Continuous Body and Hand Gesture Recognition for Natural Human-Computer Interaction

YALE SONG, DAVID DEMIRDJIAN, and RANDALL DAVIS,
Massachusetts Institute of Technology

Intelligent gesture recognition systems open a new era of natural human-computer interaction: Gesturing is instinctive and a skill we all have, so it requires little or no thought, leaving the focus on the task itself, as it should be, not on the interaction modality. We present a new approach to gesture recognition that attends to both body and hands, and interprets gestures continuously from an unsegmented and unbounded input stream. This article describes the whole procedure of continuous body and hand gesture recognition, from the signal acquisition to processing, to the interpretation of the processed signals.

Our system takes a vision-based approach, tracking body and hands using a single stereo camera. Body postures are reconstructed in 3D space using a generative model-based approach with a particle filter, combining both static and dynamic attributes of motion as the input feature to make tracking robust to self-occlusion. The reconstructed body postures guide searching for hands. Hand shapes are classified into one of several canonical hand shapes using an appearance-based approach with a multiclass support vector machine. Finally, the extracted body and hand features are combined and used as the input feature for gesture recognition. We consider our task as an online sequence labeling and segmentation problem. A latent-dynamic conditional random field is used with a temporal sliding window to perform the task continuously. We augment this with a novel technique called multilayered filtering, which performs filtering both on the input layer and the prediction layer. Filtering on the input layer allows capturing long-range temporal dependencies and reducing input signal noise; filtering on the prediction layer allows taking weighted votes of multiple overlapping prediction results as well as reducing estimation noise.

We tested our system in a scenario of real-world gestural interaction using the NATOPS dataset, an official vocabulary of aircraft handling gestures. Our experimental results show that: (1) the use of both static and dynamic attributes of motion in body tracking allows statistically significant improvement of the recognition performance over using static attributes of motion alone; and (2) the multilayered filtering statistically significantly improves recognition performance over the nonfiltering method. We also show that, on a set of twenty-four NATOPS gestures, our system achieves a recognition accuracy of 75.37%.

Categories and Subject Descriptors: I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Motion*; I.5.5 [**Pattern Recognition**]: Implementation—*Interactive systems*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Pose tracking, gesture recognition, human-computer interaction, online sequence labeling and segmentation, conditional random fields, multilayered filtering

ACM Reference Format:
Song, Y., Demirdjian, D., and Davis, R. 2012. Continuous body and hand gesture recognition for natural human-computer interaction. ACM Trans. Interact. Intell. Syst. 2, 1, Article 5 (March 2012), 28 pages.
DOI = 10.1145/2133366.2133371 http://doi.acm.org/10.1145/2133366.2133371

This work was funded in part by the Office of Naval Research Science of Autonomy program, contract no. N000140910625, and in part by the National Science Foundation grant no. IIS-1018055.
Authors' addresses: Y. Song (corresponding author), D. Demirdjian, and R. Davis, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar St., Cambridge, MA 02139; email: yalesong@csail.mit.edu.

DOI 10.1145/2133366.2133371 http://doi.acm.org/10.1145/2133366.2133371

1. INTRODUCTION

For more than 40 years, human-computer interaction has been focused on the keyboard and mouse. Although this has been successful, as computation becomes increasingly mobile, embedded, and ubiquitous, it is far too constraining as a model of interaction. Evidence suggests that gesture-based interaction is the wave of the future, with considerable attention from both the research community (see recent survey articles by Mitra and Acharya [2007] and by Weinland et al. [2011]) and from the industry and public media (e.g., Microsoft Kinect). Evidence can also be found in a wide range of potential application areas, such as medical devices, video gaming, robotics, video surveillance, and natural human-computer interaction.

Gestural interaction has a number of clear advantages. First, it uses equipment we always have on hand: there is nothing extra to carry, misplace, or leave behind. Second, it can be designed to work from actions that are natural and intuitive, so there is little or nothing to learn about the interface. Third, it lowers cognitive overhead, a key principle in human-computer interaction: Gesturing is instinctive and a skill we all have, so it requires little or no thought, leaving the focus on the task itself, as it should be, not on the interaction modality.

Current gesture recognition is, however, still sharply limited. Most current systems concentrate on one source of input signal, for example, body or hand. Yet human gesture is most naturally expressed with both body and hands: Examples range from the simple gestures we use in everyday conversations, to the more elaborate gestures used by baseball coaches giving signals to players, soldiers gesturing for tactical tasks, and police giving signals to drivers. Considering only one source of signal (e.g., body or hand) severely restricts the expressiveness of the gesture vocabulary and makes interaction far less natural.

Gesture recognition can be viewed as a task of statistical sequence modeling: Given example observation sequences, the task is to learn a model that captures spatio-temporal patterns in the sequences, so that the model can perform sequence labeling and segmentation on new observations. One of the main challenges here is the task of online sequence segmentation. Most current systems assume that signal boundaries and/or the length of the whole sequence are known a priori. However, interactive gesture understanding should be able to process continuous input seamlessly, that is, with no need for awkward transitions, interruptions, or indications of boundaries between gestures. We use the terms *unsegmented* and *unbounded* to clarify what we mean by continuous input. Continuous input is unsegmented, that is, there is no indication of signal boundaries, such as the gesture start and end. Continuous input is also unbounded, that is, the beginning and the end of the whole sequence are unknown, regardless of whether the sequence contains a single gesture or multiple gestures. This is unlike work in most other areas with continuous input. In speech recognition, for example, most systems rely on having signal segmentation (e.g., by assuming that silence of a certain length indicates the end of a sentence) and deal with bounded conversations (e.g., making an airline reservation). Interactive gesture understanding from input that is continuous (both unsegmented and unbounded) requires that sequence labeling and segmentation be done simultaneously with new observations being made.

This article presents a new approach to gesture recognition that tracks both body and hands, and combines the two signals to perform online gesture interpretation and segmentation continuously, allowing richer gesture vocabulary and more natural human-computer interaction. Our main contributions are threefold: a unified framework for continuous body and hand gesture recognition; a new error measure, based on Motion History Image (MHI) [Bobick and Davis 2001], for body tracking that captures dynamic attributes of motion; and a novel technique called *multilayered filtering* for robust online sequence labeling and segmentation.

We demonstrate our system on the NATOPS body and hand gesture dataset [Song et al. 2011b]. Our extensive experimental results show that examining both static and dynamic attributes of motion improves the quality of estimated body features, which in turn improves gesture recognition performance by 6.3%. We also show that our multilayered filtering significantly improves recognition performance by 15.78% when added to the existing latent-dynamic conditional random field model. As we show in Section 4, these improvements are statistically significant. We also show that our continuous gesture recognition system achieves a recognition accuracy of 75.37% on a set of twenty-four NATOPS gestures.

Section 1.1 gives an overview of our system; Section 1.2 reviews some of the most related work in pose tracking and gesture recognition, making distinctions to our work; Section 2 describes body and hand tracking; Section 3 describes continuous gesture recognition; and Section 4 shows experimental results. Section 5 concludes with a summary of contributions and suggesting directions for future work.

Some of the material presented in this article has appeared in earlier conference proceedings [Song et al. 2011a, 2011b]. Song et al. [2011a] described gesture recognition of segmented input. This article extends our previous work to the continuous input domain and presents a new approach to performing online gesture interpretation and segmentation simultaneously (Section 3.2). Body and hand tracking was described in Song et al. [2011b]. Here, we include a deeper analysis of the body tracking, evaluating the performance of an MHI-based error measure we introduced in Song et al. [2011b] (Section 4.4). None of the experimental results reported in this article has appeared in any of our earlier work. Song et al. [2011b] also introduced a body and hand gesture dataset; here we give an experiment protocol on a set of all twenty-four gestures in the NATOPS dataset, and report a recognition accuracy of 75.37% (Section 4.7).

1.1. System Overview

Figure 1 shows an overview of our system. The three main components are a 3D upper-body posture estimator, a hand shape classifier, and a continuous gesture recognizer.

In the first part of the pipeline, image preprocessing (Section 2.1), depth maps are calculated using images captured from a stereo camera, and the images are background subtracted using a combination of an offline trained codebook background model [Kim et al. 2005] and a "depth-cut" method.

For 3D body posture estimation (Section 2.2), we construct a generative model of the human upper-body, and fit the model to observations by comparing various features extracted from the model to corresponding features extracted from observations. In order to deal with body posture ambiguities that arise from self-occlusion, we examine both static and dynamic attributes of motion. The static attributes (i.e., body posture features) are extracted from depth images, while the dynamic attributes are extracted from MHI [Bobick and Davis 2001]. Poses are then estimated using a particle filter [Isard and Blake 1998].

For hand shape classification (Section 2.3), we use information from body posture estimation to make the hand tracking task efficient: Two small search regions are defined around estimated wrist joints, and our system searches for hands in only these regions. A multiclass SVM classifier [Vapnik 1995] is trained offline using manually-segmented images of hands. HOG features [Freeman et al. 1998; Dalal and Triggs 2005] are extracted from the images and used as an image descriptor.

In the last part, continuous gesture recognition (Section 3), we form the input feature by combining body and hand information. A Latent-Dynamic Conditional Random Field (LDCRF) [Morency et al. 2007] is trained offline using a supervised body and hand gesture dataset. The LDCRF with a temporal sliding window is used to perform online sequence labeling and segmentation simultaneously. We augment this with our

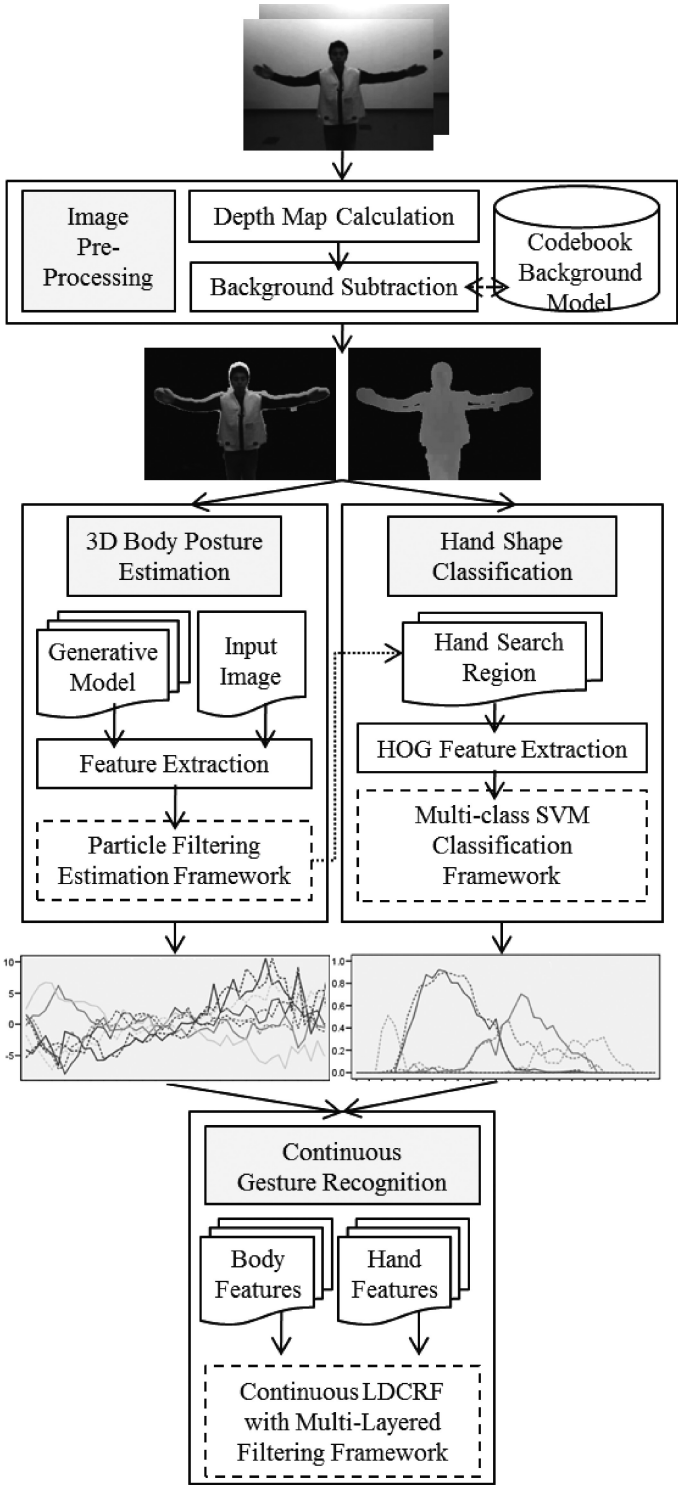


Fig. 1. A pipeline view of our unified framework for continuous body and hand gesture recognition.

multilayered filtering to make our task more robust. The multilayered filter acts both on the input layer and the prediction layer: On the input layer, a Gaussian temporal-smoothing filter [Harris 1978] is used to capture long-range temporal dependencies and make our system less sensitive to the noise from estimated time-series data, while not increasing the dimensionality of input feature vectors and keeping the model complexity the same. The prediction layer is further divided into local and global prediction layers, where we use a weighted-average filter and a moving-average filter, respectively, to take weighted votes of multiple overlapping prediction results as well as reduce noise in the prediction results.

