Real-Time Continuous Gesture Recognition for Natural Human-Computer Interaction

Ying Yin
CSAIL MIT
Email: yingyin@csail.mit.edu

Randall Davis
CSAIL MIT
Email: davis@csail.mit.edu

***Abstract*—Our real-time continuous gesture recognition system addresses problems that have previously been neglected: handling both gestures that are characterized by distinct paths and gestures characterized by distinct hand poses; and determining how and when the system should respond to gestures. Our probabilistic recognition framework based on hidden Markov models (HMMs) unifies the recognition of the two forms of gestures. Using information from the hidden states in the HMM, we can identify different gesture phases: the pre-stroke, the nucleus and the post-stroke phases. This allows the system to respond appropriately to both gestures that require a discrete response and those needing a continuous response. Our system is extensible: in only a few minutes, users can define their own gestures by giving a few examples rather than writing code. We also collected a new gesture dataset that contains the two forms of gestures, and propose a new hybrid performance metric for evaluating gesture recognition methods for real-time interaction.**

Keywords*—Gesture recognition, real-time, hidden Markov models

I. Introduction

> Imagine how nice it would be, the next time you make a presentation, you do not need to stand close to your laptop or use a remote control with limited functionality. What if you could present your work as naturally as having a conversation with your audience. You swipe your hand left and right to change slides. When you point to the slide with your hand, the display shows a pointer cursor following where you are pointing at. When you are showing a video, you use a palm forward hand pose and move left and right to fast forward or rewind the video. When you need to jump to a particular slide, you make a circle gesture to show all the slides, and say "show this slide" while pointing at that slide. You can also make a dismiss gesture to pause the slide show (making the screen black) to get full attention from the audience.

This scenario shows an application of multi-modal interface to a real-world problem. Different categories of gestures play an important part in the scenario.

Gestural input has just become popular recently with the introduction of new sensors (e.g. Kinect, Leap). As gesture recognition systems mature, gestural input will become an integral part of human computer interfaces, and hence will be an interesting area to explore in terms of visual languages.

Many gesture input interfaces still mainly make the hands function as a mouse with a limited number of other gestures. Previous research on gesture recognition usually focuses on one category of gestures, and the evaluation is also based on offline recognition. We believe that to design a gesture recognition system for natural human computer interaction (HCI), we need to start from the user interaction perspective: what are the different categories of gestures people use; when should the system respond; and how should the model be trained or defined. These are the questions we addressed when we develop our system.

The main contributions of this work include:

- We developed a unified probabilistic framework for real-time gesture recognition that combines two forms of gestures.
- We use embedded training and hidden state information to detect different gesture phases, allowing the system to respond more promptly.
- We collected a new dataset that include two forms of gestures, a combination currently lacking in the community. We also propose a hybrid evaluation metric that is more relevant to real-time interaction and different categories of gestures.

II. Gesture Taxonomy for Natural Interaction

This section introduces important concepts in gesture taxonomy that are the basis of our gesture modeling. Several gesture taxonomies have been suggested in the literature [1], [2], but the one that seems most appropriate for natural HCI and human-centric design was developed Wobbrock et al. [3]. Their study is based on eliciting natural behavior from non-technical users when interacting with a computing system.

Wobbrock et al. classify gestures in four orthogonal dimensions. As a first step, we focus on two of them (Table I) and incorporate them in designing the gesture recognition system and interface. We also further generalize the taxonomy to encompass interaction for both vertical and horizontal displays.

A. Gesture Forms and Flows

One dimension is the *form* of a gesture; we distinguish two categories in this dimension: path and pose. The path category contains gestures characterized by distinct paths without any distinct hand pose. For example, a "swipe left" gesture is characterized by a right to left motion, while a "circle" gesture

978-1-4799-4035-6/14/$31.00 ©2014 IEEE

TABLE I. TAXONOMY OF GESTURES FOR NATURAL INTERACTION.

Form	*distinct path*	with any hand pose
	distinct hand pose	with any path
Flow	*discrete*	response occurs *after* the user acts
	continuous	response occurs *while* the user acts

is characterized by a circular motion of the hand. In doing these, users typically hold their hands in some natural relaxed pose. The second category of gestures is characterized by distinct hand poses without any distinct paths. This category of gestures is usually associated with direct manipulation of some virtual objects on the interface. For example, a user may use a "point" hand pose and move around to point at different things on a display.

Another dimension is the *flow* of a gesture. A gesture's flow is discrete if the gesture is performed, delimited, recognized, and responded to as an *event* [3]. One example is the "wave" gesture which has a few repetitions of left and right movement; usually we want the system to respond at the last repetition. Flow is continuous if ongoing recognition is required and the system should respond frame by frame, as for example during a "point" gesture, where we want to show the cursor on the screen continuously moving according to the hand position.

B. Temporal Modeling of Gestures

Making gesture interaction feel natural requires a system that responds at the correct moment. As a result, it is important to consider the temporal characteristics of gestures. We set a foundation for doing this by taking account of the three phases that make up a gesture: *pre-stroke*, *nucleus*, and *post-stroke* [2]. Pre-strokes and post-strokes are movement from and to the rest position. The nucleus of a gesture has some "definite form and enhanced dynamic qualities" [1]. Every gesture must have a nucleus, which is the content-carrying part of the gesture.

