Gesture Beyond the Surface: Online Continuous Gesture Recognition

1st Author Name
Affiliation
Address
e-mail address

2nd Author Name
Affiliation
Address
e-mail address

ABSTRACT

Author Keywords
Guides; instructions; author's kit; conference publications; keywords should be separated by a semi-colon.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

RELATED WORK

Bag-of-feature approach is similar to the bag-of-word approach in document classification. To represent an image using BoW model, an image can be treated as a document. Similarly, "words" in images need to be identified too. To achieve this, it usually includes following three steps: feature detection, feature description and code book generation [5]. In the bag-of-word approach for document classification, the notion of order of words is lost [9]. Higher-order n-gram models maintain some local notion of word order.

Heng et al. did an evaluation of several feature detectors and descriptors for action recognition in video sequences [**?**]. They used the same bag-of-features SVM classification method for all the feature detector and descriptor combinations. Codebook With the bag of words

SVM does not usually handle time series data. To use SVM for time series, one can concatenate the input in a time series into one long vector sample. scale short time action recognition

Any HHMM can be converted to regular HMM by creating an HMM state for every leaf in the HHMM state transition diagram [8]. Both HHMM and HMM have advantages and disadvantages. Advantages for HHMM over a mixture of HMMs model.

- Can share substructures (S_t) across different HMMs, but not in the mixture of HMMs model.
- HHMM can provide a mult-A flat HMM cannot easily provide

Advantages of HMM

- Can specify more constraints in the the parameters, i.e. use Bakis model. to reduce the total number of parameters It is harder to impose Bakis model to HHMM since the hidden states are shared. The state transition probability for S_t in HHMM is $|S|^2 \times |G|$. In mixture HMMs model, it is $k \times |S| \times |G|$ where k is the number of states a state can transit to.

If we do not constrain the transition in the hidden states S_t, HHMM may have a fewer number of parameters because of the states sharing. If we constrain the transition, specifying more structures in the sub HMMs, the mixture of HMMs can have fewer number of parameters.

Training HMMs separately and combining them into HHMM for gesture inference. No across-gesture sharing (of hidden states)

dynamic time warping

dense sampling cite

Hand Tracking

Marcos-Ramiro et al. [7] develop a method of computing hand likelihood maps based on RGB videos. Optical flow hands show more movement. Our method is very similar to their approach but we combine both RGB images and depth images to compute the gesture salience map. They hypothesized that given a frontal, static camera pointing to the upper body of a person, hands are normally the parts of the image that show more movement. The two strong indicators are:

Space-time interest point

IMPLEMENTATION DETAILS

PCA reduction. The features are in the PC space, so they should have small correlations. So we set the covariance matrices for the Gaussian CPD to be diagonal.

Local space-time features capture characteristic shape and motion in video and provide relatively independent representation of events with respect to their spatio-temporal shifts and scales. [10]

[10]

Kinect skeleton tracking methods random forest

Submitted for review.

SYSTEM OVERVIEW

Our online continuous hand gesture recognition framework consists mainly two parts: feature extraction and gesture classification.

Feature extraction on video sequences can be broken down into two steps: 1) detect points of interest by maximizing certain salience functions; 2) compute feature descriptors to capture shape and motion in the neighborhoods of selected points using image measurements [10].

For hand gesture recognition, the points of interest are certainly the hands. While the skeleton tracking provided by the Kinect SDK is quite robust most of the time, its error for the hand joint tracking increases when the hands are close to the body or move quickly (see Figure 2). As a result, we develop the gesture salience detection method to locate the gesturing hand more accurately.

GESTURE SALIENCE DETECTION

Similar to Marcos-Ramiro et al. [7], we define gesture salience as a function of the closeness of the motion to the observer (e.g., the camera) and the magnitude of the motion.

For a given data frame from the RGB-D camera, we combine both the RGB and the depth data to compute gesture salience. Both the RGB and the depth data can be noisy. RGB cameras are sensitive to lighting conditions (cite). Skin color based detection can be sensitive to the clothes users wear. Depth cameras based on structured light sensors, such as the first generation Kinect sensor, compare the projected infra-red pattern with the reflected one to determine depth [11]. As a result, they do not work well on objects that are highly reflective (mirrors and shiny metal) or highly absorptive (fluffy and/or dark materials such as human hair) [1]. By combining the two, the overall gesture salience detection can be more robust to noise.

The following are the detailed steps of our gesture salience detection method for each frame. We show the illustrations in Figure 1.

Skin Segmentation

We use an off-the-shelf simple skin color detection method which is not trained on our data set to do a binary segmentation. The reason is to make the method generalizable to any users and environment. We compute a binary skin mask, M^S, based on the RGB image converted into YCrCb color space. We also find the user mask, M^U obtained from the Kinect SDK based on the depth image. We then align M^S with M^U and find their intersection $M^{S \wedge U}$, which indicates the skin region on the user.