1.2. Related Work

The topics covered in this article range broadly from body and hand tracking to gesture recognition with online sequence labeling and segmentation. This section reviews some of the most relevant work; comprehensive survey articles include Poppe [2007] for body tracking, Erol et al. [2007] for hand tracking, and Mitra and Acharya [2007] and Weinland et al. [2011] for gesture recognition.

Gesture-based interfaces typically require robust pose tracking. This is commonly done by wearing specially designed markers or devices (e.g., Vicon motion capture system or colored gloves [Yin and Davis 2010]). However, the most natural form of gestural interaction would not require additional markers or sensors attached to the body. We take a vision-based approach and perform motion tracking based on data from a single stereo camera, not using any special marker device attached to the body.

Several successful vision-based pose tracking approaches have been reported, falling generally into two categories: *model-based* methods, which try to reconstruct a pose model by fitting a kinematic model to the observed image [Deutscher et al. 2000; Sminchisescu and Triggs 2003; Lee and Cohen 2006]; and *appearance-based* methods, which assume a pose vocabulary and try to learn a direct mapping from features extracted from images to the vocabulary [Brand 1999; Shakhnarovich et al. 2003; Mori and Malik 2006]. Model-based methods are in general not affected by a camera viewpoint, do not require a training dataset, and are generally more robust in 3D pose estimation. Appearance-based methods require a large training dataset and in general are more sensitive to camera viewpoints, but once a mapping function is learned, classification is performed efficiently. Recent works take a hybrid approach, combining ideas from the two conventional methods and using advanced depth sensing cameras for 3D data acquisition (e.g., Time of Flight (ToF) [Gokturk et al. 2004] or structured light [Fofi et al. 2004]). Schwarz et al. [2011] use a ToF camera to obtain depth images. They detect anatomical landmarks to fit a skeleton body model, solving constrained inverse kinematics. A graph is constructed from the depth data, and geodesic distances between body parts are measured, making the 3D positions of anatomical landmarks invariant to pose. Similar to our work, they use optical flow between subsequent images to make tracking robust to self-occlusion. Shotton et al. [2011] obtain depth images from a structured light depth sensing camera (i.e., Microsoft Kinect). They take an object recognition approach: A per-pixel body part classifier is trained on an extensive training dataset. The results are reprojected onto 3D space, and local means are used to generate confidence-scored 3D proposals of body joints.

In this work, we take a model-based approach for body posture estimation, because reconstructing body posture in 3D space provides important information, such as pointing direction. Hand shapes, by contrast, are more categorical, that is, it is typically not crucial to distinguish fine-grained details of hand shape in order to understand a body and hand gesture. Therefore, we take an appearance-based approach to hand shape classification.

Fig. 2. Input image (left), depth map (middle), and mask image (right). The "T-pose" shown in the figures is used for body tracking initialization.

There have been active efforts to build a principled probabilistic graphical model for sequence modeling based on discriminative learning. Lafferty et al. [2001] introduced Conditional Random Fields (CRF), a discriminative learning approach that does not make conditional independence assumptions. Quattoni et al. [2007] introduced Hidden Conditional Random Fields (HCRF), an extension to the CRF that incorporates hidden variables. Many other variants of the CRF have been introduced since then [Sutton et al. 2004; Gunawardana et al. 2005; Wang and Mori 2009], but most of them could not handle continuous input, limiting their use in real-world applications.

Morency et al. [2007] presented a Latent-Dynamic Conditional Random Field (LDCRF) that is able to perform sequence labeling and segmentation simultaneously. An LDCRF assumes a disjoint set of hidden state variables per label, allowing it to do parameter estimation and inference efficiently using belief propagation [Pearl 1988]. They showed that the model is capable of capturing the substructure of a class sequence and can learn dynamics between class labels, allowing the model to perform sequence labeling and segmentation simultaneously. However, the forward-backward message-passing schedule used in belief propagation limited its use to bounded input sequences only. In this work, we use an LDCRF with a temporal sliding window to predict sequence labels and perform segmentation online, augmenting the original framework with our multilayered filtering, preserving the advantages of belief propagation and extending the previous work to the unbounded input domain.

2. OBTAINING BODY AND HAND SIGNALS

In this section, we describe body and hand tracking, which receives input images from a stereo camera and produces body and hand signals by performing 3D body posture estimation and hand shape classification.

We describe image preprocessing in Section 2.1, which produces depth maps and mask images (see Figure 2). We describe 3D body posture estimation in Section 2.2 and hand shape classification in Section 2.3.

2.1. Image Preprocessing

The system starts by receiving pairs of time-synchronized images recorded from a Bumblebee2 stereo camera, producing 320 x 240 pixel resolution images at 20 FPS. While recording video, the system produces depth maps and mask images in real time (see Figure 2). Depth maps allow us to reconstruct body postures in 3D space and resolve some of the pose ambiguities arising from self-occlusion; mask images allow us to concentrate on the objects of interest and ignore the background, optimizing the use of available computational resources. We obtain depth maps using a manufacture-provided SDK.[1]

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

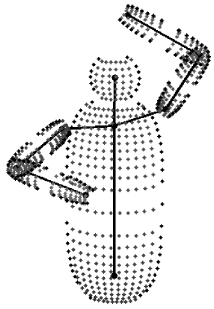


Fig. 3. Generative model of the human upper-body model. The model includes 6 body parts (trunk, head, upper and lower arms for both sides) and 9 joints (chest, head, navel, left/right shoulder, elbow, and wrist).

We obtain mask images by performing background subtraction. Ideally, background subtraction could be done using depth information alone by the "depth-cut" method: Filter out pixels whose distance is further from camera than a foreground object, assuming there is no object in between the camera and the subject. However, as shown in Figure 2, depth maps typically have lower resolution than color images, meaning that the resolution of mask images produced would be equally low resolution. This motivates our approach of performing background subtraction using a codebook approach [Kim et al. 2005], then refining the result with the depth-cut method.

The codebook approach works by learning a per-pixel background model from a history of 2D color background images sampled over a period of time, then segmenting out the "outlier" pixels in new images as foreground. Since this approach uses RGB images, it produces high-resolution mask images. One weakness of the codebook approach is, however, its sensitivity to illumination and shadows, arising because the codebook defines a foreground object as any set of pixels whose color values are noticeably different from the previously learned background model. To remedy this, after input images are background subtracted using the codebook approach, we refine the result using the depth-cut method described before.

2.2. 3D Body Posture Estimation

The goal here is to reconstruct upper-body posture in 3D space given the input images. We formulate this as a sequential Bayesian filtering problem, that is, having observed a sequence of images $\mathbf{Z}_t = \{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$ and knowing the prior state density $p(\mathbf{x}_t)$, make a prediction about a posterior state density $p(\mathbf{x}_t \mid \mathbf{Z}_t)$, where $\mathbf{x}_t = (x_{1,t} \ldots x_{k,t})$ is a k-dimensional vector representing the body posture we are estimating.

2.2.1. Generative Upper-Body Model. Our generative model of the human upper-body is constructed in 3D space, using a skeletal model represented as a kinematic chain and a volumetric model described by superellipsoids [Barr 1981] (see Figure 3). The model includes 6 body parts (trunk, head, upper and lower arms for both sides) and 9 joints (chest, head, navel, left/right shoulder, elbow, and wrist). The shoulder is modeled as a 3 DOF ball-and-socket joint, and the elbow is modeled as a 1 DOF revolute joint, resulting in 8 model parameters in total. Coordinates of each joint are obtained by solving the forward kinematics problem, following the Denavit-Hartenberg convention [Denavit and Hartenberg 1955], a compact way of representing n-link kinematic structures. We prevent the model from generating anatomically implausible body postures by constraining joint angles to known physiological limits [NASA 1995].

The human shoulder has historically been the most challenging part for human body modeling [Engin 1980]. It has a complicated anatomical structure, with bones,

muscles, skin, and ligaments intertwined, making modeling of shoulder movement difficult [Feng et al. 2008].

We improve on our basic model of the human upper-body by building a more precise model of the shoulder, while still not increasing the dimensionality of the model parameter vector. To capture arm movement more accurately, after a body model is generated, the shoulder model is refined analytically using the relative positions of other body joints.

Therefore, the generation of a body model is a two-step procedure: Given the eight joint angle values, we first solve the forward kinematics problem, obtaining the coordinates of each joint. Then we compute the angle φ between the chest-to-shoulder line and the chest-to-elbow line, and update the chest-to-shoulder angle θ^{CS} as[2]

$$\theta^{CS'} = \begin{cases} \theta^{CS} + \frac{\varphi}{\theta^{CS}_{MAX}} & \text{if elbow is higher than shoulder,} \\ \theta^{CS} - \frac{\varphi}{\theta^{CS}_{MIN}} & \text{otherwise,} \end{cases} \tag{1}$$

where θ^{CS}_{min} and θ^{CS}_{max} are minimum and maximum joint angle limits for chest-to-shoulder joints [NASA 1995]. Figure 3 illustrates our generative model, rendered after the chest-to-shoulder angles θ^{CS} are adjusted (note the left/right chest-to-shoulder angles are different). This simplified model mimics shoulder movement in only one dimension, up and down, but works quite well if the subject is facing the camera, as is commonly true for human-computer interaction.

With these settings, an upper-body posture is parameterized as

$$\mathbf{x} = (G\, R)^T, \tag{2}$$

where G is a 6-dimensional global translation and rotation vector, and R is an 8-dimensional joint angle vector (3 for shoulder and 1 for elbow, for each arm). In practice, once the parameters are initialized, we fix all but (x, z) translation elements of G, making $\mathbf{x}$ a 10-dimensional vector.

2.2.2. Particle Filter. Human body movements can be highly unpredictable, so an inference that assumes its random variables form a single Gaussian distribution can fall into a local minima or completely lose track. A particle filter [Isard and Blake 1998] is particularly well suited to this type of task for its ability to maintain multiple hypotheses during inference, discarding less likely hypotheses only slowly. It represents the posterior state density $p(\mathbf{x}_t \mid \mathbf{Z}_t)$ as a multimodal non-Gaussian distribution, which is approximated by a set of N weighted particles: $\{(\mathbf{s}_t^{(1)}, \pi_t^{(1)}), \ldots, (\mathbf{s}_t^{(N)}, \pi_t^{(N)})\}$. Each sample $\mathbf{s}_t$ represents a pose configuration, and the weights $\pi_t^{(n)}$ are obtained by computing the likelihood $p(\mathbf{z}_t \mid \mathbf{x}_t = \mathbf{s}_t^{(n)})$, and normalized so that $\sum_N \pi_t^{(n)} = 1$.

The joint angle dynamic model is constructed as a Gaussian process.

$$\mathbf{x}_t = \mathbf{x}_{t-1} + e, \quad e \sim \mathcal{N}(0, \sigma^2) \tag{3}$$

Once N particles are generated, we obtain the estimation result by calculating the Bayesian Least Squares (BLS) estimate.

$$\mathbb{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)}\ f\big(\mathbf{s}_t^{(n)}\big) \tag{4}$$

Iterative methods need a good initialization. We initialize our generative model at the first frame: The initial body posture configurations (i.e., joint angles and limb lengths)

[2]Note that the angle φ is not an additional model parameter, because it is computed analytically using joint positions.

