Gesture Spotting and Recognition Using Salience Detection and Concatenated Hidden Markov Models

Ying Yin
Massachusetts Institute of Technology
yingyin@csail.mit.edu

Randall Davis
Massachusetts Institute of Technology
davis@csail.mit.edu

ABSTRACT

We developed a gesture salience based hand tracking method, and a gesture spotting and recognition method based on concatenated hidden Markov models. A 3-fold cross validation using the ChAirGest development data set with 10 users gives an F1 score of 0.907 and an accurate temporal segmentation rate (ATSR) of 0.923. The average final score is 0.9116. Compared with using the hand joint position from the Kinect SDK, using our hand tracking method gives a 3.7% absolute increase in the recognition F1 score.

Categories and Subject Descriptors

H.5.2 [**Information Systems**]: Information Interfaces and Presentation—*User Interfaces*

Keywords

Gesture recognition; HMM; gesture spotting; Kinect

1. INTRODUCTION

With the introduction of relatively low-cost sensors for tracking body or hand movement, such as Microsoft's Kinect and the Leap Motion sensor, we are observing an increasing interest in using these sensors for natural human computer interaction. Gesture input is a main part of natural interaction, and as a result, improving the accuracy of continuous gesture spotting and recognition remains an important topic.

We present our approach to the gesture spotting and recognition problem, highlighting two main contributions: improved hand position detection based on gesture salience and gesture spotting using concatenated hidden Markov models (HMMs).

Our work builds on several related works. Marcos-Ramiro et al. [4] developed a method of computing hand likelihood maps based on RGB videos. Our hand tracking method is similar, but we use both RGB and depth data. In common with [8, 10], we use hidden Markov models (HMMs) to model dynamic gestures, but we train models separately for the pre-stroke, nucleus and post-stroke phases, concatenate them together and use Viterbi decoding to optimally segment the gesture sequences.

In the following sections, we explain our system in three main parts: feature extraction, temporal segmentation, and gesture spotting and recognition.

ICMI'13, In proceedings of the 15th ACM International Conference on Multimodal Interaction, pages 489-494, 2013.

2. HAND FEATURE EXTRACTION

We use both the Kinect and the Xsens data from the ChAirGest corpus [7] to extract hand motion feature vectors for gesture modeling. It is relatively easy to obtain features from the Xsens data. We choose to use linear acceleration (x, y, z), angular velocity (x, y, z) and Euler orientation (yaw, pitch, roll) from the Xsens unit on the hand to form a 9-dimensional feature vector $\underline{x}_t^{\text{xsens}}$ for every time frame t. From the Kinect sensor, we extract the position of the gesturing hand in (x, y, z) relative to the shoulder center joint to form a 3-dimensional vector $\underline{x}_t^{\text{kinect}}$. Combining the two, we have a 12-dimensional feature vector $\underline{x}_t = [\underline{x}_t^{\text{kinect}}, \underline{x}_t^{\text{xsens}}]$.

We use the Kinect skeleton tracking result for the shoulder center joint position, but do not use it for the hand position because it is not accurate when the hands are close to the body or when the hands are moving fast. We developed an improved method for hand tracking based on gesture salience using both RGB and depth data.

2.1 Gesture Salience Detection

Similar to Marcos-Ramiro et al. [4], we define gesture *salience* as a function of both the closeness of the motion to the observer (e.g., the camera) and the magnitude of the motion. There are 4 steps in our method (Figure 1).

2.1.1 Skin Segmentation

We use an off-the-shelf simple skin color detection method to compute a binary skin mask at time t, M_t^S, based on the RGB image. We also find the user mask, M_t^U obtained from the Kinect SDK based on the depth image. We align the two to find their intersection $M_t^{S \wedge U}$, which indicates the user's skin region.

2.1.2 Motion Detection

We compute the motion mask for the current depth frame based on 3 frames. We first filter each depth frame by the user and skin mask $M_t^{S \wedge U}$, and then smooth it through a median filter to obtain D_t (Figure 1(b)). Equation (1) computes the binary mask, $M_{t \vee t-1}^M$, indicating pixels whose depth values have changed from time $t-1$ to t (Figure 1(c)).