Even though the end of the post-stroke phase can be more easily detected by finding the start of the rest position, we want to do more than this. Since nucleus is the meaningful part of the gesture, for a discrete flow gesture, we want the system to respond immediately at the end of the nucleus phase instead. To make the system more responsive, we address the more challenging problem of detecting the start and end of the nucleus phase from the pre-stroke and post-stroke phases. This also allows the system to respond to continuous flow gesture immediately at the start of the nucleus phase.

III. RELATED WORK

Many previous efforts have focused on one category of gesture, though some efforts have attempted to handle multiple categories.

A. Gesture with Distinct Hand Poses

One group of prior work focuses on classifying a set of predefined static hand poses frame by frame. Freeman and Roth [4] use histogram of local orientations, a precursor of histogram of oriented gradients (HOGs) [5] for hand pose recognition. Recognition is based on selecting the feature vector in the training set that is closest to the test feature vector. Suryanarayan et al. [6] use a volumetric shape descriptor computed from depth data as the feature vector, and use Support Vector Machine (SVM) for classification.

B. Gesture with Distinct Paths

The gesture production process has a direct analogy to the speech production process [7], and as a result many previous efforts have used hidden Markov models (HMMs) to recognize path gestures [8], [9]. More recently, discriminative models such as conditional random fields (CRF) and its variants, such as hidden CRF [10] and latent dynamic CRF (LDCRF) [11], have also been applied to gesture recognition with improved recognition results. However, discriminative models may require more data to train [12], and may also take more time to train as the parameter space of the model is larger.

Morency et al. [11] use LDCRF to learn the transition parameters between gestures. In our case, we assume the transition between gestures for interaction is uniform: each gesture is equally likely to transit to every other gesture.

C. Different Types of Gestures

Keskin et al. [13] propose a unified framework to allow concurrent usage of hand gestures, shapes and hand skeletons. Hand gestures are modeled with mixture of HMMs using spectral clustering. Hand shape classification and hand skeleton estimation are based on randomized decision forests. Hand gesture classification is active all the time. The framework estimates a set of posteriors for the hand shape label at each frame, and continuously use these posteriors and the velocity vector as observation to spot and classify known gestures. They distinguish gestures with pure motion and pure hand shape by thresholding the magnitude of the velocity vector. However, they did not mention handling gestures with distinct hand poses but with arbitrary movement. For this type of gestures, it will be hard to manually set a velocity threshold to distinguish them from gestures with distinct paths.

Oka et al. [14] developed a system that allows both direct manipulation and symbolic gestures, but requires the user to indicate the gesture type by the extension of the thumb.

D. Online Recognition

Song et al. uses LDCRF with a temporal sliding window to perform online sequence labeling and segmentation simultaneously [15]. Françoise et al. [16] use hierarchical HMMs to model musical gestures using motion data from the Wii remote controller, and use fixed-lag smoothing for real-time recognition and segmentation. Our method is similar to Françoise et al.'s, but we consider different *forms* of gestures in a single framework.

E. Commercial Systems

Many commercial gesture recognition systems uses if-then rules based on heuristics. For example, the Leap Motion plugin[1] for the Reveal.js presentation framework uses the number of fingers detected and the changes in the x and y coordinates between consecutive frames to detect swipe and point gestures. While if-then rules could be easy to define for a small number of simple gestures, it may be hard for more complex gestures. For example, it may be hard to define a circle gesture using if-then rules and the rules may conflict with each other.

[1] https://github.com/hakimel/reveal.js/blob/master/plugin/leap/leap.js

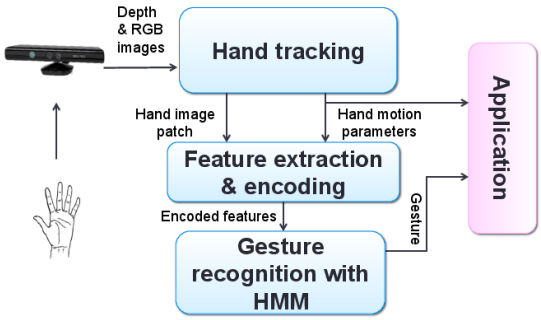

Fig. 1. System overview.

The gesture interaction provided by the Kinect-based games is one of the popular ones. Based on the observation of the interaction available in Kinect games, it seems that the system is only looking for one gesture (wave) or body pose (the exit pose[2]) at a time, and the rest of the time, it is just tracking the hand and the body.

F. Datasets

Many public datasets for evaluating gesture recognition contain only one *form* of gesture [17]–[19]. The Chalearn Gesture Dataset (CGD 2011) [20] contains nine gesture categories corresponding to various settings and application domains. It contains both static postures and dynamic gestures. In this dataset, a static posture is one in which a single posture is held for a certain duration. For a static hand posture, the hand is held at similar positions for multiple instances of the same gesture. In this case, the static postures also have distinct paths so they could be handled by the same method as the dynamic gestures. This dataset does not contain gestures with distinct hand poses but arbitrary movement (Table I row 2).

IV. System Overview

We develop our system based on the understanding of the gestures for natural interaction. Fig. 1 shows an overview of our system design. We use a RGB-depth sensor (the Kinect sensor) for hand tracking so that we can extract features for hand poses.