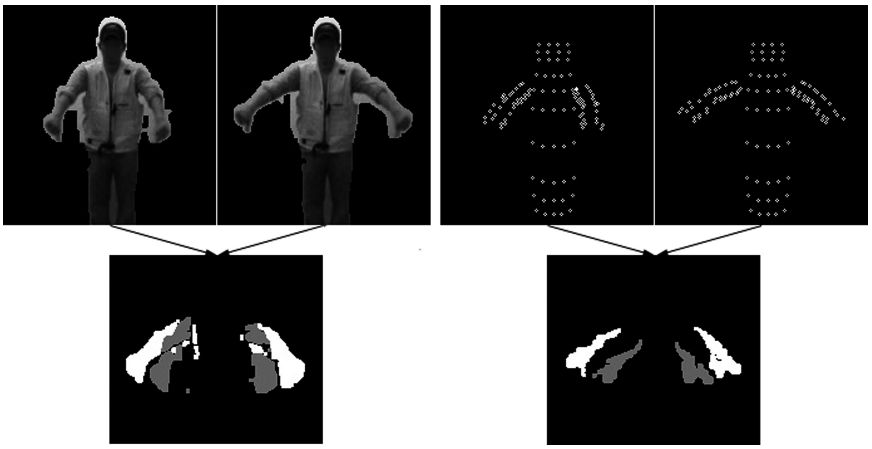


Fig. 4. Motion history images of the observation (left) and the estimated model (right). White pixel values indicate an object has appeared in the pixel; gray pixel values indicate there was an object in the pixel but it has moved; black pixel values indicate there has been no change in the pixel.

are obtained by having the subject assume a static "T-pose" (shown in Figure 2), and fitting the model to the image with exhaustive search. This typically requires no more than 0.3 seconds (on an Intel Xeon CPU 2.4 GHz machine with 4GBs of RAM).

2.2.3. Likelihood Function. The likelihood function $p(\mathbf{z}_t|\mathbf{x}_t = \mathbf{s}_t^{(n)})$ measures the goodness-of-fit of an observation $\mathbf{z}_t$ given a sample $\mathbf{s}_t^{(n)}$. We define it as an inverse of an exponentiated fitting error $\varepsilon(\mathbf{z}_t, \mathbf{s}_t^{(n)})$.

$$p\big(\mathbf{z}_t \mid \mathbf{x}_t = \mathbf{s}_t^{(n)}\big) = \frac{1}{\exp\big\{\varepsilon\big(\mathbf{z}_t, \mathbf{s}_t^{(n)}\big)\big\}} \tag{5}$$

The fitting error $\varepsilon(\mathbf{z}_t, \mathbf{s}_t^{(n)})$ is a weighted sum of three error terms computed by comparing features extracted from the generative model to the corresponding features extracted from input images. The three features include a 3D visible-surface point cloud, a 3D contour point cloud, and a Motion History Image (MHI) [Bobick and Davis 2001]. The first two features capture discrepancies in static poses; the third captures discrepancies in the dynamics of motion. We chose the weights for each error term empirically.

The first two features, 3D visible-surface and contour point clouds, are used frequently in body motion tracking (e.g., Deutscher et al. [2000]) for their ability to evaluate how well the generated body posture fits the actual pose observed in the image. We measure the fitting error by computing the sum-of-squared Euclidean distance errors between the point cloud of the model and the point cloud of the input image (i.e., the 3D data supplied by the image preprocessing step described earlier).

The third feature, an MHI, is an image where each pixel value is a function of the recency of motion in a sequence of images (see Figure 4). This often provides useful information about dynamics of motion, as it indicates where and how the motion has occurred. We define an MHI-based error term to measure discrepancies in the dynamics of motion.

An MHI is computed from I_{t-1} and I_t, two time-consecutive 8-bit unsigned integer images whose pixel values span from 0 to 255. For the generative model, I_t is obtained by rendering the model generated by a sample $\mathbf{s}_t^{(n)}$ (i.e., rendering an image of what body posture $\mathbf{s}_t^{(n)}$ would look), and I_{t-1} is obtained by rendering $\mathbb{E}[f(\mathbf{x}_{t-1})]$, the model generated by the estimation result from the previous step (Eq. (4)). For the input

images, I_t is obtained by converting an RGB input image to YCrCb color space and extracting the brightness channel[3], and this is stored to be used as I_{t-1} for the next time step. Then an MHI is computed as

$$I_{MHI} = \text{thresh}(I_{t-1} - I_t, 0, 127) + \text{thresh}(I_t - I_{t-1}, 0, 255), \tag{6}$$

where $\text{thresh}(I, \alpha, \beta)$ is a binary threshold operator that sets each pixel value to β if $I(x, y) > \alpha$, and zero otherwise. The first term captures pixels that were occupied at the previous time step but not in the current time step. The second term captures pixels that are newly occupied in the current time step. We chose the values 0, 127, and 255 to indicate the time information of those pixels: 0 means there has been no change in the pixel, regardless of whether or not there was an object; 127 means there was an object in the pixel but it has moved; while 255 means an object has appeared in the pixel. This allows us to construct an image that concentrates on the moved regions only (e.g., arms), while ignoring the unmoved parts (e.g., trunk, background). The computed MHI images are visualized in Figure 4.

Given the MHIs of the generative model and the observation, one can define various error measures. In this work, we define an MHI error as

$$\varepsilon_{MHI} = \mathbb{C}\text{ount}\,[\text{thresh}(I', 127, 255)], \tag{7}$$

where

$$I' = \text{abs}\big(I_{MHI}(\mathbf{z}_t, \mathbf{z}_{t-1}) - I_{MHI}\big(\mathbf{s}_t^{(n)}, \mathbb{E}\big[f(\mathbf{x}_{t-1})\big]\big)\big). \tag{8}$$

This error function first subtracts an MHI of the model $I_{MHI}(\mathbf{s}_t^{(n)}, \mathbb{E}[f(\mathbf{x}_{t-1})])$ from an MHI of the observation $I_{MHI}(\mathbf{z}_t, \mathbf{z}_{t-1})$, and computes an absolute-valued image of it (Eq. (8)). Then it applies the binary threshold operator with the cutoff value and result value (127 and 255, respective), and counts nonzero pixels with $\mathbb{C}\text{ount}\,[\cdot]$ (Eq. (7)). We set the cutoff value to 127 to penalize the conditions in which two MHIs do not match at the current time step, independent of the situation at the previous time step.[4]

We evaluate the effectiveness of the MHI-based error measure in Section 4.4, where we compare gesture recognition accuracy of the models trained on features estimated with and without the MHI-based error measure.

2.3. Hand Shape Classification

The goal of hand shape classification is to classify hand shapes made contemporaneously with gestures into one of several canonical hand shapes. We selected four hand shapes (thumb up and down, palm open and close) that are often used in hand signals, particularly on the NATOPS gestures used in this work (see Figure 5).

2.3.1. Search Region. As searching for hands in an entire image can be time consuming, we use the information about wrist position computed in body posture estimation to constrain the search for hands in the image. We create a small search region around each of the estimated wrist positions, slightly larger than the average size of an actual hand image, and search for a hand shape in that region using a sliding window. Estimated wrist positions are of course not always accurate, so a search region might not contain a hand. We compensate for this by including information on hand location from the previous step's hand shape classification result. If a hand is found at time $t-1$, for time t we center the search region at the geometric mean of the estimated wrist

[3]Empirically, most of the variation in images is better represented along the brightness axis, not the color axis [Bradski and Kaehler 2008].

[4]As mentioned, our error measure in Eq. (7) concentrates on errors at the current time step only. However, note that Eq. (6) also offers information on the errors at the previous time step as well.

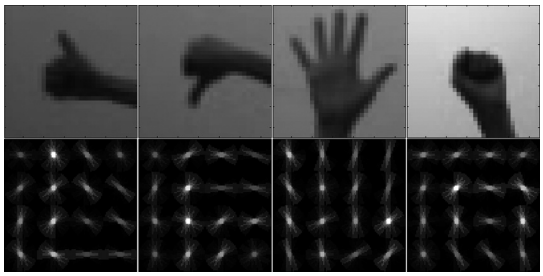


Fig. 5. Four canonical hand shapes defined in this work (thumb up and down, palm open and closed), and visualization of their HOG features. The four hand shapes are selected from the NATOPS dataset [Song et al. 2011b]. HOG features are computed with an image size of 32×32 pixels, cell size of 4×4 pixels, and block size of 2×2 cells (8×8 pixels), with 9 orientation bins. This results in 16 blocks in total. Bright spots in the visualization indicate places in the image that have sharp gradients at a particular orientation; the orientation of the spot indicates orientation of the gradients.

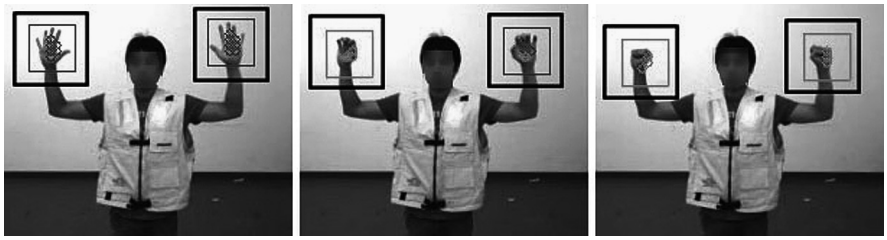


Fig. 6. Search regions around estimated wrist positions (outer rectangles) and clustering of multiple classification results. Our search region was 56×56 pixels (outer rectangles); the sliding window was 32 x 32 pixels (inner rectangles). Inner rectangles indicate clustered results (blue/red: palm open/closed), and small circles are individual classification results (best viewed in color).

position and the hand position at time $t - 1$. Our search region was 56×56 pixels; the sliding window was 32×32 pixels (see Figure 6).

2.3.2. HOG Features. HOG features [Freeman et al. 1998; Dalal and Triggs 2005] are image descriptors based on dense and overlapping encoding of image regions. The central assumption of the method is that the appearance of an object is rather well characterized by locally collected distributions of intensity gradients or edge orientations, even without having the knowledge about the corresponding gradient or edge positions that are globally collected over the image.

HOG features are computed by dividing an image window into a grid of small regions (cells), then producing a histogram of the gradients in each cell. To make the features less sensitive to illumination and shadowing effects, the same image window is again divided into a grid of larger regions (blocks), and all the cell histograms within a block are accumulated for normalization. The histograms over the normalized blocks are referred to as HOG features. We used a cell size of 4×4 pixels, block size of 2×2 cells (8×8 pixels), window size of 32×32 pixels, with 9 orientation bins. Figure 5 shows a visualization of the computed HOG features.

2.3.3. Multiclass SVM Classifier. To classify the HOG features, we trained a multiclass SVM classifier [Vapnik 1995] using LIBSVM [Chang and Lin 2011], with 5 classes (i.e., the four canonical hand poses plus "no hand"). Since HOG features are high dimensional, we used an RBF kernel to transform input data into the high-dimensional feature space. We trained a multiclass SVM following the one-against-one method [Knerr et al. 1990] for fast training, while obtaining comparable accuracy to the

one-against-all method [Hsu and Lin 2002]. We performed a grid search and 10-fold cross-validation for parameter selection.

2.3.4. Training Dataset. To train the SVM classifier, a training dataset was collected from the NATOPS dataset, choosing the recorded video clips of the first 10 subjects (out of 20). Positive samples (the four hand poses) were collected by manually selecting 32×32 pixel images that contained hands and labeling them; negative samples ("no hand") were collected automatically after collecting positive samples, by choosing two random foreground locations and cropping the same-sized images. We applied affine transformations to the positive samples, to make the classifier more robust to scaling and rotational variations, and to increase and balance the number of samples across hand shape classes. After applying the transformations, the size of each class was balanced at about 12,000 samples.

2.3.5. Clustering. Each time a sliding window moves to a new position within a search region, the HOG features are computed, and the SVM classifier examines them, returning a vector of $k+1$ probability estimates (k hand classes plus one negative class; $k = 4$ in our current experiments). We thus get multiple classification results per search region, with one from each sliding window position. To get a single classification result per search region, we cluster all positive classification results (i.e., classified into one of the k positive classes) within the region, averaging positions and probability estimates of the results (see Figure 6).