(a) (b) (c) (d) (e)

Figure 1: Gesture salience detection steps: (a) RGB image under low lighting condition; (b) depth map D_t filtered by skin and user mask, $M_t^{S \wedge U}$. False detection of skin is due to the similar colors between clothes and skin; (c) motion mask, $M_{t \vee t-1}^{M}$, indicating moved pixels for time t and $t-1$; (d) salience map with red color indicating high probability of the salience; (e) final gesture salience bounding box, B_t. (Best viewed in color. Based on data from ChAirGest corpus [7].)

$D_{t \vee t-1}$ is the absolute difference between D_t and D_{t-1}, and T is the threshold operator that filters out small changes in depth value (with a threshold of 15mm). To obtain the motion mask, M_t^M for time t only, we use $M_{t-1 \vee t-2}^{M}$, the motion mask for $t-1$ and $t-2$ as well (see Equation (2), AND and XOR are indicated by $\wedge$ and $\oplus$).

$$M_{t \vee t-1}^{M} = T(D_{t \vee t-1}) \quad (1)$$

$$M_t^M = M_{t \vee t-1}^{M} \oplus (M_{t \vee t-1}^{M} \wedge M_{t-1 \vee t-2}^{M}) \quad (2)$$

2.1.3 Salience Map

We compute histograms of depth values in both D_t and $D_{t \vee t-1}$ and then apply histogram normalization to obtain cumulative distributions H_t and $H_{t \vee t-1}$. H_t represents the probability of salience given a depth value, while $H_{t \vee t-1}$ represents the probability of salience given a depth difference value. The lower the depth value or the higher the depth difference value, the higher the salience probability. We use histogram equalization to reduce the effect of outliers, so that a single large depth value will not suppress the salience probabilities of other depth values. The salience map (Figure 1(d)) can then be computed for each pixel (x, y):

$$S_t(x,y) = H_t(D_t(x,y)) \times H_{t \vee t-1}(D_{t \vee t-1}(x,y)) \times M_t^M$$

The multiplication of the binary motion mask M_t^M allows us to consider only the motion due to the user at t.

2.1.4 Salience Location

The final step of locating the most salient region in a frame is finding the contour, C_t, from the salience map S_t that has a perimeter greater than the smallest possible hand perimeter and with the highest average salience for all the pixels inside the contour.

When motion is slow, the motion mask usually indicates the edge of the moving object. As a result, the center of C_t may not be the center of the moving object (in our case, the user's hand). Hence, we use 2 iterations of Camshift [2] on the depth image D_t with a starting search location at the center of C_t to refine the final bounding box, B_t, of gesture salience (Figure 1(e)).

Figure 2 shows examples of our hand tracking result (red regions). It is more reliable than the hand joint locations from the Kinect SDK. In the Experimental Evaluation section (Section 5), we show that using our salience detection method to extract hand position features gives 3.7% absolute increase in gesture recognition F1 score compared to using the hand joint position from the Kinect SDK.

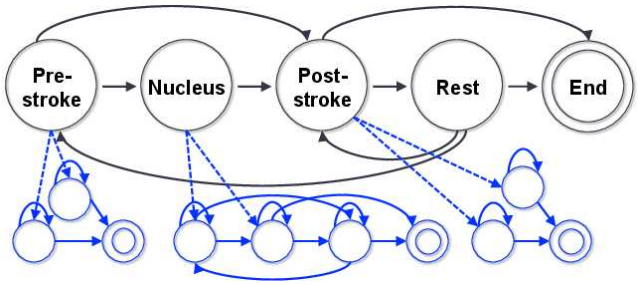


Figure 3: Temporal gesture model with different phases. Each phase can be modeled as an HMM. Dashed arrows represent initial state transitions and double circles represent end states.