Motion Detection

Similar to Cutler and Turk [2], 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 $M^{S \wedge U}$ at time t, and then smooth it through a median filter to obtain D_t (Figure 1(b)). Equation (1) computes the binary mask, $M^M_{t \vee t-1}$, indicating moved pixels at either time t or $t-1$ (Figure 1(c)). $D^D_{t \vee t-1}$ is the absolute difference between D_t and D_{t-1}, and T is the threshold operator that filters out small change in depth value (we set the threshold to be 15mm). To obtain the motion mask, M^M_t for time t only, we use $M^M_{t-1 \vee t-2}$, the motion mask for $t-1$ and $t-2$ as well (see Equation (2)-(3), symbols $\wedge$ and $\oplus$ are the AND and XOR operators respectively).

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

$$M^M_{t-1} = M^M_{t \vee t-1} \wedge M^M_{t-1 \vee t-2} \tag{2}$$

$$M^M_t = M^M_{t \vee t-1} \oplus M^M_{t-1} \tag{3}$$

Salience Map

We apply histogram equalization to both D_t and $D^D_{t \vee t-1}$ to obtain cumulative distributions H_t and $H^D_{t \vee t-1}$. H_t represents the probability of salience given a depth value and $H^D_{t \vee t-1}$ 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. Finally the salience map (Figure 1(d)) can by computed for each pixel (x, y) as

$$S_t(x,y) = H_t(D_t(x,y)) \times H^D_{t \vee t-1}(D^D_t(x,y)) \times M^M_t \tag{4}$$

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

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 in side 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 [1] on the depth image D_t with a start search location at then center of C_t to refine the final bounding box, B_t, of gesture salience (Figure 1(e)).

Figure 2 compares the results of gesture salience detection for hand gestures with the skeleton tracking results from the Kinect SDK.

ONLINE CONTINUOUS GESTURE RECOGNITION

Feature Descriptor

From the bounding box, B_t, of the detected gesture salience, we compute both motion trajectory feature X^M_t and hand pose feature X^P_t.

The motion trajectory feature X^M_t include relative position from B_t to the center of the shoulders ($\mathbf{p}$), velocity ($\mathbf{v}$), and acceleration ($\mathbf{a}$). The position of the shoulder center from the Kinect skeleton tracking is relatively accurate all the time. The 3-dimensional vectors $\mathbf{p}$, $\mathbf{v}$, $\mathbf{a}$ are in the world coordinate systems.

[1]`http://msdn.microsoft.com/en-us/library/hh855356.aspx`

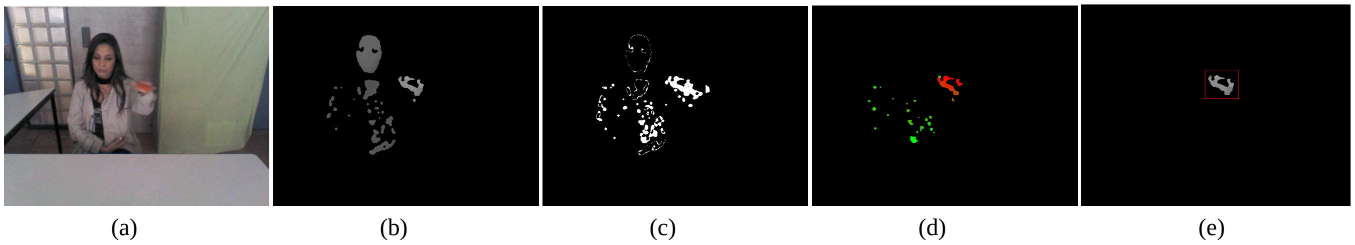

Figure 1. Gesture salience detection steps: (a) RGB image under low lighting condition; (b) depth map D_t filtered by skin and user mask, $M^{S \wedge U}$. False detection of skin due to clothes color similar to skin color; (c) motion mask, $M^M_{t \vee t-1}$, indicating moved pixels for time t and $t-1$; (d) salience map with red color indicating high probability of the salience; (e) final gesture salience bounding box, B.