2.4. Output Features

From the body and hand tracking described before, we get a 12-dimensional body feature vector and an 8-dimensional hand feature vector. The body feature vector includes 3D joint velocities for left/right elbows and wrists. To obtain this, we first generate a model with the estimated joint angles and fixed-length limbs, so that all generated models have the same set of limb lengths across subjects. This reduces cross-subject variances resulting from different limb lengths. Then we log coordinates of the joints relative to the chest, and take the first-order derivatives.

The hand feature includes probability estimates of the four predefined hand poses for left/right hand, dropping the fifth class "no hand" (because as with any set of probabilities that sum to one, N-1 values are enough).

We turn next to continuous gesture recognition that uses the body and hand features described.

3. GESTURE RECOGNITION

The goal here is to perform online sequence labeling and segmentation simultaneously, given a continuous stream of body and hand feature signals that are automatically tracked using the methods we described. For each image $\mathbf{z}_t$, we extract body posture features $\mathbf{x}_t^{(B)} \in \mathbb{R}^{N(B)}$ (Section 2.2) and hand shape features $\mathbf{x}_t^{(H)} \in \mathbb{R}^{N(H)}$ (Section 2.3). To form an input feature vector, we concatenate the body and hand features; that is, each $\mathbf{x}_t$ is represented as a dual-signal feature vector

$$\mathbf{x}_t = \left(\mathbf{x}_t^{(B)} \;\; \mathbf{x}_t^{(H)}\right)^T. \tag{9}$$

To state the problem more precisely: Given a continuous stream of input feature vectors $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$, our task is to infer the label sequence $\mathbf{y} = \{y_1, \ldots, y_t\}$, where each $y_t \in \mathcal{Y}$ is a member of a finite class label set.

Our gesture recognizer is based on work by Morency et al. [2007], where they presented a Latent-Dynamic Conditional Random Field (LDCRF) that is able to perform sequence labeling and segmentation simultaneously from bounded input. We extend

the LDCRF to work with unbounded input, without changing the efficient inference method (i.e., belief propagation). We define a temporal sliding window and perform online gesture recognition, augmenting the model with multilayered filtering. We describe our LDCRF model (Section 3.1) and detail the multilayered filtering with a temporal sliding window (Section 3.2).

3.1. Latent-Dynamic CRFs

An LDCRF [Morency et al. 2007] is a discriminative graphical model with latent variables that is able to learn both internal dynamics (within-class structure) and external dynamics (between-class structure) of sequences. Once it learns the internal and external dynamics successfully, it can label a sequence of multiple instances (e.g., multiple gesture sequences stitched together), allowing it to perform sequence segmentation by looking at discontinuities in the predicted labels.

An LDCRF represents the sequence dynamics as a tree-structured graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of vertices $\mathcal{V}$ representing random variables (observed or hidden) and a set of edges $\mathcal{E}$ representing dependencies between random variables. The random variables include the observation sequence $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$, the label sequence $\mathbf{y} = \{y_1, \ldots, y_t\}$, and the hidden state sequence $\mathbf{h} = \{h_1, \ldots, h_t\}$ that models the underlying structure of the sequence dynamics. The hidden state sequence forms a linear chain structure, with two edges from each hidden state variable h_t to a label variable y_t and to an observation variable $\mathbf{x}_t$ at each time frame.

The posterior probability distribution $p(\mathbf{y} \mid \mathbf{x};\ \mathbf{\Lambda})$ of the label sequence $\mathbf{y}$ given the observation sequence $\mathbf{x}$ is constructed as

$$p(\mathbf{y} \mid \mathbf{x};\ \mathbf{\Lambda}) = \sum_{\mathbf{h}} p(\mathbf{y} \mid \mathbf{h}, \mathbf{x};\ \mathbf{\Lambda})\, p(\mathbf{h} \mid \mathbf{x};\ \mathbf{\Lambda}), \tag{10}$$

where $\mathbf{\Lambda} = (\lambda, \omega)$ is a set of real-valued model parameters to be estimated. In order to make the computation tractable, an LDCRF assumes a disjoint set of hidden state variables $h \in \mathcal{H}_y$ per class label y, which makes $p(\mathbf{y} \mid \mathbf{h}, \mathbf{x};\ \Lambda) = 0$ for $h \notin \mathcal{H}_y$. Therefore, Eq. (10) becomes

$$p(\mathbf{y} \mid \mathbf{x};\ \mathbf{\Lambda}) = \sum_{\mathbf{h}:\forall h \in \mathcal{H}_y} p(\mathbf{h} \mid \mathbf{x};\ \mathbf{\Lambda}). \tag{11}$$

The conditional distribution $p(\mathbf{h} \mid \mathbf{x};\ \mathbf{\Lambda})$ of the hidden state sequence $\mathbf{h}$ given the observation sequence $\mathbf{x}$ in Eq. (11) is modeled as

$$p(\mathbf{h} \mid \mathbf{x};\ \mathbf{\Lambda}) = \frac{1}{Z} e^{\mathbf{\Lambda}^\intercal \cdot \Phi(\mathbf{h}, \mathbf{x})}, \tag{12}$$

where $\Phi(\mathbf{h}, \mathbf{x})$ is a *potential* function and $Z = \sum_{\mathbf{h}} e^{\mathbf{\Lambda}^\intercal \cdot \Phi(\mathbf{h}, \mathbf{x}; \mathbf{\Lambda})}$ is a *partition* function for normalization. The potential function should be designed carefully to capture complex dependencies in the input sequence while making the computation tractable. The potential function is factorized with feature functions $f_k(\cdot)$ and $g_k(\cdot)$ as

$$\mathbf{\Lambda}^\intercal \cdot \Phi(\mathbf{h}, \mathbf{x}) = \sum_{t \in \mathcal{V}} \sum_k \lambda_k f_k(h_t, \mathbf{x}) + \sum_{(s,t) \in \mathcal{E}} \sum_k \omega_k g_k(h_s, h_t, \mathbf{x}). \tag{13}$$

The first term $\lambda_k f_k(\cdot)$ represents singleton potentials defined over a single hidden variable $h_t \in \mathcal{V}$; the second term represents pairwise potentials defined over a pair of hidden variables $(h_s, h_t) \in \mathcal{E}$. The feature function $f_k(\cdot)$ captures the relationship between a hidden variable $h_t \in \mathcal{H}$ and observations $\mathbf{x}$, and is of the dimension $|\mathcal{H}| \times |\mathbf{x}|$, where $|\mathcal{H}| = |\mathcal{Y}| \times |\mathcal{H}_y|$. The feature function $g_k(\cdot)$ captures the relationship between a pair of hidden variables $(h_s, h_t) \in \mathcal{H}$, and is of the dimension $|\mathcal{H}| \times |\mathcal{H}|$. For example, with

10 class labels, 5 hidden states per label, and 20-dimensional input feature vector, there are 1000 model parameters λ_k for $f_k(\cdot)$ and 2500 model parameters ω_k for $g_k(\cdot)$.

Given a training dataset $\mathcal{D} = \{\mathbf{y}_i, \mathbf{x}_i\}_{i=1}^N$, we find the optimal model parameters $\mathbf{\Lambda}^* = (\lambda^*, \omega^*)$ by optimizing a conditional log-likelihood objective function using gradient descent, specifically L-BFGS [Nocedal and Wright 1999], which has been shown to outperform other optimization algorithms for similar sequential models [Sutton et al. 2004]. Our objective function is defined as

$$L(\mathbf{\Lambda}) = \sum_{i=1}^{N} \log p(\mathbf{y}_i \mid \mathbf{x}_i; \mathbf{\Lambda}) - \frac{1}{2\sigma^2}||\mathbf{\Lambda}||^2, \tag{14}$$

where the second term is the Gaussian prior with variance σ^2 that is introduced to prevent overfitting of the training data. The gradient of $L(\mathbf{\Lambda})$ is obtained by taking the first-order partial derivatives of Eq. (14) with respect to λ_k and ω_k. See Morency et al. [2007] for the derivation. The function error for the optimization is defined as

$$E(\mathbf{\Lambda}) = \sum_{i=1}^{N} \log Z(\mathbf{h} \mid \mathbf{x}_i; \mathbf{\Lambda}) - \log Z(\mathbf{h}' \mid \mathbf{x}_i; \mathbf{\Lambda}), \tag{15}$$

where $\mathbf{h}'$ is the *mask* hidden states that sets any transition probability between different class labels to zero, and $Z(\cdot)$ is the partition function we described earlier. This definition of the function error effectively forces parameter learning to follow the LDCRF assumption, that is, that there is a disjoint set of hidden states per class, by giving higher function error scores to a parameter space that violates this assumption.

For efficient inference, belief propagation [Pearl 1988] is used to compute the marginal probabilities $p(h_t = a \mid \mathbf{x};\ \mathbf{\Lambda})$ and $p(h_s = a, h_t = b \mid \mathbf{x};\ \mathbf{\Lambda})$ and the partition Z that are necessary to compute gradients and function errors described before.

Prediction is performed on a per-frame basis by finding the most probable label sequence $\mathbf{y}^*$, given an input sequence $\mathbf{x}$.

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{Y}} \sum_{\mathbf{h} \in \mathcal{H}_y} p(\mathbf{h} \mid \mathbf{x};\ \mathbf{\Lambda}^*) \tag{16}$$

3.2. Temporal Sliding Window and Multilayered Filtering

While belief propagation allows efficient inference, it requires that the input sequence be bounded, because it is based on a forward-backward message-passing schedule. This limits LDCRFs to work only on bounded input. Various algorithms have been introduced that allow online prediction of continuous input sequences (e.g., forward-only inference algorithm [Murphy 2002]). However, to maintain the advantages of belief propagation, that is, robust and efficient inference, we follow the original implementation of the LDCRF and use a sliding window to perform prediction online.

To make sequence labeling and segmentation more robust, we augment this temporal sliding window approach with the multilayered filter. We define three layers of filters: in the input layer, we define a Gaussian temporal-smoothing filter; in the local prediction layer we define a weighted-average filter; finally, in the global prediction layer we define a moving-average filter. We first describe filtering on the input layer, then the temporal sliding window, and filtering on the local and global prediction layers.

3.2.1. Gaussian Temporal-Smoothing Filter on the Input Layer. Learning temporal patterns from estimated input sequences can be quite challenging due to the long-range dependencies among observations and the low Signal-to-Noise Ratio (SNR). The input signal patterns tend to exhibit long-range temporal dependencies, for example, body

parts move coherently as time proceeds, hand shapes are articulated in relation to body postures in a time-sequence, etc. Because body and hand signals are obtained via statistical estimation, the signals also exhibit high-frequency fluctuations (i.e., noise).

To capture long-range dependencies in the hidden CRF formulation, Quattoni et al. [2007] defined a temporal window and concatenated signals within the window, creating a single large input feature. However, this increases the number of model parameters and does not deal with noisy input.

Instead, we use a Gaussian temporal-smoothing filter to compute a weighted mean of neighboring input features, not only capturing long-range dependencies but also making the model less sensitive to noise. This approach has an advantage of keeping the dimensionality of input feature vectors unchanged, hence not increasing the model complexity.

The Gaussian filter defines a normalized w-point weight window $g(w)$ and performs a convolution of the input signals with $g(w)$. The weight window $g(w)$ is computed from

$$g(w)[n] = e^{-\frac{1}{2}\left(\alpha \frac{n}{w/2}\right)^2}, \tag{17}$$

where $-\frac{w-1}{2} \leq n \leq \frac{w-1}{2}$, and α is inversely proportional to the standard deviation of a Gaussian distribution.[5] Intuitively, the Gaussian kernel computes for each time frame a weighted mean of w neighboring feature vectors. This enables the computed feature vector at each time frame to incorporate long-range observations as well as to reduce signal noise.

Song et al. [2011a] showed that this approach significantly improves recognition accuracy on segmented input. In this work, we evaluate this approach on the continuous input (Section 4.5).