3. TEMPORAL SEGMENTATION

We used all the training data from all the users to create a Gaussian model for the rest positions and a Gaussian model for non-rest positions.

During recognition, an observation $\underline{x}_t$ is first classified as a rest or a non-rest position. It is a non-rest position if

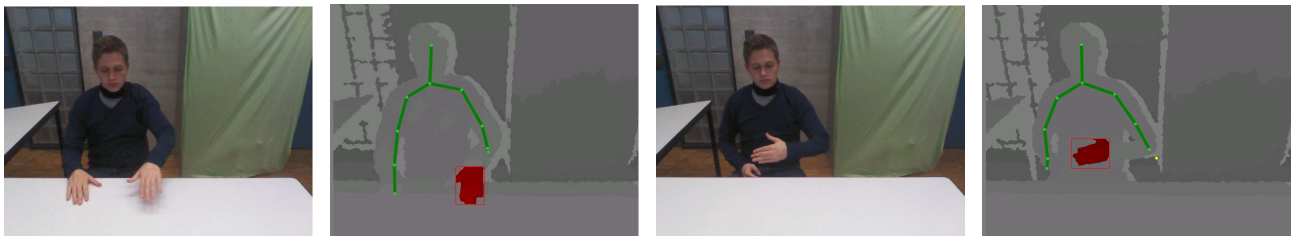
$$N(\underline{x}_t; \mu_{\text{NON-REST}}, \Sigma_{\text{NON-REST}}) \geq N(\underline{x}_t; \mu_{\text{REST}}, \Sigma_{\text{REST}})$$

where N represents the Gaussian probability. Sequences of continuous observations from non-rest positions longer than d seconds are further classified into different gestures based on trained HMMs. The threshold value d is the lower bound of gesture duration, and can be varied for different gesture sets. We set it to be 0.25s for our evaluation data set based on empirical result.

4. GESTURE SPOTTING AND RECOGNITION

4.1 Temporal Gesture Modeling and Training

Previous research suggests that a gesture consists of three phases: *pre-stroke*, *nucleus*, and *post-stroke* [6]. The pre-stroke phase consists of a preparatory movement that sets the hand in motion from some resting position. The nucleus of a gesture has some "definite form and enhanced dynamic qualities" [3]. Finally, the hand either returns to the resting position or repositions for the new gesture phase. Each gesture phase includes a sequence of hand/arm movement that can be modeled using HMMs (see Figure 3).



(a) (b) (c) (d)

Figure 2: Comparison of hand tracking results. Our method (red region) gives more reliable result on hand tracking compared to the off-the-shelf Kinect software (green line). (Best viewed in color. Based on data from ChAirGest corpus [7].)

Because we have the ground truth labeling of pre-stroke, nucleus and post-stroke phases, we can train an HMM for each phase for each gesture.

As each phase can have variable length, we model the termination probability for each hidden state s as $t(\text{END}|s)$. Given a sequence of observation $\underline{X}_1^T = \underline{x}_1 \dots \underline{x}_T$, and the corresponding hidden states sequence $S_1^T = s_1 \dots s_T$, we define the probability

$$p(\underline{X}_1^T, S_1^T; \underline{\theta}) = t(s_1)t(END|s_T)\prod_{t=2}^{T} t(s_t|s_{t-1})\prod_{t=1}^{T} e(\underline{x}_t|s_t)$$

where $\underline{\theta}$ represents the model parameter vector which includes the initial state probabilities $t(s)$, the state transition probabilities $t(s'|s)$, and the emission probabilities $e(\underline{x}|s)$ for $s, s' \in \{1, 2, \dots k\}$. We use a mixture of 6 Gaussians for the emission probability to model user variations.

Given N training sequences, we use the expectation maximization (EM) algorithm to estimate the model parameters. In particular, the update for the termination probability during the ith iteration is

$$t^i(END|s) = \frac{\sum_{j=1}^{N} \overline{count}(j, s \to END; \underline{\theta}^{i-1})}{\sum_{j=1}^{N}\sum_{s'} \overline{count}(j, s \to s'; \underline{\theta}^{i-1})}$$

where $\overline{count}(j, s \to END; \underline{\theta}^{i-1})$ is the expected count of s being the end state. We can use the usual forward-backward algorithm to compute all the expected sufficient statistics by adding a dummy END state to the end of each sequence.