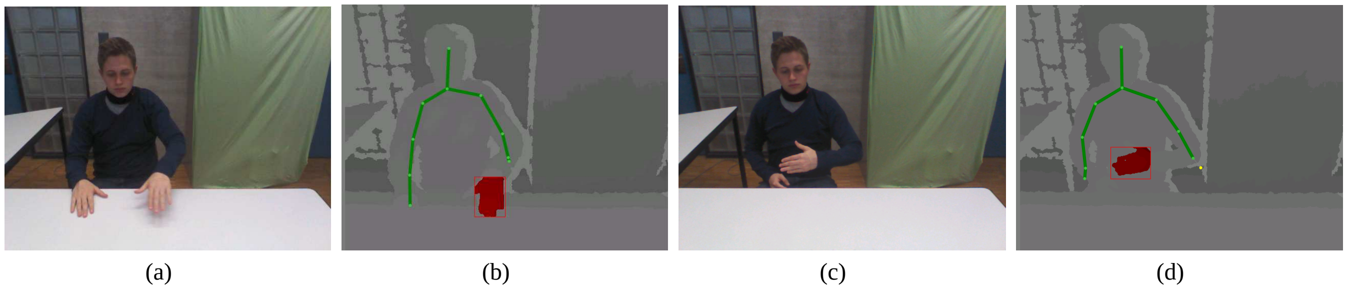

Figure 2. Comparison between gesture salience detection for hand gestures and skeleton tracking from the Kinect SDK. The green lines are the skeleton tracking results. The red region is the detected salient gesture region using our method.

For hand pose features, we extract a 64×64 px image patch, I_t, with depth-mapped values from B_t. We denoise, I_t, using a morphological close operation and then compute the HOG feature descriptor [3], I_t^H, from it. The cell size we use is 4×4 px and the number of orientation bins is 9. Our earlier work (under submission) shows that finer grained HOG descriptor for hand poses gives better recognition accuracy. We also only use one fold of normalization because depth values are less affected by illumination variation and contrast normalization is not necessary. This results in a $14 \times 14 \times 9$ length I_t^H.

Similar to Wang et al. [10], we apply dimension reduction on I_t^H concatenated as a column vector using principal component analysis on training data. The final hand pose feature X_t^P is obtained by projecting I_t^H to a lower dimensional space with dimension k.

The final feature vector X_t combined from X_t^M and X_t^P has dimension $9 + k$. They are standardized to have zero mean and unit variance. Figure 3 shows a visualization of the feature vectors with $k = 7$ for a sequence of continuous gestures with rest poses in between. The regions of smoothness indicate rest poses while the regions of rapid changes indicate gestures.

Training AHMM for Gesture Recognition

Our earlier work (under submission) has shown that the 1-level abstract hidden Markov model (AHMM) closely models the gesture production process and incorporates gesture recognition and gesture segmentation at the same time. In this work we give a quantitative comparison between a model trained from AHMM and one trained from a mixture of HMMs. (cite)

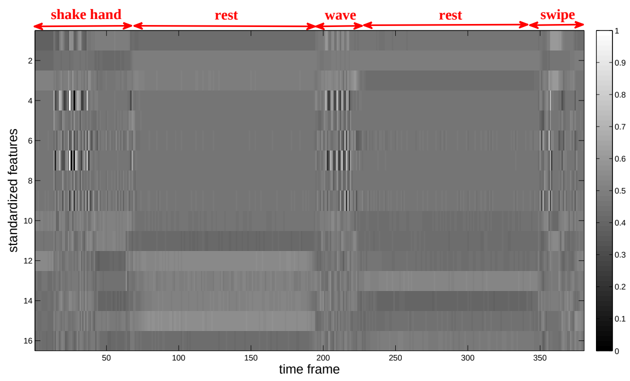

Figure 3.

Figure [?] shows the model represented in a dynamic Bayesian network (DBN). It is closely related to the hierarchical hidden Markov model [8]. Node G_t represents the gesture that the user is making at time t. It includes different gesture phases the system can recognize. Node S_t is the hidden state of the hand movement, which is essentially a vector quantization of the actual, observed (but noisy) feature vector X_t. Node F_t^G is a binary variable that indicates the end of a gesture. It is "on" (has value 1) if the lower level HMM at time t has just "finished", otherwise it is "off" (value 0). G_{t+1} can only change value from G_t if F_t^G is "on". Note that G_t, S_t, and F_t^G take discrete values, so the conditional probability distributions (CPDs) for them can take a tabular form. X_t is a vector with continuous values and we use a Gaussian distribution for its CPD, i.e.,

$$P(X_t = x_t | S_t = i) = N(x_t; \mu_i, \Sigma_i) \tag{5}$$

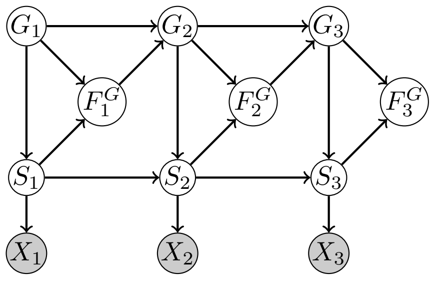

Figure 4. Gray color indicates observed nodes.