The gesture recognition model estimates the current most likely gesture label and gesture phase information based on the input stream of feature vectors. The gesture information, together with smoothed hand position information are sent to the application level at each time frame. We explain the main modules in detail in the following sections.

V. Hand Tracking and Feature Extraction

As one category of gestures is characterized by distinct hand poses, we need to track the full hand, rather than just treating the hand as one point. We also need to derive a feature vector that represents the hand shape as well.

We base our hand tracking on information from the skeleton tracking of the Kinect SDK, which is relatively robust for standing articulated body poses. At each time frame, we use the hand joint position reported from the SDK as an initial rough estimate of the bounding box of the hand in the depth frame. We align the RGB and the depth frames, use skin detection to filter out non-skin pixels in the bounding box, and then refine the bounding box using 4 interactions of CAMSHIFT [21]. We normalize the bounding box to a 32×32 px depth mapped image. We compute HOGs feature from the normalized hand image (cell size = 4, number of orientation bins = 9) (see Fig. 2). Since the depth data is less affected by change in illumination, we use only one fold of normalization in the HOG feature to speed up processing. This gives us a HOG feature of length 441 ($(32/4-1) \times (32/4-1) \times 9$). The HOG feature has been used as a hand pose descriptor in previous work [15] where it is often used as the input to SVM. Our system uses principal component analysis (PCA) to reduce the HOD dimensionality from 441 to 14, then uses it directly as part of the input feature vector to the hidden Markov model (HMM) based recognition framework.

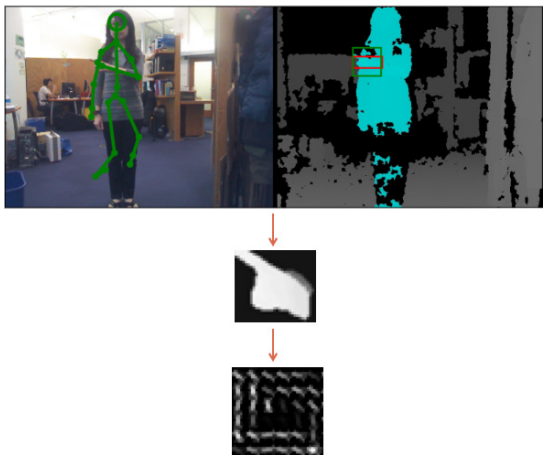

Fig. 2. Hand tracking and hand pose feature extraction.

The feature vector $\underline{x}_t$ at frame t is then a concatenation of motion features and encoded HOG features. The motion features include relative position of the hand to the shoulder joint, velocity and and acceleration, all in 3D world coordinates. The feature vector is computed for each input frame streamed from the sensor to form a sequence of feature vectors.

VI. Real-Time Continuous Gesture Recognition

The temporal model of gestures can be represented by a stochastic state machine. Each gesture phase can in turn also be represented by a stochastic state machine, with each state generating an observation (i.e. the feature vector). This process can be viewed as a hierarchical HMM (HHMM) (Fig. 3). If we assume that the states in the sub-HMMs are not shared, we can flatten the hierarchical HMM into a one-level HMM for fast inference, as the graphical model for one-level HMM does not have loops.

If we have ground truth labels for the pre-stroke, the nucleus and the post-stroke phases, we can train the sub-HMMs for each phase and each gesture separately and then combine them [22]. However in practice, for example if we want users to be able to easily add their new gestures by giving a few examples, it will be tedious to manually label

[2]http://support.xbox.com/en-US/xbox-360/kinect/how-to-use-the-kinect-hub-and-guide

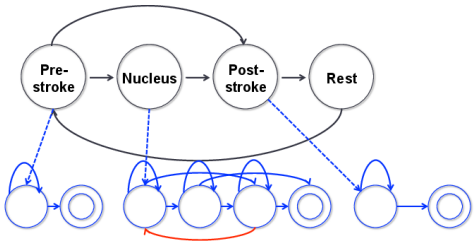

Fig. 3. State transition diagram of the hierarchical HMM representation of gesture phases. Double-ringed states are end states.

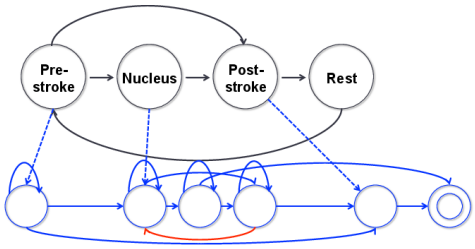

Fig. 4. Embed phase HMMs into an entire gesture.

the start and the end of the three phases. In this case, we can do embedded training [23], i.e. train each phase sub-HMM embedded in an entire gesture segment (Fig. 4).

A. Unified Framework

We do not treat the two *forms* of gestures separately: path gestures and pose gestures are handled within a single probabilistic framework. This avoids making early hard decisions (which *form* of gesture it is) which will be hard to correct later. Instead, we want to make a decision only when a response is needed, according to the *flow* of the current most likely gesture, and propagate belief probabilities as time progresses.

1) Gestures with distinct paths: We use embedded training to combine the three gesture phases together and use normal Baum-Welch algorithm to compute the maximum likelihood of the parameters.