Because there are 3 rest positions, we use 3 hidden states for both the pre-stroke and post-stroke phases. Each hidden state can be the start state and can only remain in its own state or go to the end state.

For the nucleus phase, we use 6 hidden states (chosen through cross validation) for all the gestures and use a modified Bakis [1] model to constrain the transition probabilities among the hidden states. Instead of allowing only left-right transition, we allow the last hidden state to go back to the initial state (Figure 4). This is particularly important for modeling gestures with arbitrary number of repetitions such as waving and shaking hands.

4.2 Gesture Recognition

During the recognition phase, we concatenate the HMMs trained for each phase together to form one HMM for each gesture. The transition probability from the previous phase to the next phase can be computed by multiplying the termination probabilities of the previous phase and the initial state probabilities of the next phase. Using the superscript c to denote the model parameters in the concatenated HMM, we have

$$t^c(s_{\text{nucleus}}|s_{\text{prestroke}}) = t(\text{END}|s_{\text{prestroke}}) \times t(s_{\text{nucleus}})$$

where s_{phase} denotes the hidden state variable in a particular phase. We add small transition probabilities from the pre-stroke phase to the post-stroke phase to model movements that do not have the nucleus phase.

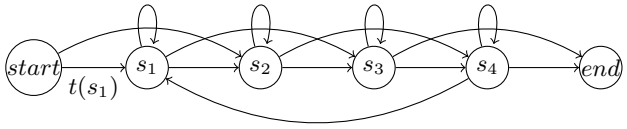


Figure 4: A state transition diagram of a modified 4-state Bakis model for the nucleus phase.

As new state transition probabilities are added, the transition probabilities among states in the same phase also need to be modified so that $\sum_{s'=1}^{K} t(s'|s) = 1$ (where K is the total number of combined hidden states) is ensured. For example

$$t^c(s'_{\text{nucleus}}|s_{\text{nucleus}}) = t(s'_{\text{nucleus}}|s_{\text{nucleus}}) \times (1 - t(\text{END}|s_{\text{nucleus}}))$$

We also add a rest state to the end of the HMM and allow the rest state to transit to the pre-stroke and the post-stroke phase with uniform probabilities (Figure 3) to accommodate short pauses during the gesture. As a result, the final HMM for each gesture has 13 hidden states. Let $\underline{\theta}_g$ be the final concatenated HMM parameters for gesture g. The classification of an observation sequence from non-rest positions is

$$\hat{g} = \arg\max_g \log p(\underline{X}_1^T; \underline{\theta}_g)$$

4.3 Gesture Spotting

Because the non-rest positions include both pre-stroke and post-stroke phases, we need to detect the start of the actual gesture (nucleus). We use the Viterbi algorithm to find the most probable hidden state sequence $\hat{s}_1 \dots \hat{s}_T$ for a given observed sequence using the mostly likely gesture model $\underline{\theta}_{\hat{g}}$. The start and the end time for a gesture nucleus are the first and the last time frame t where $\hat{s}_t \in s_{\text{nucleus}}$ respectively. Note that we are able to identify whether a hidden state belongs to the nucleus phase because we trained the three phases separately.

Figure 5 shows a recognition result visualization for one batch sequence. The first row is the ground truth with differ-

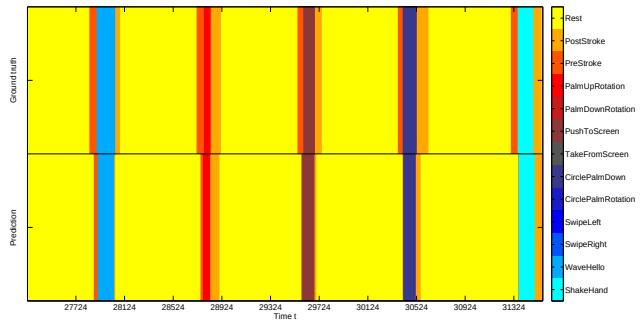


Figure 5: Visualization of a gesture recognition sequence. The pre-stroke and post-stroke phases are indicated by two orange colors (see the color bar).