3.2.2. Temporal Sliding Window. In order to perform online sequence labeling and segmentation, we define a k-point temporal sliding window: At each time t, a k-point window slides forward, and an LDCRF evaluates a sequence of k frames $\mathbf{x}_{j:t} = \{\mathbf{x}_{j=t-k+1}, \ldots, \mathbf{x}_t\}$ to predict a label sequence $\mathbf{y}_{j:t} = \{y_j, \ldots, y_t\}$, computing $p_t(\mathbf{y}_{j:t}|\mathbf{x}_{j:t}; \mathbf{\Lambda}^*)$ using Eq. (11). The prediction result can be viewed as a $|\mathcal{Y}|$-by-k matrix, where each column vector $p_t(y_i \mid \mathbf{x}_{j:t};\ \mathbf{\Lambda}^*)$ is a probability estimate of $|\mathcal{Y}|$ class labels for the i-th frame ($t-k+1 \leq i \leq t$). Figure 7 illustrates this, together with a multilayered filtering we describe next.

3.2.3. Filtering on the Prediction Layer. We divide the prediction layer into the local and global prediction layers. At the first layer, a local prediction $\bar{p}_t(y_j)$ is made for the first frame $\mathbf{x}_j$ in the current temporal sliding window (i.e., the tail edge) by computing a weighted average of k previous LDCRF prediction results for that frame $\mathbf{x}_j$ using a weight vector γ,[6]

$$\bar{p}_t(y_j) = \sum_{i=1}^{k} \gamma_i\ p_{t-i+1}(y_j \mid \mathbf{x}_{j-i+1:t-i+1};\ \mathbf{\Lambda}^*). \tag{18}$$

We have experimented with two types of weight functions: an unbiased uniform weight function ($\gamma_i = 1/k$) and a Gaussian weight function obtained using Eq. (17), where the weights are normalized so that $\sum_{i=1}^{k} \gamma_i = 1$. In our preliminary experiments, the uniform weight function performed slightly better than the Gaussian weight function, although the difference was negligible.

[5]Following Harris [1978], we set $\alpha = 2.5$.

[6]Since the size of the window is k, each frame is evaluated k times.

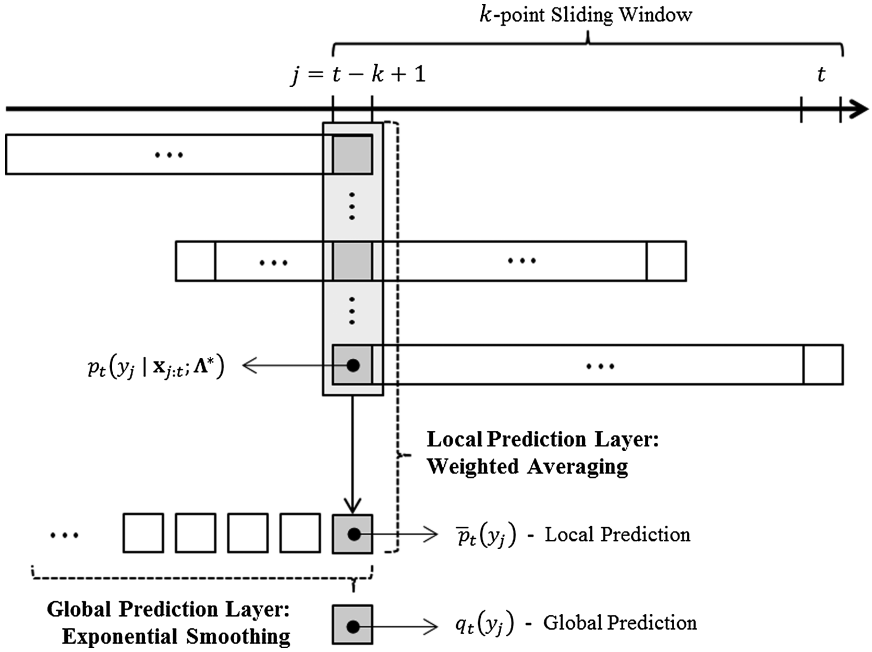


Fig. 7. A graphical illustration of the multilayered filtering acting on the prediction layer. As a k-point window slides forward, each individual frame is evaluated k times using an LDCRF model. At each time t, a label for the first frame in the window $\mathbf{x}_{j=t-k+1}$ is predicted based on the k previous LDCRF prediction results, using weighted averaging and exponential smoothing.

At the second layer, a global prediction $q_t(y_j)$ is made on the local prediction results using exponential smoothing. Our choice of the exponential smoothing is based on the intuition that the smoothing rate should be adaptive to how confident the local prediction is, that is, put less weight to the past if the current prediction is highly confident. Exponential smoothing naturally formulates this idea, by having one free parameter for setting the smoothing rate; we set this adaptively to the maximum marginal probability of the local prediction. To state more precisely, we compute the global prediction $q_t(y_j)$ as

$$q_t(y_j) = \alpha \cdot \bar{p}_t(y_j) \; + \; (1 - \alpha) \cdot q_{t-1}(y_{j-1}), \tag{19}$$

where α is the smoothing factor that determines the level of smoothing (larger values of α reduce the level of smoothing). We set the smoothing factor to the highest probability value in the local prediction result

$$\alpha = \max \bar{p}_t(y_j), \tag{20}$$

so that the more confident a local prediction is, the less smoothing performed.

Finally, sequence labeling is done by selecting a label with the highest probability

$$y_j^* = \arg\max_{y' \in \mathcal{Y}} q_t(y_j = y'). \tag{21}$$

Sequence segmentation is performed using the same method the LDCRF uses (i.e., a segment is determined when the predicted label changes over time). One difference is that our segmentation is performed using a history of global prediction results $q_t(y_j)$ instead of using an LDCRF's prediction result.

4. EXPERIMENTS

In this section, we report on experiments designed to evaluate the techniques we introduced in this article on a task of continuous body and hand gesture recognition. Specifically, we evaluate the effect of various model parameter variables on the system's performance in terms of the recognition accuracy and the training time. The model parameter variables include:

—the number of hidden variables per class ($|\mathcal{H}_y|$) (Section 3.1);
—the regularization factor (σ^2) (Section 3.1);
—the size of the Gaussian temporal-smoothing window (w) (Section 3.2.1);
—the size of the temporal sliding window (k) (Section 3.2.2).

We also evaluate the effectiveness of the MHI-based error measure for body posture estimation (see Section 2.2) by comparing the performance of the system trained on the features estimated with and without the MHI-based error measure.

For our experiments, we used the NATOPS dataset [Song et al. 2011b], a real-world body and hand gesture vocabulary used for aircraft handling scenarios.[7] We describe this dataset in Section 4.1, explain the experimental setting in Section 4.2, and detail each of the experiments subsequently. Finally, we give a fixed experiment protocol for the NATOPS dataset, and report recognition performances on continuous gesture recognition in Section 4.7.

4.1. Dataset

The Naval Air Training and Operating Procedures Standardization (NATOPS) is an official gesture vocabulary for the U.S. Navy aircraft carrier environment, which defines a variety of body and hand gestures that carrier flight deck personnel use to communicate with the pilots. The NATOPS dataset [Song et al. 2011b] contains twenty-four such gestures in the form of video clips and automatically tracked signals. For each gesture class there are 400 samples, performed by 20 subjects repeating 20 times (9,600 samples in total). The samples were recorded at 20 FPS, each sample varying between 30 and 60 frames (46 frames on average, equal to 2.3 seconds). The video clips include RGB color images, depth maps, and mask images in separate files. Automatically tracked signals include the 12-dimensional body feature and the 8-dimensional hand feature we extracted (see Section 2.4).

We selected six gestures from the NATOPS dataset for the experiments in this article (see Figure 8). They form three pairs that are difficult to distinguish without knowing both body and hand poses. For example, gesture #20 and #21 have the same body motion, while the hand motion is in opposite order, that is, palms open to closed in gesture #20, and vice versa in gesture #21. For a complete list of all twenty-four gestures, see Song et al. [2011b].

4.2. Methodology

Unless otherwise noted, all experiments were conducted using an N-fold cross-validation approach, where 1/N of the entire dataset is held out for testing, another 1/N held out for validation, with the remainder used for training. This is then repeated N times. In our experiment, N was set to 5; thus, each model was trained on the data from 12 subjects, validated on the data from 4 subjects, and tested on the data from the remaining 4 subjects. This was repeated 5 times.

The model parameters $\mathbf{\Lambda} = (\lambda, \omega)$ (see Eq. (10)) were initialized at random, with a manually selected random seed. For each set of experiments, where the goal was to see

[7]The dataset is publicly available at http://groups.csail.mit.edu/mug/natops/.

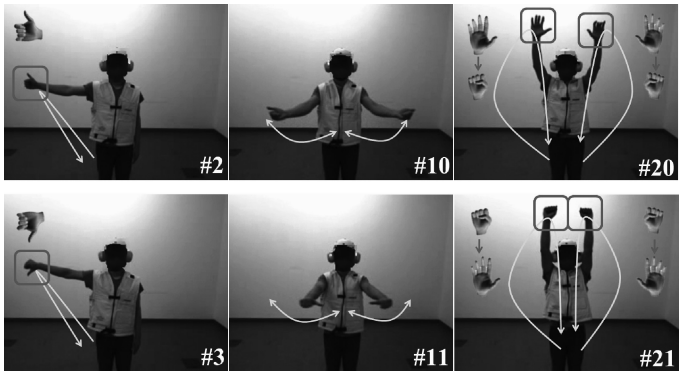


Fig. 8. Six NATOPS aircraft handling signal gestures. Body movements are illustrated in arrows, and hand poses are illustrated with synthesized images of hands. Rectangles indicate hand poses are important in distinguishing the gesture pair.

the effect of one particular variable (e.g., window size), we used the same random seed so that the initial values of $\mathbf{\Lambda}$ are the same as long as the number of model parameters $\mathbf{\Lambda}$ is the same. F1 scores ($= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$) were computed on a per-frame basis. The maximum number of iterations in the L-BFGS optimization was set to 1000.

When testing the LDCRF, we simulated a continuous input stream by concatenating the segmented gesture sequences in random order, setting the number of repetitions of each gesture randomly between one and five. Note, however, that we trained the LDCRF using segmented gesture sequences, setting the between-label transition functions (i.e., $g_k(h_s, h_t, \mathbf{x})$ for all $h_s \in \mathcal{H}_y$ and $h_t \in \mathcal{H}_{y'}, y \neq y'$ in Eq. (13)) to random real values. We speculate that training the LDCRF using unsegmented gesture sequences may result in better recognition performance; we plan to explore this in the future.

All experiments were conducted on a Matlab distributed computing cluster with 6 workers on 3 computers (2 workers per computer), with each computer having two six-core CPUs (Intel Xeon X5650, 2.67 GHz) and 64GBs of RAM.

4.3. Experiment 1: The Number of Hidden States and the Regularization Factor

Ideally, selecting the optimal values of model parameter variables (e.g., window size) should be based on cross-validation. However, performing cross-validation of all parameters in our system at once would be computationally challenging, since the number of test cases grows as the product of all model parameter variables. For simplicity, we first find the optimal number of hidden states per class ($|\mathcal{H}_y|$) and the optimal regularization factor (σ^2). Once we find the optimal values for $|\mathcal{H}_y|$ and σ^2, we fix these values in the subsequent experiments and concentrate on evaluating the effect of the other model parameter variables.

Note that, in order to focus on the effect of $|\mathcal{H}_y|$ and σ^2, we tested the *batch LDCRF* method [Morency et al. 2007] in this first experiment, that is, the test sequence was unsegmented but bounded, and the LDCRF evaluated the whole test sequence at once, not using a temporal sliding window.

