion by using pose estimations from multiple sensors placed at different viewing angles. One of the primary advantages of our approach is to avoid fusing sensor data at the 3-D depth-map level, which is not available from all modern sensors such as the Leap Motion Sensor. Instead, we achieve wider flexibility by intelligently analyzing each sensor system's independently derived skeletal pose estimations. A key challenge that makes skeletal pose fusion especially difficult is the tendency of the underlying pose estimation algorithms being trapped in local minimas that can be substantially different from each other. This explains why most classical sensor fusion techniques, such as the Kalman filter [9], are ineffective

Manuscript received February 4, 2015; revised May 5, 2015 and July 12, 2015; accepted July 23, 2015. This work was supported by the Natural Sciences and Engineering Research Council of Canada and Alberta Innovates Technology Futures. This paper was recommended by Associate Editor J. Wachs.

The authors are with the Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada (e-mail: nrossol@ualberta.ca; locheng@ualberta.ca; basu@ualberta.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2015.2467212

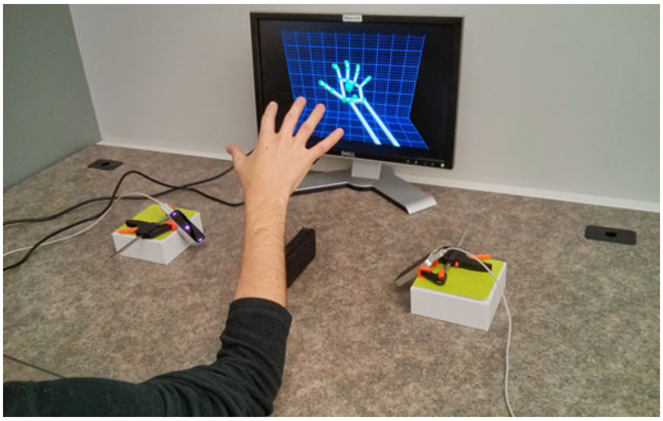


Fig. 1. Example setup used in our lab to capture a single hand pose from two different viewing angles.

in this problem domain, because there is no useful difference in the generated noise profiles to make a meaningful quality evaluation of the sensor data. To address this issue, we present a more robust strategy, which demonstrates that by selecting an appropriately designed subset of the skeletal pose estimation parameters, we can build an offline model. The offline model can then be used in real time to intelligently select pose estimations, while still running at over 120 frames/s. In our experimental prototype involving dual sensors (see Fig. 1), we analyze the pose estimation accuracy from different angles for a number of hand gestures typically used to control displays (e.g., pinch, tap, and open hand). In this context, we demonstrate that we are able to achieve a 31.5% reduction in pose estimation error compared with using only a single sensor. We are able to effectively eliminate the false hand poses that interfere with accurate gesture recognition.

Our contributions are summarized as follows.

1) We improve pose estimation accuracy of state-of-the-art hand-tracking systems through a novel technique for analyzing skeletal hand poses from multiple sensors.
2) We experimentally demonstrate how the improved pose estimations can have a meaningful improvement on recognizing gestures used for controlling display interfaces.

II. Related Work

A. Real-Time Hand Pose Estimation

CV-based hand gesture recognition can be broken down into two main categories: 1) model-based approaches concerned with pose estimation; and 2) appearance-based approaches [10]. Model-based approaches involve determining and tracking the entire articulated pose of the hand as it moves, including the 3-D position and orientation of the wrist, and the deflection angles of every joint. On the other hand, appearance-based approaches use other characteristics in the captured images, such as the silhouette, contour, color, area, and pixel flow, to predict the intended hand gesture [11], [12]. Despite the benefit of knowing the full pose of a moving hand, most real-time systems make use of indirect cues from appearance-based approaches due to processing time constraints [13]. Early studies have shown that accurate CV model-based hand pose estimation algorithms are too computationally expensive to run in real time [6]. Fortunately, recent hardware advances have helped deliver low-cost commercial depth sensors, which support real-time and more precise hand pose estimation.

In 2011, Oikonomidis *et al.* [6] successfully developed an accurate model-based approach for tracking the 3-D position, orientation, and full articulation of a hand at 15 Hz on high-end hardware. Their approach used a combination of video and depth images from a Microsoft Kinect sensor as input, which were processed by a modified particle swarm optimization algorithm. Although still computationally intensive, the 15-Hz frame rate can be achieved using only a single computer, making use of a highly parallelized GPU implementation. The authors then expanded their approach to include interactions between two hands [14].

In 2013, Keskin, *et al.* [15] proposed a novel approach for real-time hand pose recognition using random decision forests on a synthetically generated dataset of hand poses. Their approach is similar to the approach used by Microsoft Research on Kinect to create the real-time skeletal pose recognition system [16]. By applying their classifier to label each depth pixel, and then mean-shift to identify the position of each part of the hand, full pose estimation is achieved. The classifier is able to run at a rate of up to 30 Hz. Although the authors claimed that the technique should theoretically be able to classify arbitrary hand poses given enough training data, classification of only a few discrete poses (namely, hand poses related to American Sign Language) was presented. This limitation is also present in the real-time discrete hand pose system proposed by Romero *et al.* [17]. In contrast, we are interested in the continuous space of all possible hand poses. Our goal is to maximize flexibility and future extensibility of our system.

A drawback of traditional approaches is that they are not able to robustly capture small precise gestures (like quick subtle finger taps), due to their low sampling rates. It is necessary to estimate human hand poses at high sampling rates because of the rapid motions of the hand. Previous work has reported that human hands can reach speeds of up to 5 m/s and the wrist can reach rotational speeds of up to 300 °/s during normal gestures [13]. Hand poses between successive frames become increasingly uncorrelated with each other as the hand moves faster [13]. The combination of high movement speeds and low sampling rates often leads to inaccurate gesture recognition. More accurate results can be obtained from advanced sensors with higher sampling rates. For this reason, we used the leap motion sensor [8] to test our method.