Through cross-validation, we choose to use one hidden state for pre-stroke and post-stroke phases. We use the Bakis (left-right) model [24] for the nucleus phase, but add a backward transition from the last hidden state to the first one for gestures with an arbitrary number of repetitions (e.g., "wave" gesture). We use a mixture of Gaussians to model the emission probabilities for each hidden state. We also estimate the termination probabilities as in [22].

2) Gesture with distinct hand poses: We use one hidden state to represent this form of gestures. Let s_{pose} be the single hidden state for the nucleus phase for a gesture with a distinct hand pose (Fig. 5). Instead of doing embedded training, we directly compute the maximum likelihood estimates of the mixture of Gaussians parameters for emission probability of s_{pose}. Let $\underline{x}_1^T$ be a sequence of feature vectors corresponding to a gesture with a distinct hand pose. The feature vector sequence also contains random variations in the hand movement path. We use Expectation Maximization to estimate the means, covariance matrices and mixture probabilities for the mixture of Gaussians.

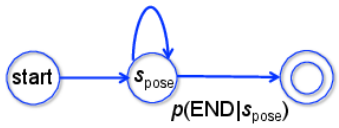

Fig. 5. State transition diagram of a single state HMM for gestures with distinct hand poses.

Since there is only one hidden state for s_{pose}, its transition probability is 1. Its termination probability is estimated according to the expected duration of the gesture. The self-arc on a state in an HMM defines a geometric distribution over waiting time [25]. In the case of a single state HMM, the probability of remaining in state s_{pose} for exactly d steps is $P(d) = p(1-p)^{d-1}$, where $p = P(END|s_{\text{pose}})$ is the termination probability for s_{pose}. This means the expected number of steps remaining in state s_{pose} is $\frac{1}{p}$. We assume that the minimum duration of a gesture with distinct hand pose is one second (30 frames). The termination probability $P(END|s_{\text{pose}})$ is then set to be less than $1/30$.

We also use one hidden state to model the rest position in a similar way.

B. Combined HMM for Real-Time Recognition

We train the HMMs separately for each gesture, then combine them into a hierarchical model, assuming uniform transition probabilities among gestures. The hierarchical model allows us to do simultaneous segmentation and recognition. We want to avoid doing segmentation first and then find the most likely HMM for the given sequence, because segmentation based on differentiating rest positions versus non-rest positions will not allow the system to respond fast enough. We want the system to respond at the beginning of the post-stroke phase instead. In addition, making a hard decision on segmentation can introduce errors that are hard to correct later.

For fast inference, we flatten the hierarchical HMM into a regular HMM by creating an HMM state for every leaf in the HHMM state transition diagram [25]. The individual HMM for each path gesture starts with a hidden state for the pre-stroke, then 1 or 2 hidden states for the nucleus, followed by a hidden state for the post-stroke. Each pose gesture has a hidden state for the nucleus. The combined HMM has a sequential labeling for these models' hidden states, with the hidden state label for the pre-stroke of the second gesture model following the hidden state label for the post-stroke of the previous gesture, etc (Fig. 6).

C. Online Inference

Once we have a trained model, we use fixed-lag smoothing [25] to do online inference on the flattened HMM for real-time gesture recognition. Fixed-lag smoothing is a modified forward-backward algorithm. Unlike online filtering, which estimates the belief state at current time t using forward pass only, we estimate the state at $t-L$, given all the evidence up to the current time t, i.e., compute $\gamma_{t-L}(s) \stackrel{\text{def}}{=\!=} P(S_{t-L} = s|\underline{x}_1^t)$, where $L > 0$ is the lag. Introducing lag time is a trade-off between accuracy and responsiveness. Using some future evidence to smooth the estimate can increase the accuracy

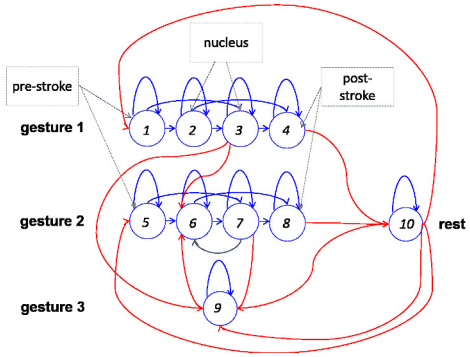

Fig. 6. Combined HMM. The red lines are examples of new transitions combining the HMMs. Not all possible transitions are shown to avoid cluttering the diagram.

while adding some delay. However if the delay is small, it might be unnoticeable. In the Experiment Evaluation section (Section IX), we show details about the relationship between L and the recognition performance.

Fixed-lag smoothing can be implemented efficiently. We compute forward probabilities α_t normally and keep a history window of $\alpha_{t-L} \dots \alpha_t$. At every time frame, we compute backward probabilities β from current time t to $t-L$. Then we can compute

$$\gamma_{t-L} = \alpha_{t-L} \cdot \beta_{t-L} \tag{1}$$

The time complexity at each time frame is $O(N_s^2 L)$ where N_s is the total number of hidden states in the flattened HMM. Note that at time t, the belief state at $t-L$ is committed, while the belief state from $t-L+1$ to t will still be revised later.