Table I shows the experimental results, with the optimal parameter value selected as the best performing case on the validation split (shown in bold face). We show F1 scores of the test results on the training, validation, and test split. We also show the average F1 scores of the test results on the validation and test splits (Avg (V+T)), as both the validation split and test split contain unseen data during model training. All the F1 scores and training times reported in the table are average values from

Table I. Experiment 1 Results on Different Numbers of Hidden States per Class ($|\mathcal{H}_y|$) and the Regularization Factor (σ^2), Tested on the Batch LDCRF Method

Number Of Hidden States Per Class, $|\mathcal{H}_y|$ ($\sigma^2 = 0$)

$\lvert\mathcal{H}_y\rvert$	Train Set	Validate Set	Test Set	Avg (V+T)	Training Time	Params
3	.9213 (.02)	.8567 (.04)	.8407 (.05)	.8487 (.05)	71 m (5.24)	684
5	.9167 (.02)	.8264 (.06)	.8429 (.06)	.8346 (.06)	161 m (7.15)	1,500
7	**.9487 (.03)**	**.8749 (.03)**	**.8520 (.07)**	**.8634 (.05)**	**298 m (14.79)**	**2,604**
9	.9434 (.02)	.8594 (.05)	.8727 (.06)	.8660 (.05)	449 m (25.27)	3,996

Regularization Factor, σ^2 ($|\mathcal{H}_y| = 7$)

σ^2	Train Set	Validate Set	Test Set	Avg (V+T)	Training Time	Params
0	.9487 (.03)	.8749 (.04)	.8520 (.07)	.8634 (.05)	298 m (14.79)	2,604
1	.9580 (.01)	.8510 (.08)	.8591 (.07)	.8520 (.07)	216 m (43.79)	2,604
1000	**.9514 (.03)**	**.8785 (.04)**	**.8645 (.05)**	**.8645 (.05)**	**305 m (35.73)**	**2,604**

Recognition results are the mean F1 scores and their standard deviations, tested on the training, validation, and test split. On the experiment of $|\mathcal{H}_y|$, the regularization factor was set at 0 (i.e., no regularization). On the experiment of σ^2, $|\mathcal{H}_y|$ was set at 7 (optimal parameter value). The column "Avg (V+T)" shows average F1 scores of test results on the validation and the test split. The column "Params" shows the total number of model parameters ($\mathbf{\Lambda}$).

Table II. Experiment 2 Results on the Effectiveness of the MHI-Based Error Measure for Body Posture Estimation, Tested on the Batch LDCRF Method

MHI vs. no-MHI on Gesture Recognition ($|\mathcal{H}_y| = 7$, $\sigma^2 = 1{,}000$)

MHI	Train Set	Validate Set	Test Set	Avg (V+T)	Training Time	Params
O	**.9514 (.03)**	**.8785 (.04)**	**.8645 (.05)**	**.8715 (.04)**	**305 m (35.73)**	**2,604**
X	.9047 (.02)	.8234 (.02)	.8162 (.04)	.8198 (.03)	367 m (53.91)	2,604

In all experiments, $|\mathcal{H}_y| = 7$, $\sigma^2 = 1{,}000$. The difference of the average F1 scores between MHI and no-MHI was statistically significant: $t(18) = 3.00$, $p = 0.007$.

the 5-fold cross-validation. Although there were no significant differences among the conditions, the best performing $|\mathcal{H}_y|$ was 7, and the best performing σ^2 was 1,000 on the validation split. We use these values in the subsequent experiments.

4.4. Experiment 2: The MHI-Based Error Measure for Body Posture Estimation

In Section 2.2 we described an MHI-based error measure for body posture estimation, and claimed that using static and dynamic attributes of motion together improves estimating body postures. Here we demonstrate this claim, evaluating the effectiveness of the MHI-based error measure by comparing the recognition performance of the models trained using features estimated with and without the MHI-based error measure, which we refer to as "MHI" and "no-MHI", respectively. We set $|\mathcal{H}_y| = 7$ and $\sigma^2 = 1,000$ as found in our previous experiment.

Table II details the experimental results. The average F1 score (Avg (V+T)) of MHI was 6.3% higher than no-MHI (.0517 higher F1 score), and the difference was statistically significant ($t(18) = 3.00$, $p = 0.007$).[8]

4.5. Experiment 3: Gaussian Temporal-Smoothing Filter

In this experiment, we studied the effect of the Gaussian temporal-smoothing filter (Section 3.2.1), varying the size of the window from 0 (no smoothing) to 9 (19 frames, roughly a one-second-sized window). We also compared this approach to Quattoni et al. [2007], which forms the input feature vector by concatenating all observations in the window to capture long-range dependencies.

[8]Note that the T-tests we perform hereafter are based on per-frame classification results. Therefore, strictly speaking, the statistical independence assumption made in the T-test is violated; however, the T-test results are still informative, as is commonly reported in the literature.

Table III. Experiment 3 Results on Different Size of the Gaussian Temporal-Smoothing Window (w), Tested on the Batch LDCRF Method

Gaussian Temporal-Smoothing Window (w) ($\lvert\mathcal{H}_y\rvert = 7$, $\sigma^2 = 1{,}000$)						
w	Train Set	Validate Set	Test Set	Avg(V+T)	Training Time	Params
0	.9514 (.03)	.8785 (.04)	.8645 (.05)	.8715 (.04)	305 m (35.73)	2,604
1	.9409 (.02)	.8680 (.05)	.8393 (.07)	.8537 (.06)	397 m (28.48)	2,604
3	.9586 (.02)	.8815 (.05)	.8715 (.06)	.8763 (.06)	444 m (53.68)	2,604
5	**.9658 (.02)**	**.9048 (.04)**	**.8833 (.05)**	**.8940 (.05)**	**497 m (12.40)**	**2,604**
7	.9656 (.02)	.8853 (.04)	.8649 (.04)	.8751 (.04)	471 m (32.07)	2,604
9	.9682 (.01)	.8796 (.04)	.8816 (.05)	.8806 (.04)	496 m (43.49)	2,604

In all experiments, $\lvert\mathcal{H}_y\rvert = 7$ and $\sigma^2 = 1{,}000$. The optimal value of w was 5, which corresponds to 0.55 second sized window in 20 FPS.

Table III shows the experimental results on the effect of the Gaussian temporal-smoothing filter. The optimal value of the window size was 5 (11 frames, roughly a half-second). This result is consistent with our previous experiment on isolated gesture recognition [Song et al. 2011a] (i.e., the optimal size of the window was the same), indicating that it started losing some important local/high-frequency gesture information when the Gaussian window size was larger than half a second. Although there was no added complexity to the model in terms of the number of parameters, the training time did increase by roughly 200 mins as w was increased from 0 to 5. We believe this increase in the training time is mainly due to the optimization engine performing more iterations of computing the objective function (Eq. (14)), especially during the line search in the L-BFGS.

Table IV shows the experimental results on comparing the Gaussian Temporal-Smoothing Window (GTSW) to the Temporal-Concatenating Window (TCW) introduced in Quattoni et al. [2007]. Note that there was no 5-fold cross-validation in this test: the test split included data from the first five subjects, the validation split included data from the next five subjects, and the training split included the rest. As expected, the model complexity of the TCW increased linearly as the window size increased, while in the case of the GTSW it stayed the same (see the Params and the Training Time). The average F1 scores of the GTSW show similar pattern as in Table III. But surprisingly the average F1 scores of the TCW significantly decreased as the size of the window increased. We speculate that this is due to the increased model complexity with not enough training data. Especially when $w > 3$, we can see a serious model overfitting, resulting in a sharp decline of the recognition performance with an increased training time (see the graph in Table IV).

4.6. Experiment 4: Continuous LDCRF with Multilayered Filtering

In this experiment, we turn to the continuous LDCRF, that is, simultaneous online sequence labeling and segmentation on a continuous input stream (see Section 3.2.2). We study how sensitive the recognition performance is to the size of a temporal sliding window, especially when augmented with our multilayered filtering. We compare this to the continuous LDCRF without filtering and the batch LDCRF method. The continuous LDCRF without filtering makes predictions directly from Eq. (16), selecting the prediction result of the current time frame as the latest frame in the current temporal sliding window (i.e., the head edge). Therefore, continuous LDCRF without filtering does not have a fixed time delay.[9] The batch LDCRF makes predictions once the

[9]While using the centered frame with a $2 \times k$ window may result in better recognition performance, this introduces delay, as in our multilayered filtering approach. For this reason we compare our approach to this head-edge approach in our experiments.

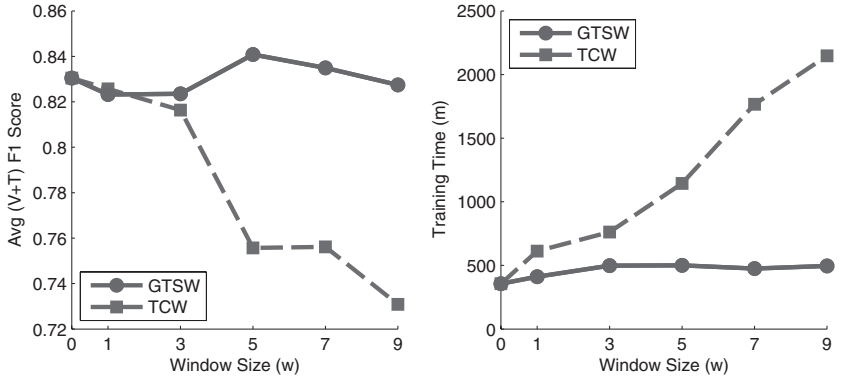
Table IV. Experiment 3 Results on Two Different Window Types on the Input Layer: Gaussian Temporal-Smoothing Window (GTSW) and Temporal-Concatenating Window (TCW), Both Using the Batch LDCRF Method

Gaussian Temporal Smoothing Window (GTSW)

ω	Train Set	Validate Set	Test Set	Avg(V+T)	Training Time	Params
0	.9684	.8341	.8268	.8304 (.01)	356 m	2,604
1	.9768	.8128	.8336	.8232 (.01)	498 m	2,604
3	.9668	.7897	.8573	.8235 (.05)	411 m	2,604
5	**.9869**	**.8542**	**.8275**	**.8408 (.02)**	**500 m**	**2,604**
7	.9835	.8390	.8308	.8349 (.01)	475 m	2,604
9	.9789	.8350	.8199	.8274 (.01)	495 m	2,604

Temporal Concatenating Window (TCW) [Quattoni et al. 2007]

ω	Train Set	Validate Set	Test Set	Avg(V+T)	Training Time	Params
0	**.9684**	**.8341**	**.8268**	**.8304 (.01)**	**356 m**	**2,604**
1	.9824	.8027	.8487	.8257 (.03)	612 m	4,284
3	.9733	.8204	.8123	.8163 (.01)	763 m	7,644
5	.9463	.7538	.7576	.7557 (.00)	1,144 m	11,004
7	.9327	.7327	.7796	.7562 (.03)	1,766 m	14,364
9	.8939	.6984	.7633	.7308 (.05)	2,147 m	17,724



In all experiments, $|\mathcal{H}_y| = 7$ and $\sigma^2 = 1{,}000$. The graphs show the average of test results on the validation and the test split. As expected, the model complexity of the TCW increased linearly as the window size increased, while that of our GTSW stayed the same. This resulted in a noticeable performance difference both in the average F1 score (bottom left) and the training time (bottom right).

observation is finished, hence the delay time is infinite when we deal with an unbounded stream of input.

In the experiment 3, we studied the Gaussian temporal-smoothing filter alone. Here we first investigate filtering on the prediction layer alone, varying the size of the temporal sliding window k from 20 to 80 (i.e., one- to four-second-sized window). Once we find the optimal window size, we fully operate the multilayered filter (i.e., filtering both on the input layer and the prediction layer) and compare it to the continuous LDCRF without filtering and the batch LDCRF method. Following the previous experiment, we set $|\mathcal{H}_y| = 7$ and $\sigma^2 = 1,000$ in all tests.

Figure 9 shows a qualitative comparison of sequence segmentation of the three approaches: the two continuous LDCRFs (with and without filtering on the prediction layer) and the batch LDCRF method. As seen in the figure, the continuous LDCRF without filtering on the prediction result fluctuates over time, while our method changes slowly, yielding a more robust sequence segmentation.