The leap motion sensor is an example of a new generation of CV-based sensors that provides real-time hand-tracking and pose estimation. Unlike past Infrared Red (IR) pattern light depth sensors (such as the first generation Microsoft Kinect), or time-of-flight sensors (such as the Soft Kinectic DepthSense Camera), the leap motion sensor is able to provide much higher 3-D positional accuracy for hands and fingertips (better than 1 mm of precision). Its sampling rate exceeds 120 frames/s [8]. However, the sensor is unable to provide a full high-resolution

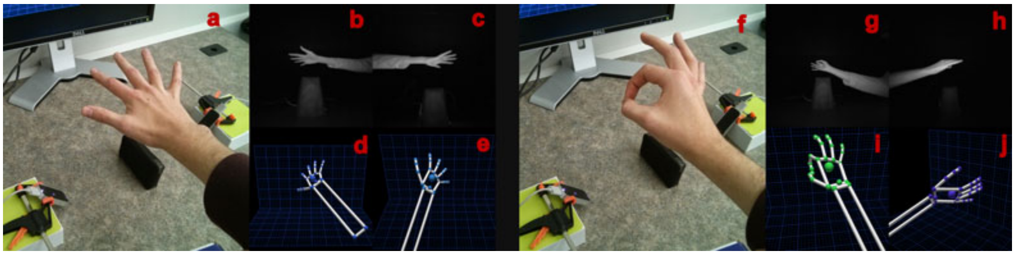


Fig. 2. Hand poses and sensors are labeled from (a) to (j). In the pair of images on the left, the open hand pose (a) is visible by both sensors as (b) and (c). The sensors give a similar pose estimation (d) and (e) in their local sensor coordinate spaces. In the pair of images on the right, a pinch pose (f) is tested. The sensor on the left has a fairly good view (g) and gives an estimate of (i). However, this pose is mostly occluded and seen as (h) from the right side sensor, which leads to an estimate of (j). Note that the pinch pose has become an open hand, which disagrees with the left sensor.

depth map due to USB bandwidth limitations. In addition, its maximum effective tracking distance is only around 50 cm from the device.

B. Sensor Fusion

High-occlusion situations impose a major challenge for analyzing hand poses. As previously mentioned, many skeletal hand pose estimation techniques follow particle-filter approaches, which can trap pose estimation algorithms in a local minimum due to inadequate visible data to resolve ambiguity. The pose can be incorrectly interpreted for a considerable amount of time, or even indefinitely if the user's hand continues to stay in a static pose. Typical real-time sensor fusion techniques, such as the Kalman filter [9], are unsuitable in this context. This is because the similar noise profiles of correct and incorrect hand poses will result in equal evaluation weighting, making it impossible to identify the correct pose [18]. While using finger motion, such as the angular velocities of finger joints, to predict future positions may be helpful, this method is not feasible for many applications given rapid finger movements.

Regardless of the underlying hand pose recognition technique, or sensor used, occlusion is a major problem when using a single vision-based sensor. We propose fusing data from multiple sensors placed at different viewing angles, and analyzing the skeletal poses directly instead of examining the depth maps, because the latter may not always be available. This is the case with the leap motion sensor, which does not generate any depth maps or 3-D point clouds.

We tested a two-sensor setup capturing a pose from two different viewing angles. When there is a disagreement between the sensors at a particular time step due to possible occlusion blocking one of the views, our computation model evaluates the different estimates and chooses the one that best fits the continuous stream of skeleton poses. Fig. 2 shows examples of agreement and disagreement between the two sensors. Our technique is intended to generate a fused sequence of optimal poses over time, and we are able to demonstrate (in Section IV) that this approach is more accurate than the alternative of fusing the sensor images at each time-step.

C. Gesture-Controlled Displays

Over the past decades, there have been many studies on hand-tracking and touchless displays. Real-time hand tracking is important in many applications, including medical ones. A primary motivation in healthcare is reducing spreading biological contamination by avoiding touching a device, and the resulting time/cost savings from reduced sterilization [3].

One of the earlier works in CV-based gesture control of medical displays was completed by Grange *et al.*, in 2004 [2]. Their system-controlled computer mouse movements in an operating room display via hand gestures. Stereo color cameras and a combination of static background subtraction and pixel color thresholding were used to track the user's hand positions in a 3-D space. As the system did not track fingers, virtual mouse clicks were performed by either pushing the hand forward 20 cm, or holding it absolutely still for several seconds. Similarly, the Gestix's system developed in 2008 [4] used a nearly identical approach but did not control a virtual cursor. Instead, the authors used communicative gestures (such as hand swipes or circular motions) to perform various tasks. As is typical in appearance-based approaches that use color information to segment the user's hands, both of these approaches are vulnerable to segmentation errors caused by changes in illumination, shadows, or dynamic backgrounds.

In 2011, Gallo *et al.* [19] proposed a Kinect-based interface for visualizing medical images in a sterile surgery room. The user was required to be standing and using both hands for interaction. The system tracked the 3-D position of both hands at 30 Hz and recognized the hand as either being in the open or closed state. As with the previous works, the inability to track the finger positions means that the gesture requires large hand movements for recognition purposes, which can cause the user physical fatigue over time [20]. Another limitation of the system is that their method requires two hands to be free for operation, which is not possible when one hand is used to hold clinical tools.

Our system can be used with either one or both hands and is designed for gestures that are commonly adopted by touch-screen interfaces.

III. Implementation

Our multisensor skeletal pose estimation approach is composed of the following steps.