We can then compute the most likely hidden state at $t-L$:

$$\hat{s} = \arg\max_s \gamma_{t-L}(s) \tag{2}$$

We map the most likely hidden state to the gesture label it belongs to (including the rest position) and the gesture phase. In this way we achieve simultaneous segmentation and recognition.

Gesture events are detected at the boundary of a phase change: start pre-stroke, start gesture nucleus and start post-stroke. This information, together with the gesture label for the nucleus phase, are sent to the application level.

Fig. 7 shows a visualization of the most likely hidden states based on the online fixed-lag smoothing inference with $L = 5$. This is based on an input sequence of 6 gestures. The first 3 gestures are "circle" and the last 3 gestures are "shake hand" gestures. Notice that in the first segment, at the beginning, the most likely hidden state is the pre-stroke for "shake hand", but since we do not need to respond at this time, the wrong estimate does not matter. After a few more frames, the estimates are updated to have the correct most likely gesture label and the system responds correctly when it detects the start of the post-stroke of "circle" gesture.

VII. Data Collection and User Study

Previous related work does not appear to have gesture data sets that include both gestures with distinct paths and

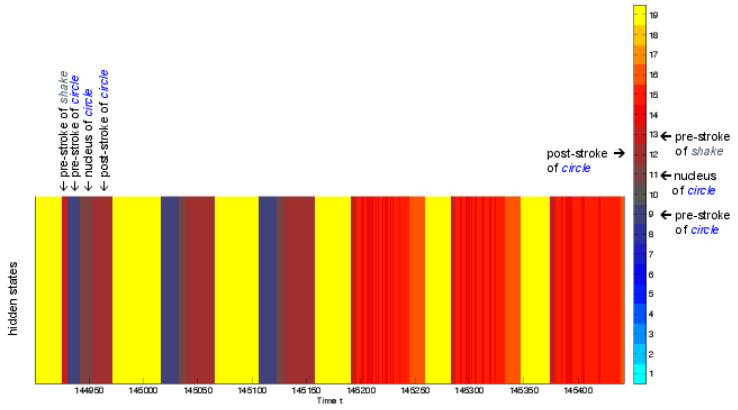

Fig. 7. Most likely hidden states using fixed-lag smoothing. Different colors indicate different hidden states. Yellow indicates rest position.

TABLE II. List of gestures recorded in the dataset.

#	Name of gesture	Form	Comment
1	Swipe left	distinct path	simple path
2	Swipe right	distinct path	simple path
3	Circle	distinct path	complex path
4	Horizontal wave	distinct path	has arbitrary repetitions
5	Point	distinct hand pose	arbitrary path
6	Palm forward	distinct hand pose	arbitrary path
7	Grab	distinct hand pose	arbitrary path

gestures with distinct hand poses. To evaluate our method, we collected a dataset with a vocabulary of 7 one-hand/arm gestures focusing on combining these two forms of gestures. They are also chosen to span over different potential difficulties (see the comments in Table II).

The dataset contains data from 10 participants each doing 4 sessions. All the participants are university students. The participants were shown video demonstration of each gesture at the beginning.

A. Recording Setup and Procedure

In each session, the participant stands at about 1.5m from the Kinect for Windows sensor (version one), and performs each gesture 3 times according to the text prompts on a screen indicating the name of the gesture to perform. The order of the gestures is random and the time between each gesture is random (between 2s and 6s). The first 2 sessions have "Rest" prompts between each gesture, telling participants to go to the rest position (hands relaxing at the side of the body), and the second 2 sessions do not have "Rest" prompts so participants can choose to rest or not between consecutive gestures. This too distinguishes the dataset from previous ones [18], [20] where gestures are always delimited by rest positions.

Unlike Ruffieux et al. [18], we do not show video demonstration every time the participants perform a gesture because we want a realistic scenario. In real practice, it is unlikely that a user will follow a video demonstration every time he/she does a gesture. The result of this is that there will be more variations among the gestures.

To motivate movement for gestures with distinct hand poses that require a continuous response, the text prompt asks

participants to draw random letters in the air with the specified hand pose.

The full corpus contains $10P \times 4S \times 7G \times 3R = 840$ gesture occurrences where P = participants, S = sessions, G = unique gestures, R = repetitions per gesture. There are approximately 96 minutes of continuous recording, recorded in the raw data from the Kinect sensor, including RGB, depth and skeleton data. Both the RGB and depth data have a resolution of 640×480px. The frame rate is 30 frame per second (FPS).

B. Qualitative Observations

We find that there is considerable variations in the way participants perform each gesture even they were given the same demonstration video. Major variations are observed in speed, the magnitude of motion, the paths and hand poses.

For example, some participants do swipe left and right in a rather straight horizontal line, while others have a more curved path. Some participants start the circle gesture at the bottom, while others start at the top. Some participants do swipe left and right with a palm forward pose while others have less distinct hand poses (hand is more relaxed). However, within users, they are quite consistent within each gesture.