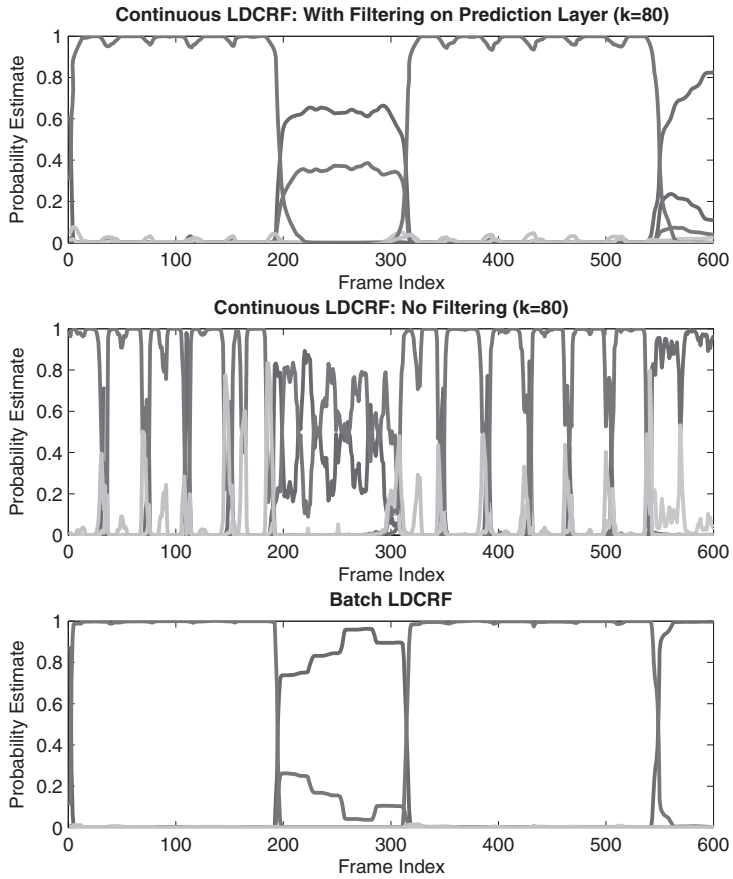


Fig. 9. Probability estimate plots from experiment 4, visually comparing sequence segmentation of the three approaches. Three sequence segmentation points are at roughly the 200th, the 310th, and the 550th frames. Line colors indicate different gesture classes. The continuous LDCRF without filtering (middle) fluctuates over time, resulting in inaccurate sequence segmentation. In contrast, our method (top) changes slowly, similar to the batch LDCRF (bottom), yielding a more robust sequence segmentation (best viewed in color).

Table V shows the experimental results on the continuous LDCRF with filtering on the prediction layer, compared to the continuous LDCRF without filtering and the batch LDCRF method. It also shows a graph comparing the average F1 scores of the three approaches, with their means and standard deviations. In the case of the continuous LDCRF with filtering on the prediction layer, the optimal window size was 60, although the average F1 score was higher when $k = 80$. Surprisingly, when $k \geq 60$, the average F1 score was even higher than the batch LDCRF, although the difference was not statistically significant. In the case of the continuous LDCRF without filtering, under any condition it performed statistically significantly worse than the other two approaches ($p < 0.001$).

Lastly, we tested the multilayered filtering by setting the Gaussian window size $w = 5$, the temporal window size $k = 60$ (optimal values obtained from cross-validations in the previous experiments). Table VI shows the experimental results. The continuous LDCRF with multilayered filtering outperformed the continuous LDCRF without filtering by 15.78% (increasing the average F1 score from .7725 to .8944), and the batch

Table V. Experiment 4 Results on Continuous LDCRF, with or Without Using Filtering on the Prediction Layer, and the Batch LDCRF Method

Continuous LDCRF: Filtering on Prediction Layer

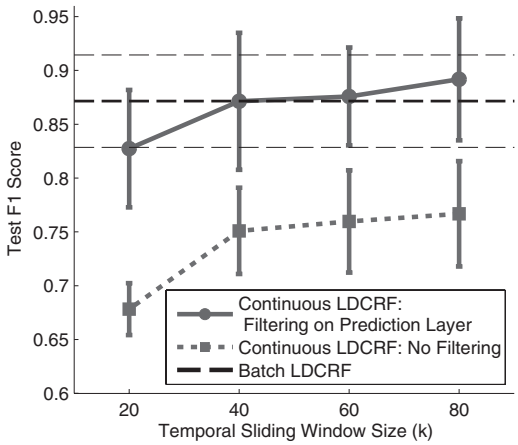
k	Delay	Train Set	Validate Set	Test Set	Avg(V+T)	Training Time	Params
20	1.0 s	.9005 (.03)	.8270 (.04)	.8274 (.05)	.8272 (.04)	360 m (25.76)	2,604
40	2.0 s	.9544 (.02)	.8713 (.03)	.8713 (.06)	.8713 (.05)	332 m (26.32)	2,604
60	**3.0 s**	**.9587 (.02)**	**.8907 (.04)**	**.8758 (.05)**	**.8833 (.04)**	**332 m (27.73)**	**2,604**
80	4.0 s	.9557 (.02)	.8842 (.03)	.8917 (.06)	.8880 (.04)	358 m (42.01)	2,604

Continuous LDCRF: No Filtering

k	Delay	Train Set	Validate Set	Test Set	Avg(V+T)	Training Time	Params
20	0.0 s	.7443 (.03)	.6899 (.05)	.6783 (.02)	.6841 (.04)	349 m (49.20)	2,604
40	0.0 s	.8160 (.02)	.7526 (.05)	.7509 (.04)	.7518 (.04)	336 m (33.09)	2,604
60	0.0 s	.8309 (.02)	.7622 (.06)	.7597 (.05)	.7610 (.05)	329 m (18.91)	2,604
80	**0.0 s**	**.8332 (.02)**	**.7783 (.06)**	**.7668 (.05)**	**.7725 (.05)**	**336 m (27.60)**	**2,604**

Batch LDCRF

k	Delay	Train Set	Validate Set	Test Set	Avg(V+T)	Training Time	Params
0	∞	.9514 (.03)	.8785 (.04)	.8645 (.05)	.8715 (.04)	305 m (35.73)	2,604



In all experiments, $|\mathcal{H}_y| = 7$ and $\sigma^2 = 1{,}000$. The graph shows average F1 scores of test results on the validate and the test split. Error bars show standard deviation.

Table VI. Experiment 4 Results on the Multilayered Filtering

Multi-layered Filtering vs. Baseline Approaches ($|\mathcal{H}_y| = 7$, $\sigma^2 = 1{,}000$)

Filter	w, k	Delay	Train Set	Validate Set	Test Set	Avg (V+T)	Training Time
I,P	**5, 60**	**3.0 s**	**.9650 (.02)**	**.8912 (.04)**	**.8977 (.05)**	**.8944 (.04)**	**442 m (30.28)**
P	0, 60	3.0 s	.9587 (.01)	.8907 (.04)	.8758 (.05)	.8833 (.04)	331 m (50.67)
X	0, 80	0.0 s	.8332 (.02)	.7783 (.06)	.7668 (.05)	.7725 (.05)	336 m (27.60)
X	0, 0	∞	.9514 (.03)	.8785 (.04)	.8645 (.05)	.8715 (.04)	305 m (35.73)

The "Filter" column indicates whether filtering was used: filtering on both the input layer and the prediction layer (I,P); filtering on the prediction layer only (P); no multilayered filtering (X); and the batch LDCRF (X). In all experiments, $|\mathcal{H}_y| = 7$, $\sigma^2 = 1{,}000$.

LDCRF method by 2.63% (increasing the average F1 score from .8715 to .8944). The difference between the two continuous LDCRF methods (IP versus X) was statistically sinificant ($t(18) = 5.63$, $p < 0.001$). Figure 10 shows an ROC plot (averaged over the 6 classes) and confusion matrices from the results of this experiment.

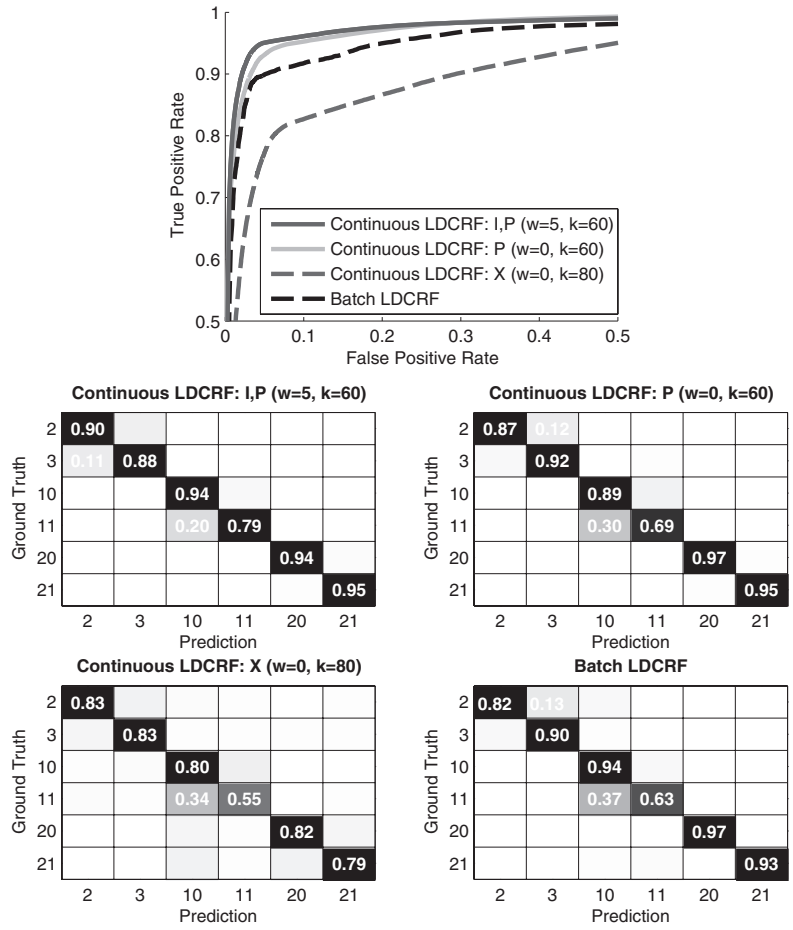


Fig. 10. An ROC plot and confusion matrices from experiment 4. The ROC plot was obtained by taking an average result of the six gesture classes. The continuous LDCRF with our multilayered filtering statistically significantly outperforms other approaches, improving both the false positive and true positive rates.

4.7. Experiment 5: Evaluation on All 24 NATOPS Gestures

The experiments described earlier suggest the optimal model parameter settings for continuous gesture recognition: the number of hidden states per class $|\mathcal{H}_y| = 7$, the regularization factor $\sigma^2 = 1,000$, the Gaussian window size $\omega = 5$, and the temporal window size $k = 60$. Using these, we trained the model on data from ten subjects, using all twenty-four gestures in the dataset (see Song et al. [2011b] for a complete list of gestures).

The experimental protocol was as follows. The test split contained data samples from the first five subjects, the validation split contained the next five subjects, and the training split contained the remaining ten subjects. Remember that in the NATOPS dataset there are 24 gestures, each of 20 subjects performed each gesture 20 times. Therefore, both the test and validation split included 2400 data samples, while the training split included 4800 data samples.