1) First, we use a trained support vector machine (SVM) [21] model to intelligently determine the optimal pose estimation from an array of sensors. We build this model offline (only once) with a training set, which uses a

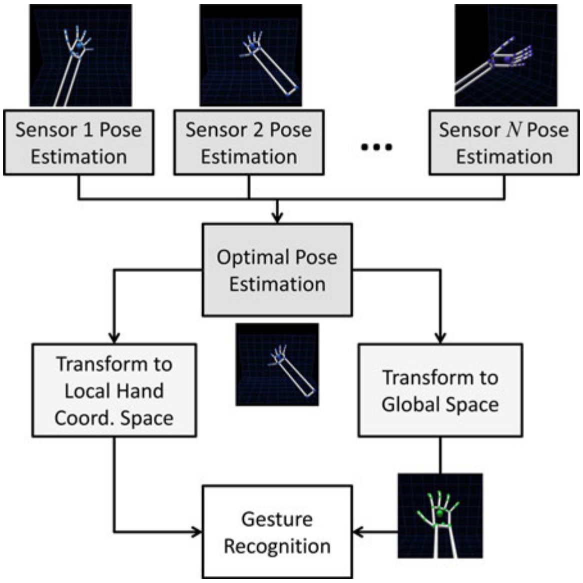


Fig. 3. Overview of our real-time execution steps.

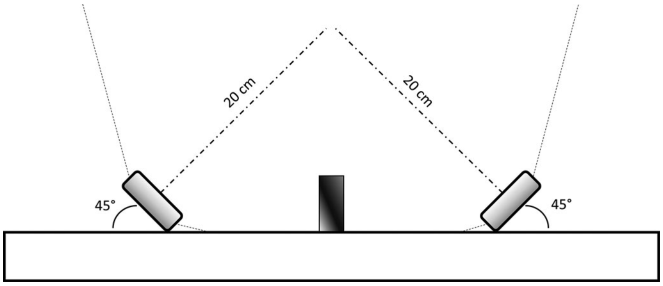


Fig. 4. Diagram of our two-sensor setup. A small nonreflecting object is placed between the sensors to prevent a direct line of sight.

feature vector composed of a subset of each sensor's output.

2) Next, we convert each sensor's pose output into a global coordinate system so that the poses of all of the fingers are represented in a single unified space. Likewise, we also keep the positions of all of the fingers in a local hand coordinate system, which provides key information for the pose estimation model.
3) Finally, the local hand pose information and the global hand pose information are input into our gesture recognition model so that dynamic and static hand gestures can be tracked.

An overview of these steps is shown in Fig. 3.

A. Sensor Array Setup

The goal of our implementation is to demonstrate the feasibility of our algorithm. We tested our approach with a two-sensor setup. As shown in Figs. 1 and 4, our setup involves two leap motion sensors aligned at a 45° angle to the table surface they are placed on. This angle is chosen to make the sensor view angles orthogonal, to optimize the amount of unique information available to each sensor during situations of high occlusion. The point at which the center of the field of view of both sensors intersects was set at 20 cm for our experiment, following the default precalibrated interaction height for the leap motion sensor.

One consideration when using multiple leap motion sensors for our configuration is that the IR (infrared) projectors of one sensor can shine directly into the IR cameras of the other, which can generate noisy data. This can be solved by positioning the sensors outside the 150° field of view of each other, or by placing a small nonreflective object in-between the sensors, to prevent a direct line of sight. We adopted the latter approach in our implementation. Based on our test results, even if a pair of leap motion sensors shine directly into one another, the additional noise in the pose estimation was insignificant provided the automatic infrared light compensation feature in the sensor configuration was disabled.

B. Selecting Optimal Pose Estimations

We define the amount of pose estimation error as the sum of the Euclidean distances (in millimeters) of each fingertip from its ground truth position. The positions are expressed locally relative to the reported palm position and normal of the hand (provided directly by the sensor data in our setup). That is, if $\mathbf{f_i}$ is a 3-D vector representing the position of the fingertip of the $\mathbf{i}$th finger on a hand, and $\mathbf{g_i}$ is the actual ground truth position, we then define the error of a hand pose estimation as

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

Given a set of pose estimations from a sensor array (two sensors with two independent estimations in our prototype), the goal at each time step is to select the single pose estimation that has minimum error E. The main issue is that it is impossible to track the difference in noise levels between accurate and highly inaccurate pose estimations. As explained previously, stable inaccurate pose estimations are a common artefact of the traditional particle filter algorithms, which makes it difficult to decide which sensor's reading is more trustworthy.

The novelty of our approach lies in intelligently determining which sensor is likely reporting the most accurate pose estimation. Specifically, we observed that even in situations with high occlusion and pose estimation error, the tracking of the palm position and orientation often remained accurate. We exploit this observation and build an SVM model that learns which hand positions and orientations are likely to be associated with higher pose estimation error. At run-time, our algorithm computes a feature vector based on each sensor's reported hand position/orientation in the local sensor space. This feature vector is then run through the prebuilt classifier in order to predict which sensor is likely providing the best pose estimation. The selected pose is then passed along for subsequent processing and gesture recognition.

For our two-sensor setup, each sensor's feature vector is composed of the following 12 floating point values per pose estimation:

1) Three (X, Y, and Z) values of the palm position relative to the sensor.
2) Three (X, Y, and Z) values of the palm plane normal relative to the sensor.
3) Three (X, Y, and Z) values of the "forward" direction of the hand (i.e., the direction where the fingers point to when extended).
4) The local "Roll" rotation of the hand.
5) The dot product between the palm normal and the direction the sensor is facing.
6) Sensor confidence estimation. This value is a floating point between 0 and 1 that is generated by an internal proprietary part of the leap motion API. Its limitations when used on its own are shown in Section IV.