C. User Preferences

We did a survey with the participants with questions that can influence gesture interface design. We asked them, given a gesture input interface:

- Whether they prefer predefined gestures or user defined gestures: 90% of the participants prefer to be able to define their own gestures if necessary while 10% of them prefer to following prefined gestures completely. As no one prefers to define their own gestures at the very beginning either.
- How to define gestures: 80% prefer defining gesture by performing the gestures themselves; no one prefers to define gestures solely via rules written in terms of positions and directions of movement of the hands. However 20% prefer to being able to do both.
- Number of repetitions per gesture for training: 50% are willing to give a maximum of 4 - 6 examples, 40% are willing ot give 1 - 3 examples, and 10% are willing to give more than 13 examples. So average maximum is about 5 repetitions.
- Number of gestures for an application: 80% think 6-10 gestures are appropriate and easy to remember for a given application, 20% prefers 1 - 5 gestures.
- Intuitiveness of the gesture vocabulary for PowerPoint presentation: average score is 4 out of 5 where 5 is very intuitive.

D. Implications for Gesture Interaction Interface

Based on our observation of the large variation in gesture execution among users and small variations within users, and the fact that a majority of participants preferring defining their own gestures if they do not like the predefined gestures, we suggest that it is more important to optimize user dependent recognition. As no one prefers to define their own gesture at the very beginning, it also means that having a reasonable predefined gesture set and basic user independent model for recognition will be useful too.

Recognition methods based on examples will allow users to train models of their own gestures easily. We also need to develop methods that require relatively few training examples.

VIII. Hybrid Performance Metric

Both frame [15] and event-based [20] metrics has been used for evaluating gesture recognition systems. A *frame* is a fixed-length, fixed-rate unit of time. It is often the smallest unit of measure defined by the system [26] and in such cases approximates continuous time. For example, in our case, a frame is a data frame consisting of a RGB data frame, a depth data frame and a skeleton data from the sensor at 30 FPS. An *event* is a variable duration sequence of frames within a continuous time-series. It has a start time and a stop time.

Each of these metrics alone may not be adequate for evaluating a real-time gesture recognition system handling different types of gestures. Frame-based evaluation is less relevant for gestures requiring discrete responses. For the ChaLearn Gesture Challenge 2012, Guyon et al. [20] use the Levenshtein distance between the ordered list of recognized events and the ground truth events. However, such event-based metrics that ignore timing of recognition are inappropriate for real-time applications where responsiveness of the system matters.

Ruffieux et al. combined a time-based metric with an event-based metirc [18]. Ward et al. [26] proposed a comprehensive scheme of combining both frame and event scoring. We believe that all three types of information – frames, events and timings – are relevant for a real-time system that responds to different types of gestures. Hence, we developed a hybrid performance metric.

For discrete flow gestures, the system responds at the end of the nucleus phase, so the evaluation should be event-based. Let $T_{\text{gt\_start\_pre}}$ be the ground truth start time of the pre-stroke phase and $T_{\text{gt\_stop\_post}}$ be the ground truth stop time of the post-stroke phase. A recognized event is considered a true positive (TP) if the time of response (T_{response}) occurs between $T_{\text{gt\_start\_pre}}$ and $T_{\text{gt\_stop\_post}} + 0.5 \times (T_{\text{gt\_stop\_post}} - T_{\text{gt\_start\_pre}})$. We allow some margin for error because there can be small ground truth timing errors. Once a TP event is detected, the corresponding ground truth event is not considered for further matching, so that multiple responses for the same gesture will be penalized. We then can compute event-based precision, recall and F_1 score.

For discrete flow gestures, we also define a Responsiveness Score (RS) as the time difference in seconds between the moment when the system responds and the moment when the hand goes to a rest position or changes gesture. Let N_{TP} be the number of true positives, then

$$RS = \frac{\sum_{i=1}^{N_{TP}} T_{\text{gt\_stop\_post}} - T_{\text{response}}}{N_{TP}} \tag{3}$$

A positive score means the responses are before the end of the post-strokes, hence higher scores are better.

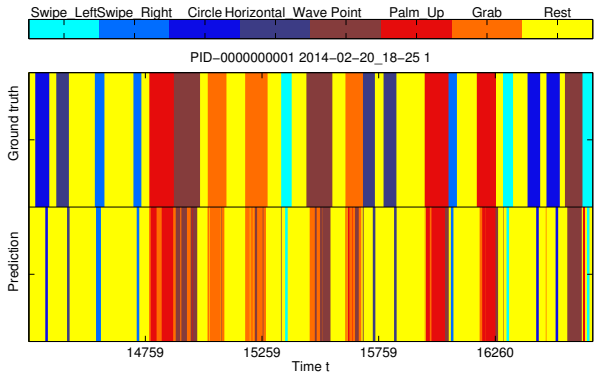

Fig. 8. Comparison between recognition result using online inference and ground truth. The colors correspond to different gestures. For discrete flow gestures (swipe left/right, circle, horizontal wave), one color segment with a fixed length is shown at the time of response. For continuous flow gestures, the recognized gesture is shown at each frame indicating frame-by-frame responses.

For continuous flow gestures, the system responds frame by frame, so it is more appropriate to use frame-based evaluation. For all the frames that are recognized as continuous gestures, we can compute the number of TPs by comparing with the corresponding frame in the ground truth. Then, we can compute frame-based precision, recall and F_1 score for all the frames corresponding to continuous flow gestures.

The average of the two F_1 scores gives an indication of the overall system performance.

IX. Experiment Evaluation

We evaluate our method based on the dataset we collected, using the evaluation metrics proposed in the previous section. The evaluation is based on the assumption that all the gestures with distinct paths (gesture #1–4) are discrete flow gestures, and gestures with distinct hand poses (gesture #5–7) are continuous flow gestures.