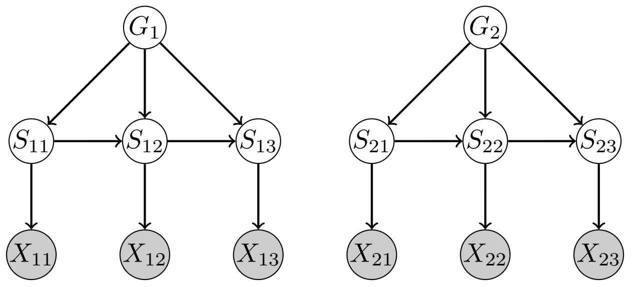

Figure 5. A mixture of HMMs [8].

During the training process, we give the model sequences of difference gestures with labeled G_t and F_t^G that correspond to the observed feature vectors X_t, but S_t is hidden. There can be different gestures in one sequence. In this way, the model can learn the termination probability given a hidden state and a gesture phase, i.e., $P(F_t^G = 1|G_t = g, S_t = i)$, and the transition probability between two hidden states given the gesture phase, i.e., $P(S_t = j|G_{t-1} = g, S_{t-1} = i)$. Note that the transition probabilities learned are for both within a gesture and between gestures.

Since S_t is hidden, we use expectation maximization (EM) algorithms to learn the maximum likelihood estimate of the model parameters for all the CPDs. The performance of EM can be highly dependent on how the algorithm is initialized [4] and it tends to converge to a local maximum of the observed data likelihood. As we model the $P(X_t|S_t)$ as a Gaussian distribution, we can view X_t as a mixture of Gaussians. The number of states S_t can take is the number of mixtures (i.e., clusters). As a result, we perform an unsupervised clustering analysis on a sub-sample of training data and use the Bayesian information criterion (BIC) [6] to find the optimal number of clusters.

Then we set the number of state for S_t to be the number of clusters, and initialize μ_i to be the cluster centers. We have tested with different clustering methods, including k-means, and mixture of Gaussians with k-means initialization, multiple random initializations, or hierarchical clustering initialization [**?**]. We found that the hierarchical clustering initialization used in the `mclust` package in R language gives the best final recognition accuracy result.

Online Inference

EXPERIMENTAL EVALUATION

Data Set

Feature detector	**Accuracy (std.)**	**F1 (std.)**
Dense sampling upper body	68.0% (4)	57.7% (8)
Skin & skeleton	72.1% (16)	64.7% (11)
Salience	**76.7%** (4)	**69.3%** (6)

Table 1.

Hand pose feature	**Accuracy (std.)**	**F1 (std.)**
No hand pose feature	76.5% (5)	68.4% (7)
Use hand pose feature	**76.7%** (4)	**69.3%** (6)

Table 2.

data set has both lighting conditions: normal and low intensity. IMU sensors, not used. Intrusive. 13 gesture phases and a random guess accuracy would be 7.8%.[2]

8 users

Comparison of Feature Detection Methods

Comparison of Recognition Training Methods

FUTURE WORK

CONCLUSION

REFERENCES

1. Bradski, G. R. Computer vision face tracking for use in a perceptual user interface, 1998.
2. Cutler, R., and Turk, M. View-based interpretation of real-time optical flow for gesture recognition. In *Proc. Automatic Face and Gesture Recognition* (1998), 416–421.
3. Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1 (2005), 886–893 vol. 1.
4. Dicintio, S. Comparing approaches to initializing the expectation-maximization algorithm. Master's thesis, The University of Guelph, 2012.
5. Fei-Fei, L., and Perona, P. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2 (2005), 524–531.
6. Fraley, C., Raftery, A. E., Murphy, T. B., and Scrucca, L. mclust version 4 for r: Normal mixture modeling for model-based clustering, classification, and density estimation. Tech. rep., Technical Report, 2012.
7. Marcos-Ramiro, A., Pizarro-Perez, D., Marron-Romera, M., Nguyen, L. S., and Gatica-Perez, D. Body communicative cue extraction for conversational analysis. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition* (2013).
8. Murphy, K. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, 2002.

[2] **`https://project.eia-fr.ch/chairgest/Pages/Overview.aspx`**

Training method	**Accuracy (std.)**	**F1 (std).**
HMM	50.6% (17)	55.6% (8)
AHMM	**76.7%** (4)	**69.3%** (6)

Table 3.

Inference	**Accuracy (std.)**	**F1 (std).**
Offline	76.7% (4)	69.3% (6)
Online (lag = 16 frames)	**76.9%** (3)	**65.2%** (9)

Table 4.

9. Russell, S. J., and Norvig, P. *Artificial intelligence: a modern approach*. Prentice-Hall, Inc., 2003.

10. Wang, H., Ullah, M. M., Klaser, A., Laptev, I., Schmid, C., et al. Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference* (2009).

11. Welsh, J., and Stewart, R. Kinect for xbox 360: Technology + two applications. `http://www.cs.virginia.edu/~jfw3x/cs4501/`, May 2011.