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

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

As is typical of an SVM training approach, the time required for training the offline model depends on the number of training samples and number of features used. In our implemented prototype, the training time for 108 data points with 12 features each was less than 0.1 s when executing on a regular computer.

C. Reprojecting Into Global and Local Spaces

After a hand pose is selected from the sensor array, it must be converted into a unified global space to interpret meaningful gestures or interactions. The matrix used to determine the hand pose in the global space can be defined manually if the placement of sensors is known precisely enough, such as in our proposed two-sensor setup. In general, for an array of multiple sensors, the user can use several static finger positions visible from each sensor to build a set of 3-D point clouds, and then apply an iterative closest point algorithm [22] to compute the specific matrix required to project the points from each sensor's local coordinate system into the global coordinate system.

Similarly, we also compute a projection into a local hand coordinate space to help in gesture recognition. If we consider a hand with a normalized palm direction vector $\hat{\mathbf{h}}_N$ (i.e., normal to the plane of the palm and in the same direction as the palm), and a normalized orthogonal forward vector $\hat{\mathbf{h}}_F$, we can compute an additional basis vector $\hat{\mathbf{h}}_C$ for the local hand coordinate system as the cross product of these two:

$$\hat{\mathbf{h}}_C = \hat{\mathbf{h}}_N \times \hat{\mathbf{h}}_F. \tag{2}$$

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

$$\mathbf{d}_M = \frac{\sum_{i=1}^{n} \hat{\mathbf{d}}_i}{n}. \tag{3}$$

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

$$\hat{\mathbf{h}}_F = \frac{\mathbf{d}_M - (\hat{\mathbf{h}}_N \cdot \mathbf{d}_M)\hat{\mathbf{h}}_N}{||\mathbf{d}_M - (\hat{\mathbf{h}}_N \cdot \mathbf{d}_M)\hat{\mathbf{h}}_N||}. \tag{4}$$

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

$$M = \begin{bmatrix} \mathbf{h}_{\hat{\mathbf{C}}_\mathrm{x}} & -\mathbf{h}_{\hat{\mathbf{N}}_\mathrm{x}} & -\mathbf{h}_{\hat{\mathbf{F}}_\mathrm{x}} & T_x \\ \mathbf{h}_{\hat{\mathbf{C}}_\mathrm{y}} & -\mathbf{h}_{\hat{\mathbf{N}}_\mathrm{y}} & -\mathbf{h}_{\hat{\mathbf{F}}_\mathrm{y}} & T_y \\ \mathbf{h}_{\hat{\mathbf{C}}_\mathrm{z}} & -\mathbf{h}_{\hat{\mathbf{N}}_\mathrm{z}} & -\mathbf{h}_{\hat{\mathbf{F}}_\mathrm{z}} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}. \tag{5}$$

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

The transformation is a 4×4 homogeneous matrix. This means that 3-D points are augmented with a "1" at the end before multiplication, and direction vectors are augmented with a "0."

D. Gesture Detection

Finally, after the hand pose is computed in the local and global spaces, we perform gesture recognition on the computed poses. We use gestures which are intuitive to the users. The interactions are based on a combination of three basic movements: air-taps, pinches, and dragging (or swiping). Finger air-taps are analogous to clicking mouse buttons or tapping a touch screen with the finger, but in mid-air. In our gesture interfaces, this is used to select items. The air-tap was chosen because it can be recognized robustly, and previous studies have demonstrated that most users find the gesture easy to learn and remember [23], [24]. Similarly, pinch gestures in the air are essentially the same

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

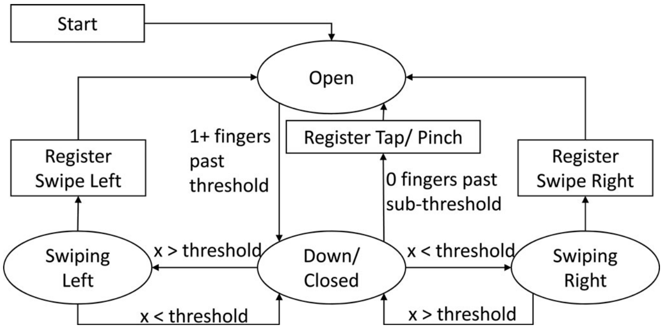


Fig. 5. Overview of the state transitions of our gesture interface design.

as those used on touch screen for zooming. Dragging (or swiping) refers to moving the hand, while at least one of the fingers is in the tap-down state (i.e., performing the first half of an air-tap, but halting before raising the finger). This is analogous to dragging file icons on a computer interface while having a mouse button down, or dragging objects across a touch-screen display with a fingertip, or swiping between pages displayed on a tablet computer. Our framework also allows pinch-dragging.

In past studies, hidden Markov models have been effective for gesture recognition in appearance-based systems [25]. In our case, however, because the actual finger positions are known with high accuracy, we use the current state of the fingers directly in a finite-state machine model, without the need for indirectly modeling hidden states. The result is a computationally inexpensive direct analysis of the finger states. Fig. 5 shows a graph summarizing the state transitions of our gesture interface design.

1) Tap and Pinch Recognition: Using our local coordinate system, it is easy to suggest that an air-tap is performed when a finger moves to a local y value exceeding a threshold value. However, observations show that false positives can occur when the user's hand is in a relaxed state, probably due to fatigue over time as shown in Fig. 6(c) and (d).