Our survey results show that it is important to allow users to quickly define and train their own gestures. Hence, we evaluate our system using user dependent training and testing. For each user in the dataset, we use the first 2 sessions of recording (6 samples per gesture) as training examples, and the last 2 sessions as testing examples. Fig. 8 shows a visualization of the recognition result on a test sequence. The figure highlights the challenges in the test sequences: there are 21 gestures in each continuous unsegmented sequence; sometimes gestures immediately follow one another. We report the average results for 10 users.

- **Compare different topologies:** We compare our unified framework with a common HMM-based approach used in previous works [8], [9], i.e., using the same left-right topology for all gestures. Table III compares the results between the two methods. The third column is the result from treating the two *forms* of gestures in the same way, i.e., all gestures have the same left-right Bakis model for their nucleus phases. The forth column is the result from using a left-right Bakis model for path gestures and a single state for pose gestures. To have a fair comparison, all hidden states have 3 mixtures of Gaussians. The result shows that our method of having different HMM topologies for the two *forms* of gestures significantly increases the precision, the recall and the F_1 score for pose gestures.

TABLE III. Results from using different topologies. The numbers in parentheses are standard deviations. The results are based on using 3 mixtures of Gaussians for all hidden states, and lag time $L = 5$ frames.

		Same topology for two *forms* of gestures	Different topologies for two *forms* of gestures
Path & discrete flow gestures	Precision	0.77 (0.14)	0.77 (0.15)
	Recall	0.89 (0.09)	0.88 (0.11)
	F_1	0.82 (0.10)	0.81 (0.11)
	Responsiveness (s)	0.6 (0.3)	0.6 (0.3)
Pose & continuous flow gestures	Precision	0.54 (0.13)	0.60 (0.10)
	Recall	0.24 (0.08)	0.62 (0.09)
	F_1	0.33(0.09)	0.61 (0.09)
Average	F_1	0.58 (0.10)	**0.71 (0.10)**

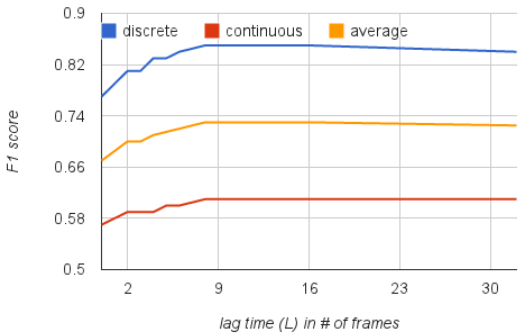

Fig. 9. F_1 scores versus lag time.

- **Compare different lag times:** Using our unified framework, we investigated how the F_1 scores change with respect to the lag time (L) in fixed-lag smoothing (Fig. 9). The performance increases as L increases, and plateaus at $L = 8$ frames which is about 0.3s at 30 FPS. This shows that more evidence does help to smooth the estimates until a limit, and we do not need to sacrifice too much delay to reach the limit. Our result shows that the responsiveness score stays around 0.6-0.7 seconds as we increase L.
- **Compare different number of mixtures:** Within a user, there may be variation in the hand pose for a gesture with distinct hand pose. For example, the "point" hand pose can have different orientations. This is why we use a mixture of Gaussians for the emission probabilities. Fig. 10 shows that the F_1 scores increases as the number of mixtures (M) increases until $M = 3$. After that, we start to see the effect of overfitting. Then, we experimented with using different M's for path and pose gestures. We set $M^{\text{path}} = 3$ for path gestures, and use a different $M_g^{\text{pose}} \in \{3 \dots 6\}$ for pose gesture g. Each M_g^{pose} is chosen using Bayesian Information Criterion [27]. Using this method, we are able to improve the overall average F_1 score to 0.81 when $L = 8$ frames.

Our system is fast to train and extensible. The average computation time for training the model for one user is about 5s with 7 gestures and 6 training examples per gesture. New

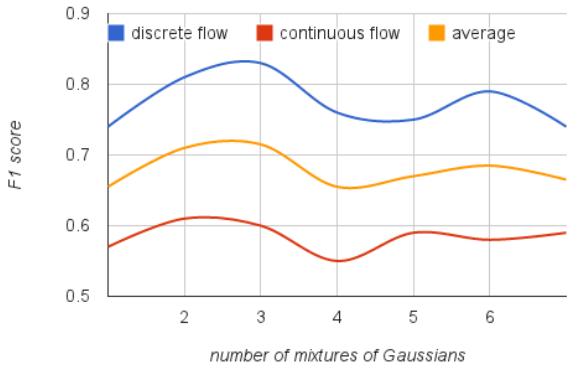

Fig. 10. F_1 scores versus number of mixtures of Gaussians. Lag time $L = 5$ frames.

gestures can be added by recording 3-6 repetitions of the gesture using the Kinect; the system will train an HMM model for the gesture and add it to the existing combined HMM. This process takes only a few minutes.

We also evaluated our method with a public dataset (ChAirGest) containing 10 path gestures performed by 10 users [22]. We obtained an F_1 score of 0.86 with offline user independent recognition. While it offers a baseline only for path gestures, it does show that our unified approach performs at the state of the art on this category of gesture.