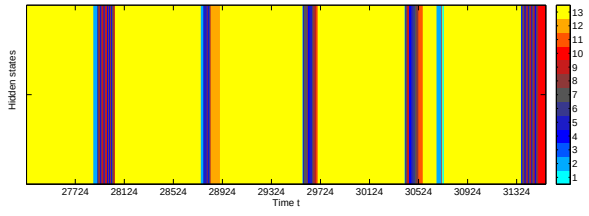


Figure 6: Visualization of the most probable hidden states of a gesture recognition sequence. Colors 1-3 indicate the pre-stroke hidden states, colors 4 - 9 indicate the nucleus hidden states, colors 10 - 12 indicate the post-stroke hidden states, and color 14 indicates the rest state.

ent colors indicating different gesture phases or the rest position. The second row is our segmentation and recognition result. Figure 6 shows the color-coded most probable hidden states for the same sequence. If a non-rest sequence does not contain hidden states belonging to the nucleus phase, it is ignored (see the blue bar at $t \sim 30700$ in Figure 6). In this way, we can spot the actual gestures while filtering out other movements.

5. EXPERIMENTAL EVALUATION

We evaluate our method based on the development data set from the ChAirGest corpus [7] with gestures captured from 10 users. There are 900 total gesture occurrences (3 recording sessions for each user) in the development data set representing three forth of the entire corpus. The remaining one forth of the corpus are not released to the public, and can be used for final evaluation on unseen data.

Table 1 compares the gesture recognition performance using different methods of computing feature vectors while keeping the recognition method the same. They are the average results of 3-fold cross validations where, in each fold, the model is trained on 2 sessions and tested on 1 session from every user. The results help to answer the following questions:

1. Does our salience based hand tracking method give better performance? The first column in Table 1 shows the results from using the salience based method to compute relative hand positions, and the second column are the results from using the hand joint positions from the Kinect SDK's skeleton tracking (version 1.6).

	Hand position from salience detection & Xsens	Hand position from Kinect skeleton & Xsens	Xsens Only
F1 Score	**0.907 (0.01)**	0.870 (0.02)	0.890 (0.02)
ATSR Score	**0.923 (0.02)**	0.930 (0.03)	0.920 (0.01)
Final Score	**0.912 (0.01)**	0.881 (0.01)	0.895 (0.01)

Table 1: Comparison of the average 3-fold cross validation results for different feature vectors. Values in parentheses are standard deviations.

Hit	259	Missed	33	Mislabel	7
Precision	0.86	Recall	0.87	F1	0.86
Avg. e_i^{start}	0.28%	Avg. e_i^{stop}	-3.17%	ATSR	0.91
				Final Score	**0.871**

Table 2: Final test result on unseen data.

The salience base method gives 3.7% absolute increase in the F1 score.

2. Does including the relative hand position in the feature vector help to increase performance? The third column shows the results from using only Xsens features. We observe that including the relative hand position computed using our hand tracking method gives 1.7% absolute increase in the F1 score. However, using the hand positions from the Kinect SDK actually decreases the performance.