Thus, we adopt an adaptive approach, taking other fingers into consideration, to compute the threshold y value of the downward distance a finger must move in order to be classified as an air-tap. It is computed using the following equation:

$$T_f = D_T(1 + W_T m_T) \qquad (6)$$

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

Once a finger has been recognized as being in the tapped state, i.e., having passed below T_f, we require that it returns to the palm plane through a different upper threshold value U_f, in order to be recognized as the untapped motion, where

$$U_f = c(T_f),\ 0 < c < 1. \qquad (7)$$

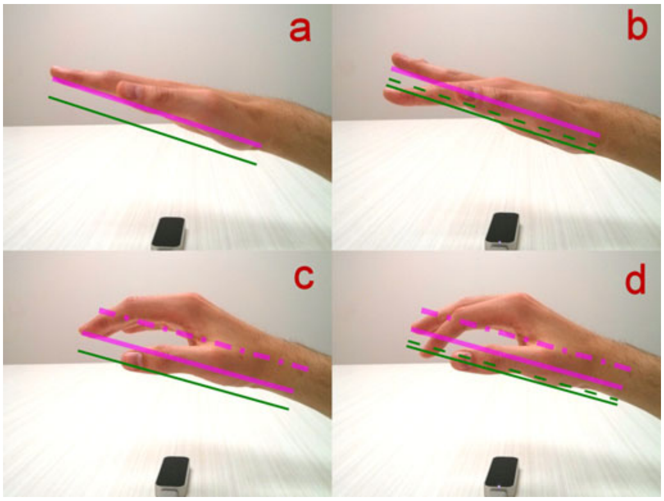


Fig. 6. Overview of the tap recognition system, which illustrates the importance of defining dynamic thresholds. The dashed magenta line indicates the palm plane as reported by the sensor. The solid magenta line indicates the side view of the offset plane parallel to the palm plane, but offset according to the average displacement of the fingertips relative to the palm plane. The solid green line indicates the threshold that must be exceeded to enter the tap down state. The dashed green line indicates the threshold that the finger must be above to return to the Open state. (a) Ideal case where the user's fingertips are mostly lying in the palm plane. (b) Tap gesture is easy to recognize. (c) More realistic case where the user has taken a more relaxed hand posture over time. (d) Tap gesture is not easy to recognize (bottom right).

Fig. 6 illustrates the tap recognition corresponding to the ideal and relaxed hand states.

A similar concept is used for pinch recognition, except that the computational complexity of the palm plane and neighboring fingers is reduced. For pinch recognition, we define two thresholds: an enter threshold, and a larger release threshold. When the Euclidean distance between the index finger and thumb drops below the enter threshold (e.g., 15 mm), a pinch gesture will be recognized as "Started." When the distance between the two pinching fingers exceeds the exit threshold (e.g., 25 mm), the pinch is recognized as "Ended."

2) Drag or Swipe Recognition: Drag gestures are recognized as hand movements that occur when one or more fingers are in the down position of an air-tap, or a pinch. Unlike interaction on a touch screen, where the user drags virtual items according to the position of their finger tip, in our model, items are first selected by the user's fingers, but then dragged according to the hand position. This is due to the lack of haptic feedback in mid-air gestures, which makes it difficult to distinguish the movements of communicative tap gestures from the movements used to manipulate the position of a virtual object. Our recognition technique is similar to using mouse buttons for selections and hand palm movements (i.e., mouse movements) to move objects on a computer screen.

IV. Evaluation With Ground Truth

In our experiments, ground truth values for hand poses were determined by using an artificial hand model that was manually placed into fixed and known poses. Since there are a large number of usable hand poses that could be created, we focus

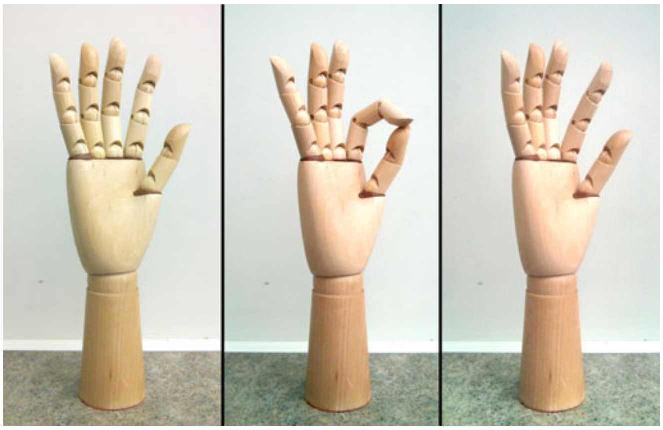


Fig. 7. Three ground-truth hand poses tracked by our system.

on tracking three hand poses that can replace certain commands commonly used on touch-screen interfaces. These three hand poses are the following:

1) an open hand;
2) a pinch gesture;
3) a tap gesture.

These gestures are shown in Fig. 7.

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

In order to investigate the ability of our multisensor technique to track these hand poses when the hand is moving, we fixed the hand position in the middle of the field of view of both sensors (placed 20 cm away) and slowly rotated the artificial hand counterclockwise along the axis of the wrist as shown in Fig. 8. The rotation was paused every 10° in order to record a measurement from each sensor. This resulted in 36 pairs of pose estimations per gesture and 108 pairs of pose estimations in total.

We then computed the pose estimation error of each sensor for all 108 cases. To train our computational model, we constructed all 108 feature vectors and labeled every case with the sensor that produced the lower pose estimation error. Note that the average pose estimation performance of both sensors were identical because both sensors eventually saw all the exact same hand poses from the same distance and rotation angle (but not at the same time). We evaluated the performance of our model using a tenfold cross validation across all 108 data points, which is a standard approach commonly used by classification techniques.

A. *Performance Analysis*

Overall, the experimental results indicated that our approach reduced the total pose estimation error by 31.5% compared with using only one sensor. We summarize the performance comparisons in Tables I and II.