X. Conclusion and Future Work

We designed our system from natural human computer interaction perspective. The evaluation shows promising results for the unified probabilistic recognition framework that handles two *forms* of gestures seamlessly. Even using only a small number of training examples (e.g. 6 per gesture), our system can achieve an average F_1 score of 0.81 for two *forms* of gestures. Using the framework, we developed a real-time (30 FPS) gesture controlled presentation application similar to the one described at the beginning of this paper.

The performance for pose gestures is lower than that for path gestures. This may be due to limit in both the pixel and the depth resolutions of the Kinect sensor. It would be interesting to test the new version of the Kinect sensor which uses a time-of-flight depth sensor and is reported to have a higher resolution. We are also going to to explore other feature descriptors and encoding methods to see whether we can improve the result for pose gestures.

Using embedded training and hidden state information, we can effectively detect gesture phases, allowing the system to respond more promptly. On average, for discrete flow gestures, the system responds 0.6s before the hand comes to rest.

We collected a new dataset that includes two *forms* of gestures, a combination currently lacking in the community, and plan to make it public. We have also proposed a hybrid evaluation metric that is more relevant to real-time interaction and different types of gestures.

References

[1] A. Kendon, *Current issue in the study of gesture*. Lawrrence Erlbaum Assoc., 1986.

[2] 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, pp. 677–695, 1997.

[3] J. Wobbrock, M. Morris, and A. Wilson, "User-defined gestures for surface computing," in *CHI*, 2009, pp. 1083–1092.

[4] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *FG*, vol. 12, 1995, pp. 296–301.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1, June 2005, pp. 886–893.

[6] P. Suryanarayan, A. Subramanian, and D. Mandalapu, "Dynamic hand pose recognition using depth data," in *ICPR*, 2010, pp. 3105–3108.

[7] S. Kettebekov and R. Sharma, "Toward natural gesture/speech control of a large display," in *Lecture Notes in Computer Science*. Springer Verlag, 2001, pp. 133–146.

[8] T. Starner and A. Pentland, "Visual recognition of American sign language using hidden Markov models," in *FG*, 1995, pp. 189–194.

[9] R. Sharma, J. Cai, S. Chakravarthy, I. Poddar, and Y. Sethi, "Exploiting speech/gesture co-occurrence for improving continuous gesture recognition in weather narration," in *FG*, 2000, pp. 422–427.

[10] S. Wang, A. Quattoni, L. Morency, D. Demirdjian, and T. Darrell, "Hidden conditional random fields for gesture recognition," in *CVPR*, vol. 2, 2006, pp. 1521–1527.

[11] L. Morency, A. Quattoni, and T. Darrell, "Latent-dynamic discriminative models for continuous gesture recognition," in *CVPR*, 2007.

[12] A. Ng and A. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, p. 841, 2002.

[13] C. Keskin, E. Berger, and L. Akarun, "A unified framework for concurrent usage of hand gesture, shape and pose," http://goo.gl/2icUwv.

[14] K. Oka, Y. Sato, and H. Koike, "Real-time fingertip tracking and gesture recognition," *IEEE Computer Graphics and Applications*, vol. 22, no. 6, pp. 64–71, 2002.

[15] Y. Song, D. Demirdjian, and R. Davis, "Continuous body and hand gesture recognition for natural human-computer interaction," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2012.

[16] J. Françoise, B. Caramiaux, and F. Bevilacqua, "Realtime segmentation and recognition of gestures using hierarchical markov models," *Mémoire de Master, Université Pierre et Marie Curie–Ircam*, 2011.

[17] Y. Song, D. Demirdjian, and R. Davis, "Tracking body and hands for gesture recognition: Natops aircraft handling signals database," in *FG*, March 2011, pp. 500–506.

[18] S. Ruffieux, D. Lalanne, E. Mugellini, and D. Roggen, "Chairgest - A challenge for multimodal mid-air gesture recognition for close HCI," in *ICMI*, 2013.

[19] S. Marcel, "Hand posture and gesture datasets," http://www.idiap.ch/resource/gestures/.

[20] I. Guyon, V. Athitsos, P. Jangyodsuk, and H. J. Escalante, "The chalearn gesture dataset (CGD 2011)," 2013.

[21] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," 1998.

[22] Y. Yin and R. Davis, "Gesture spotting and recognition using salience detection and concatenated hidden markov models," in *ICMI*, 2013, pp. 489–494.

[23] S. J. Young, "The HTK hidden Markov model toolkit: Design and philosophy," *Entropic Cambridge Research Laboratory, Ltd*, vol. 2, pp. 2–44, 1994.

[24] B. Bauer and H. Hienz, "Relevant features for video-based continuous sign language recognition," in *FG*, 2000, p. 440.

[25] K. Murphy, "Dynamic bayesian networks: representation, inference and learning," Ph.D. dissertation, University of California, 2002.

[26] J. A. Ward, P. Lukowicz, and H. W. Gellersen, "Performance metrics for activity recognition," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 1, pp. 6:1–6:23, Jan. 2011.

[27] C. Fraley and A. E. Raftery, "MCLUST version 3: an R package for normal mixture modeling and model-based clustering," DTIC Document, Tech. Rep., 2006.