The ATSR score in Table 1 stands for the accurate temporal segmentation rate which represents the performance of accurately detecting the start and stop points of gesture events [7]. Let e_i^{start} be the start error rate for a detected gesture nucleus i, and

$$e_i^{\text{start}} = \frac{\text{ground truth start time} - \text{detected start time}}{\text{ground truth duration of the gesture nucleus } i}$$

The stop error rate, e_i^{stop}, is defined similarly. Let n_{hit} be the number correctly recognized gestures (i.e., the label of the gesture nucleus i is correct, and the absolute start error and the stop error rates are less than 50%). The formula for ATSR is

$$ATSR = 1 - \frac{\sum_{i=1}^{n_{\text{hit}}} |e_i^{\text{start}}| + |e_i^{\text{stop}}|}{2 \cdot n_{\text{hit}}}$$

The per frame classification confusion matrix (Figure 7) shows that the most easily confused gestures are "take from screen" and "push to screen". These two gestures are very similar except for hand poses. This suggest that by including hand pose features may further improve the recognition accuracy. There are also more confusions at the boundary between phases, e.g., from rest to pre-stroke and from post-stroke to rest.

Table 2 shows the final test result on unseen data based on the model trained on all the development data using feature vectors computed from our salience based hand tracking method. This result is provided by the ChAirGest organizer after running our program. The decrease in performance

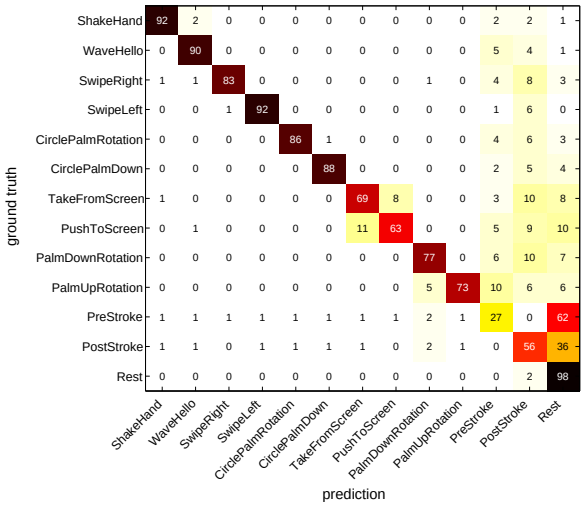


Figure 7: Per frame classification confusion matrix based on result from 3-fold cross validation using both Kinect and Xsens features. The numbers are percentages. The darker the color the higher the percentage.

may be due to the overfitting of the model on the development data.

6. CONCLUSIONS AND FUTURE WORKS

Our gesture spotting and recognition method based on concatenated HMMs trained for the three gesture phases and Viterbi decoding gives good results on the data set. Using hand position features computed from our gesture salience detection method also helps to increase the gesture recognition accuracy. Compared with using the hand position from the Kinect SDK, using our hand tracking method gives a 3.7% absolute increase in the recognition F1 score.

For future works, we are going to apply discriminative training for learning the parameters of HMMs. We are also going to compare the HMM based method with the hidden conditional random field (HCRF) based methods [5, 9]. It would be interesting to explore whether we can also concatenate the HCRF models.

For hand tracking, we can apply temporal smoothing to depth data to see whether that helps to improve accuracy.

7. REFERENCES

[1] B. Bauer and H. Hienz. Relevant features for video-based continuous sign language recognition. In *FG*, pages 440–445, 2000.

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

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

[4] A. Marcos-Ramiro, D. Pizarro-Perez, M. Marron-Romera, L. S. Nguyen, and D. Gatica-Perez. Body communicative cue extraction for conversational analysis. In *Automatic Face and Gesture Recognition*, 2013.

[5] L. Morency, A. Quattoni, and T. Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *CVPR*, pages 1–8. Ieee, 2007.

[6] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:677–695, 1997.

[7] S. Ruffieux, D. Lalanne, E. Mugellini, and D. Roggen. Chairgest - A challenge for multimodal mid-air gesture recognition for close HCI. In *ICMI (to appear)*, 2013.

[8] T. Starner and A. Pentland. Visual recognition of American sign language using hidden Markov models. In *Automatic Face and Gesture Recognition*, 1995.

[9] S. Wang, A. Quattoni, L. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *CVPR*, volume 2, pages 1521–1527, 2006.

[10] Y. Yin and R. Davis. Toward natural interaction in the real world: Real-time gesture recognition. In *ICMI*, 2010.