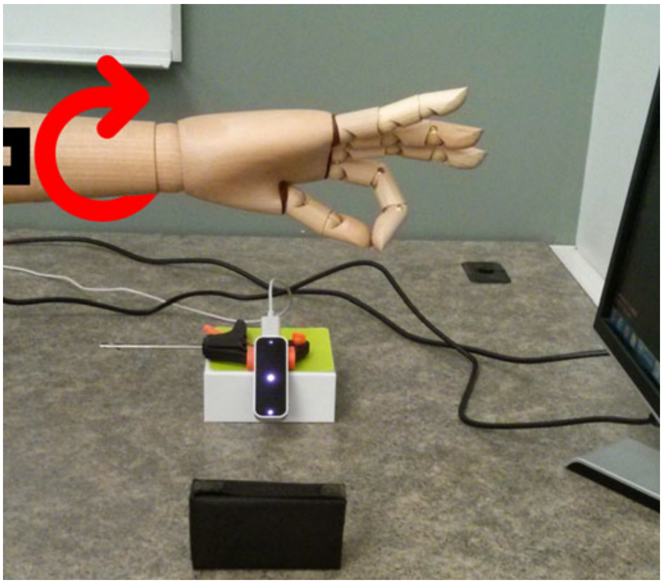


Fig. 8. Artificial hand model was mounted such that it was fixed in space relative to the sensors and rotated along the axis of the wrist.

TABLE I
EXPERIMENTAL RESULTS

Technique	Finger	Estimation Err. (mm)	Std. Err
Single Sensor	Thumb	14.20	1.22
	1st Finger	18.40	1.87
	2nd Finger	15.51	1.55
	3rd Finger	16.29	1.08
	4th Finger	12.98	1.06
	Mean	**15.48**	1.12
Averaging	Thumb	11.39	0.78
	1st Finger	16.22	1.19
	2nd Finger	13.27	1.00
	3rd Finger	13.79	0.67
	4th Finger	10.83	0.67
	Mean	**13.10**	0.69
Sensor Confidence	Thumb	11.23	0.86
	1st Finger	14.26	1.49
	2nd Finger	12.61	0.97
	3rd Finger	13.14	0.94
	4th Finger	10.24	0.83
	Mean	**12.29**	0.85
Weighted Fusion	Thumb	9.56	1.00
	1st Finger	13.44	1.61
	2nd Finger	12.89	1.54
	3rd Finger	12.56	0.91
	4th Finger	9.88	0.78
	Mean	**11.66**	1.01
Our Approach	Thumb	9.53	0.85
	1st Finger	10.09	0.97
	2nd Finger	10.74	0.77
	3rd Finger	12.49	0.88
	4th Finger	10.14	0.84
	Mean	**10.60**	0.74

The average pose estimation error (in millimeters) is calculated from the sum of the Euclidean distances (in millimeters) of each fingertip from its ground truth position [see (1)] divided by the number of fingers. Using this metric, the worst-case and best-case performances were found to be 21.45 and 9.5 mm,

TABLE II
OVERALL PERFORMANCE COMPARISON

Technique	Mean Pose Estimation Error (mm)	% Optimal
Worst Case	21.45	0%
Best Case	9.50	100%
Single Sensor	15.48	50%
Averaging	13.10	69.9%
Sensor Confidence	12.29	76.6%
Weighted Fusion	11.66	81.9%
Our Approach	10.60	90.8%

respectively (see Table II). The worst case represents the theoretical worst average pose estimation error that could result from our approach. For it to occur, our classifier would need to incorrectly choose the sensor with the worse pose estimation every time (i.e., have 0% accuracy). Similarly, the best case represents the result if our approach consistently chose the sensor with the smaller pose estimation error. In this context, our approach is able to achieve an optimal accuracy of 90.8% on the average.

B. Comparison with Alternative Approaches

The result obtained from a single sensor shown in Tables I and II represents the average amount of pose estimation error. Given our symmetric setup, it is also the amount of pose estimation error expected if we choose randomly between the two sensors. Although the mean sensor pose estimation error was approximately 15.48 mm, there was quite a large variation in estimation error across the dataset with a maximum recorded error of 114.24 mm, and the minimum recorded error of less than 0.07 mm. When the pose estimation error exceeded 30 mm, the sensor was producing a pose estimation that was substantially different from the ground truth. There were about 22 such cases in our dataset of 108. Thus, we conclude that occlusion can make the pose estimation performance of a single sensor unreliable.

We also analyzed the performance of using unweighted averaging to fuse pose estimations. That is, for each finger in the local hand coordinate space, the final pose is the midpoint between the two sensors' estimates. This technique generates results similar to those from a Kalman filter. Based on our experiments, the method can help reduce the errors from infrequent poor pose estimations. A drawback is that if the system enters a state where the user's hand pose is constantly at a poor viewing angle for one sensor, the averaging will increasingly reduce the overall pose estimation accuracy.

Another comparison was with using the leap motion sensor's self-reported confidence score. In this measurement, the sensor with the higher reading is selected based on their reported confidences using a scale of 0 to 1. This approach showed approximately 76% accuracy, suggesting the feasibility of including this parameter in the training model. However, it is ineffective by itself in comparison with our approach, which has better performance by taking the palm position and orientation into account. Furthermore, not all depth sensors provide self-reported confidence scores.

Our technique aims at generating a fused sequence of optimal poses over time. We also analyze the "Weighted Fusion" method, which fuses the sensor output at each time step. In this approach, we fit a logistic regression model to the SVM outputs in order to produce probability estimates indicating how confident the model was with each selection. We then used these values directly to compute a combined weighted average of the hand poses. The results showed that the weighted fusion method did not perform as well as our proposed approach. For example, in cases of high occlusion, the weighted fusion method can have hand pose estimations that are well over 10 cm away from the ground truth, while our proposed method has higher accuracy (see Tables I and II).