Figure 11 shows a confusion matrix of the result from the test split. The overall accuracy was 75.37% and the F1 score was 0.7349. The result from the validation split

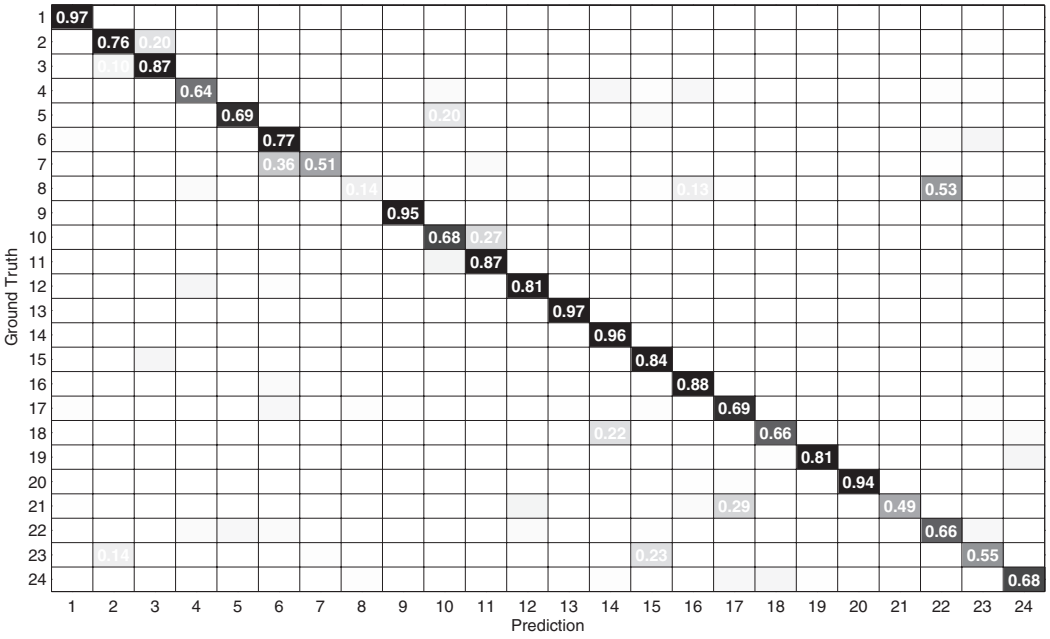


Fig. 11. Confusion matrix from experiment 5. An LDCRF with multilayered filter was trained on data from ten subjects, including all twenty-four gestures in the NATOPS dataset. From the previous experiments, we set the model parameter values as $|\mathcal{H}_y| = 7$, $\sigma^2 = 1,000$, $\omega = 5$, $k = 60$. Only the result from the test split is shown. The overall accuracy on the test split it was 75.37%, while on the validation split it was 86.35%.

showed better recognition performance, with the accuracy of 86.35% and the F1 score of 0.8559, which we suspect is due to variance in subjects' gestures.

5. CONCLUSION AND FUTURE WORK

We presented a unified framework for continuous gesture recognition that tracks upper body and hands, and combines the two sources of information to interpret body and hand gestures. Our system tracks upper body postures by fitting a kinematic upper-body model to observations under multihypothesis Bayesian filtering. We devised a feature function that combines both static and dynamic attributes of motion, defining an MHI-based error measure to capture the dynamic attributes of motion. We showed that our feature function significantly improves the accuracy of estimated signals, demonstrating it on a task of gesture recognition and showing that this approach yields a 6.3% performance improvement. The estimated body postures are used to guide searching for hands. We perform coarse-grained hand tracking, defining a vocabulary of four canonical hand shapes (i.e., thumb up/down, palm open/closed). The HOG image descriptors are extracted around the estimated wrist position, and an offline trained multiclass SVM classifies hand shapes. These body and hand signals are then combined to form the input feature for the gesture recognition.

We used an LDCRF as a building block of our gesture recognition system, and used a temporal sliding window to perform online sequence labeling and segmentation. We augmented this with our multilayered filtering: It performs filtering on the input layer using a Gaussian temporal-smoothing filter, allowing the input signal to embrace long-range temporal dependencies and have a low signal-to-noise ratio. Filtering on the prediction layer is done using a weighted-average filter and a moving-average filter, allowing us to take weighted votes of multiple overlapping prediction results as

well as reduce noise in the prediction results. We demonstrated this approach on a task of continuous gesture recognition using the NATOPS dataset, and showed that the multilayered filtering produces a 15.78% recognition performance improvement over the nonfiltering method. We also showed that our continuous gesture recognition system achieves a recognition accuracy of 75.37% on a set of twenty-four NATOPS gestures.

Our current system can be improved in a number of ways. We performed body posture estimation and hand shape classification serially, using estimated wrist positions to search for hands. However, once the hands are detected, they could be used to refine the body posture estimation (e.g., by inverse kinematics). Context-sensitive pose estimation may also improve performance. There is a kind of grammar to gestures in practice: for the NATOPS scenario as an example, once the "brakes on" gesture is performed, a number of other gestures are effectively ruled out (e.g., "move ahead"). Incorporating this sort of context information might significantly improve estimation performance.

Our multilayered filtering introduces some delay, for example, one to four seconds in experiment 4 (see Section 4.6). Whether or not this delay is permissible depends on the application. For the task of aircraft deck handling (for which the NATOPS signals were created) this seems acceptable. The unmanned aerial vehicles will be moving at around 2–3mph when being directed, and are heavy objects incapable of fast changes in speed or direction when on the ground. Hence a 2–3-second delay appears acceptable on this task.

As mentioned in Section 3.2, forward-only inference [Murphy 2002] eliminates the need for a temporal sliding window in continuous sequence labeling. For example, Huang et al. [2011] implemented real-time CRF using the forward-only inference so that it can make predictions in real time. We plan to implement real-time LDCRF using the forward-only inference and compare its performance to our multilayered filtering approach.

Lastly, in order for our system to be interactive, it is necessary to allow two-way communication between the users and the system. Although it is crucial for a system to be able to understand human gestures, it is also necessary for the system to have an appropriate feedback mechanism, that is, the system has to be able to gesture back, just as a human would do in the same situation. There are many questions to be answered: What does it mean for a system to gesture? How can we define a natural feedback mechanism? How natural can a system's feedback be? We look forward to exploring these questions in future work.

ACKNOWLEDGMENTS

The authors would like to thank the editors and anonymous reviewers for their comments and suggestions, which significantly helped improving the quality of this article. The authors also would like to thank Mary L. Cummings for helping us understanding the aircraft carrier environment, Louis-Philippe Morency for providing us a Matlab distributed computing cluster that was used to run all the experiments, and Chang-Yoon Park for assisting us recording video clips for the NATOPS dataset.

REFERENCES

Barr, A. 1981. Superquadrics and angle-preserving transformations. *IEEE Comput. Graph. Appl. 1,* 1, 11–23.

Bobick, A. F. and Davis, J. W. 2001. The recognition of human movement using temporal templates. *IEEE Trans. Pattern Anal. Mach. Intell. 23,* 3, 257–267.

Bradski, G. and Kaehler, A. 2008. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA.

Brand, M. 1999. Shadow puppetry. In *Proceedings of the IEEE International Conference on Computer Vision*. 1237–1244.

CHANG, C.-C. AND LIN, C.-J. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol. 2,* 3, 27.

DALAL, N. AND TRIGGS, B. 2005. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 886–893.

DENAVIT, J. AND HARTENBERG, R. S. 1955. A kinematic notation for lower-pair mechanisms based on matrices. *ASME J. Appl. Mechan. 23*, 215–221.

DEUTSCHER, J., BLAKE, A., AND REID, I. D. 2000. Articulated body motion capture by annealed particle filtering. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2126–2133.

ENGIN, A. 1980. On the biomechanics of the shoulder complex. *J. Biomechan. 13,* 7, 575–581, 583–590.

EROL, A., BEBIS, G., NICOLESCU, M., BOYLE, R. D., AND TWOMBLY, X. 2007. Vision-based hand pose estimation: A review. *Comput. Vis. Image Understand. 108,* 1-2, 52–73.

FENG, X., YANG, J., AND ABDEL-MALEK, K. 2008. Survey of biomechanical models for the human shoulder complex. Tech. rep., SAE International.

FOFI, D., SLIWA, T., AND VOISIN, Y. 2004. A comparative survey on invisible structured light. In *Proceedings of SPIE Machine Vision Applications in Industrial Inspection XII*.

FREEMAN, W. T., ANDERSON, D. B., BEARDSLEY, P. A., DODGE, C., ROTH, M., WEISSMAN, C. D., YERAZUNIS, W. S., KAGE, H., KYUMA, K., MIYAKE, Y., AND ICHI TANAKA, K. 1998. Computer vision for interactive computer graphics. *IEEE Comput. Graph. Appl. 18,* 3, 42–53.

GOKTURK, S., YALCIN, H., AND BAMJI, C. 2004. A time-of-flight depth sensor—System description, issues and solutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop.*

GUNAWARDANA, A., MAHAJAN, M., ACERO, A., AND PLATT, J. C. 2005. Hidden conditional random fields for phone classification. In *Proceedings of the 9th European Conference on Speech Communication and Technology*. 1117–1120.

HARRIS, F. 1978. On the use of windows for harmonic analysis with the discrete fourier transform. *Proc. IEEE 66,* 1, 51–83.

HSU, C.-W. AND LIN, C.-J. 2002. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Netw. 13,* 2, 415–425.

HUANG, L., MORENCY, L.-P., AND GRATCH, J. 2011. Virtual rapport 2.0. In *Proceedings of the 11th International Conference on Intelligent Virtual Agents*. Lecture Notes in Computer Science Series, vol. 6895, Springer, 68–79.

ISARD, M. AND BLAKE, A. 1998. CONDENSATION—Conditional density propagation for visual tracking. *Int. J. Comput. Vis. 29,* 1, 5–28.

KIM, K., CHALIDABHONGSE, T. H., HARWOOD, D., AND DAVIS, L. S. 2005. Real-Time foreground-background segmentation using codebook model. *Real-Time Imag. 11,* 3, 172–185.

KNERR, S., PERSONNAZ, L., AND DREYFUS, G. 1990. Single-Layer learning revisited: A stepwise procedure for building and training a neural network. In *Neurocomputing: Algorithms, Architectures and Applications*, J. Fogelman, Ed., Springer-Verlag.

LAFFERTY, J. D., MCCALLUM, A., AND PEREIRA, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann, 282–289.

LEE, M. W. AND COHEN, I. 2006. A model-based approach for estimating human 3d poses in static images. *IEEE Trans. Pattern Anal. Mach. Intell. 28*, 905–916.

MITRA, S. AND ACHARYA, T. 2007. Gesture recognition: A survey. *IEEE Trans. Syst. Man, Cybernet. C: Appl. Rev. 37,* 3, 311–324.

MORENCY, L.-P., QUATTONI, A., AND DARRELL, T. 2007. Latent-Dynamic discriminative models for continuous gesture recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

MORI, G. AND MALIK, J. 2006. Recovering 3d human body configurations using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell. 28,* 7, 1052–1062.

MURPHY, K. 2002. Dynamic bayesian networks: Representation, inference and learning. Ph.D. thesis Computer Science Division, UC, Berkeley.

NASA. 1995. *Man-Systems Integration Standards:* Vol. 1. Section 3. Anthropometry and Biomechanics. http://msis.jsc.hasa.gov/sections/section03.htm.

NOCEDAL, J. AND WRIGHT, S. J. 1999. *Numerical Optimization*. Springer-Verlag.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Poppe, R. 2007. Vision-Based human motion analysis: An overview. *Comput. Vis. Image Understand. 108,* 1-2, 4–18.

Quattoni, A., Wang, S. B., Morency, L.-P., Collins, M., and Darrell, T. 2007. Hidden conditional random fields. *IEEE Trans. Pattern Anal. Mach. Intell. 29,* 10, 1848–1852.

Schwarz, L. A., Mkhitaryan, A., Mateus, D., and Navab, N. 2011. Estimating human 3d pose from time-of-flight images based on geodesic distances and optical flow. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*. 700–706.

Shakhnarovich, G., Viola, P. A., and Darrell, T. 2003. Fast pose estimation with parameter-sensitive hashing. In *Proceedings of the IEEE International Conference on Computer Vision*. 750–759.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. 2011. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Sminchisescu, C. and Triggs, B. 2003. Kinematic jump processes for monocular 3d human tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 69–76.

Song, Y., Demirdjian, D., and Davis, R. 2011a. Multi-Signal gesture recognition using temporal smoothing hidden conditional random fields. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*. 388–393.

Song, Y., Demirdjian, D., and Davis, R. 2011b. Tracking body and hands for gesture recognition: Natops aircraft handling signals database. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*. 500–506.

Sutton, C. A., Rohanimanesh, K., and McCallum, A. 2004. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann.

Vapnik, V. N. 1995. *The Nature of Statistical Learning Theory*. Springer, New York.

Wang, Y. and Mori, G. 2009. Max-Margin hidden conditional random fields for human action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 872–879.

Weinland, D., Ronfard, R., and Boyer, E. 2011. A survey of vision-based methods for action representation, segmentation and recognition. *Comput. Vis. Image Understand. 115,* 2, 224–241.

Yin, Y. and Davis, R. 2010. Toward natural interaction in the real world: Real-Time gesture recognition. In *Proceedings of the 12th International Conference on Multimodal Interfaces/International Workshop on Machine Learning for Multimodal Interaction*. 15.

Received December 2010; revised December 2011; accepted December 2011