Our analysis using a paired one-tailed t-test showed that there was a significant difference ($\alpha = 0.05$) in the scores for our approach (mean = 10.60, std. deviation = 7.65) compared with using a single sensor (mean = 15.48, std. deviation = 11.65); $t = 5.15$, $p < 0.0001$. Similarly, there was a significant difference in our approach compared with using unweighted averaging (mean = 13.10, std. deviation = 7.19); $t = 3.93$, $p < 0.0001$ and also between using our approach compared with using the sensor confidence (mean = 12.29, std. deviation = 8.82); $t = 2.88$, $p = 0.0024$.

Additionally, there was a significant difference between using our weighted fusion approach (mean = 11.66, std. deviation = 10.52) compared with using a single sensor; $t = 3.94$, $p < 0.0001$, and also between using our weighted fusion approach in comparison with using unweighted averaging; $t = 2.15$, $p = 0.0171$. However, the weighted fusion approach did not show a significant difference from the sensor confidence approach; $t = 0.74$, $p = 0.231$. In light of this, we recommend the use of our original approach for most applications.

C. Analysis of Classification Performance

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

A large difference in the pose estimation error between the two sensors for a data point typically indicates that the sensors had a large disagreement between each other, and that likely one of them (but perhaps both) was producing a very low quality pose estimation. When the difference between the two sensors was low, it indicates that both sensors were likely producing very similar pose estimations, and thus, the choice of which sensor to use was less important. In our analysis, we computed the difference in sensor pose estimation errors for all our data points. The corresponding distributions are shown in Fig. 9. Note that 1) instances having a higher median (difference in pose estimation error) are correctly classified, and 2) cases with the worst outliers are also classified correctly.

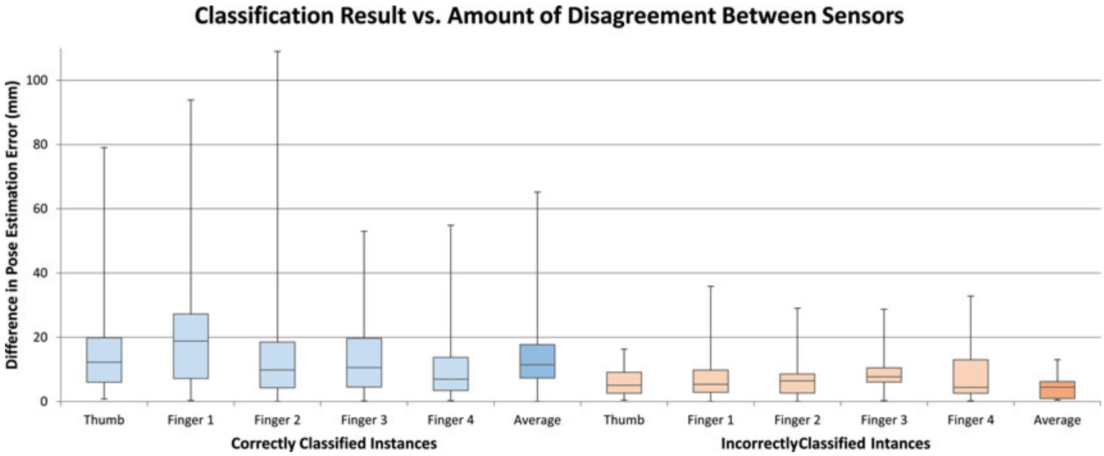


Fig. 9. This plot shows the distribution of the level of disagreement between the sensors for correctly and incorrectly classified instances. Note that when disagreement between the two sensors is high (probably due to occlusion), our approach was able to classify correctly (left half of plot). When the data point was incorrectly classified, the difference between the sensor measurements tended to be small (right half of plot), and thus, it did not add significant error by choosing one sensor or the other. This shows that our approach can achieve high overall accuracy.

Our analysis shows that for the correctly classified instances, the median difference between pose estimations was 11.44 mm. For the incorrectly classified instances, the median difference was only 4.47 mm. This verifies that in cases where there was a large disagreement between the two sensors, our approach often made the correct decision. The missed instances were cases where the sensors only had a small level of disagreement. This analysis shows that our approach successfully eliminates the worst pose estimations in most cases.

V. Conclusion

We have presented a novel technique for improving full hand pose recognition accuracy from multiple sensors at frame rates in excess of 120 pose estimations per second. Our technique achieves this through a computational model, which is built to intelligently select the more accurate pose estimation at each time step, based on a subset of the underlying pose estimation data from each sensor. Our technique is most useful for improving the quality of tracking accuracy for gesture-controlled display interfaces, including medical interfaces. Our experimental results show that we were able to reduce the overall pose estimation error by over 30% in a two-sensor setup relative to the single sensor approach.

Other techniques that attempt to evaluate high-resolution images at high sampling rates from a large number of sensors quickly run into hardware limitations, computational challenges, and bandwidth constraints. In comparison, our approach is able to scale better with a large number of sensors because the data processed (i.e., the pose estimations) are at the skeletal pose level and are much smaller. A key contribution of our work is that it demonstrates the viability of combining pose estimations from multiple sensor systems, even without the presence of the underlying image data.

Acknowledgment

The authors would like to thank S. Khan for his input and feedback.

References

[1] N. Rossol, I. Cheng, R. Shen, and A. Basu, “Touchfree medical interfaces,” in *Proc. 36th Ann. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug. 2014, pp. 6597–6600.

[2] S. Grange, T. Fong, and C. Baur, “M/ORIS: A medical/operating room interaction system,” in *Proc. 6th Int. Conf. Multimodal Interfaces*, 2004, pp. 159–166.

[3] C. Grätzel, T. Fong, S. Grange, and C. Baur, “A non-contact mouse for surgeon-computer interaction,” *Technol. Health Care*, vol. 12, no. 3, pp. 245–257, Aug. 2004.

[4] J. P. Wachs, H. I. Stern, Y. Edan, M. Gillam, J. Handler, C. Feied, and M. Smith, “A gesture-based tool for sterile browsing of radiology images,” *J. Amer. Med. Informat. Assoc.*, vol. 15, no. 3, pp. 321–323, 2008.

[5] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, “Vision-based hand-gesture applications,” *Commun. ACM*, vol. 54, no. 2, pp. 60–71, 2011.

[6] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, “Efficient model-based 3D tracking of hand articulations using kinect,” in *Proc. Brit. Mach. Vis. Conf.*, 2011, pp. 101.1–101.11.

[7] C. von Hardenberg and F. Bérard, “Bare-hand human-computer interaction,” in *Proc. Workshop Perceptive User Interfaces*, 2001, pp. 1–8.

[8] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, “Analysis of the accuracy and robustness of the leap motion controller,” *Sensors*, vol. 13, pp. 6380–6393, 2013.

[9] S.-L. Sun and Z.-L. Deng, “Multi-sensor optimal information fusion Kalman filter,” *Automatica*, vol. 40, no. *6*, pp. 1017–1023, 2004.

[10] V. I. Pavlovic, R. Sharma, and T. S. Huang, “Visual interpretation of hand gestures for human-computer interaction: A review,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 677–695, Jul. 1997.

[11] Y. Zhu, G. Xu, and D. J. Kriegman, “A real-time approach to the spotting, representation, and recognition of hand gestures for humancomputer interaction,” *Comput. Vis. Image Understanding*, vol. 85, no. 3, pp. 189–208, 2002.

[12] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, “Robust part-based hand gesture recognition using kinect sensor,” *IEEE Trans. Multimedia, .*, vol. 15, no. 5, pp. 1110–1120, Aug 2013.

[13] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, “Vision-based hand pose estimation: A review,” *Comput. Vis. Image Understanding*, vol. 108, no. 12, pp. 52–73, 2007.

[14] I. Oikonomidis, “Tracking the articulated motion of two strongly interacting hands,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 1862–1869.

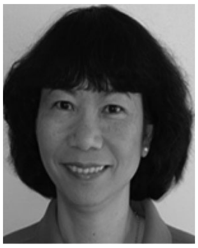
[15] C. Keskin, F. Kra, Y. Kara, and L. Akarun, “Real time hand pose estimation using depth sensors,” in *Consumer Depth Cameras for Computer Vision* (ser. Advances in Computer Vision and Pattern Recognition), A. Fossati, J. Gall, H. Grabner, X. Ren, and K. Konolige, Eds. London, U.K.: Springer, 2013, pp. 119–137.

[16] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, “Real-time human pose recognition in parts from single depth images,” *Commun. ACM*, vol. 56, no. 1, pp. 116–124, Jan. 2013.

[17] J. Romero, H. Kjellstrom, and D. Kragic, "Monocular real-time 3D articulated hand pose estimation," in *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots*, Dec. 2009, pp. 87–92.
[18] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 37, no. 3, pp. 311–324, May 2007.
[19] L. Gallo, A. Placitelli, and M. Ciampi, "Controller-free exploration of medical image data: Experiencing the kinect," in *Proc. 24th Int. Symp. Comput.-Based Med. Syst.*, 2011, pp. 1–6.
[20] J. Segen and S. Kumar, "Look ma, no mouse!" *Commun. ACM*, vol. 43, no. 7, pp. 102–109, Jul. 2000.
[21] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.
[22] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *Proc. SPIE*, vol. 1611, pp. 586–606, 1992.
[23] D. Vogel and R. Balakrishnan, "Distant freehand pointing and clicking on very large, high resolution displays," in *Proc. 18th Annu. ACM Symp. User Interface Softw. Technol.*, 2005, pp. 33–42.
[24] W. Fikkert, P. Vet, G. Veer, and A. Nijholt, "Gestures for large display control," in *Gesture in Embodied Communication and Human-Computer Interaction* (ser. LNCS, vol. 5934), New York, NY, USA: Springer, 2010, pp. 245–256.
[25] A. D. Wilson and A. F. Bobick, "Hidden Markov models for modeling and recognizing gesture under variation," in *Hidden Markov Models*. River Edge, NJ, USA: World Scientific, 2002, pp. 123–160.

Nathaniel Rossol received the M.S. degree in engineering from the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, in 2010, and the Ph.D. degree from the Department of Computing Science, University of Alberta, in 2015.

He is currently a Postdoctoral Fellow with the Department of Computing Science, University of Alberta. His current research interests include human–computer interaction, multimedia signal processing, and computer graphics.



Irene Cheng (SM'09) received the Ph.D. degree in computing science from the University of Alberta, Edmonton, AB, Canada.

She is currently the Scientific Director of the Multimedia Research Group, the Director of the Master with Specialization in Multimedia Program, and an Adjunct Professor with the University of Alberta, Edmonton, AB, Canada. Her research interests include multimedia computing and transmission, levels-of-detail, 3DTV, visualization, and perceptual quality assessment. In particular, she introduced applying human perception—Just-Noticeable-Difference (JND) and Relative Change—following psychophysical methodology for multiscale analytic.

Anup Basu (M'90–SM'02) received the Ph.D. degree in computer science from the University of Maryland, College Park, MD, USA.

He is currently a Professor with the Department of Computing Science, University of Alberta, Edmonton, AB, Canada, and is an iCORE-NSERC Industry Research Chair. His current research interests include 3-D/4-D image processing and visualization especially for medical applications, multimedia in education and games, and wireless 3-D multimedia transmission.