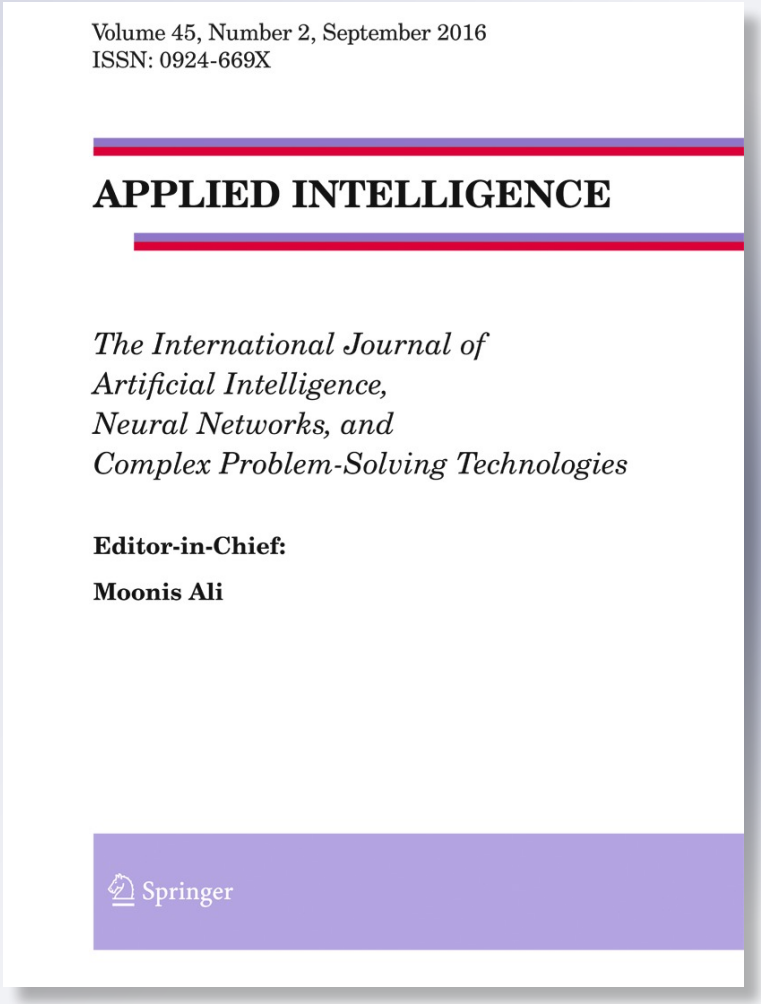
Selective temporal filtering and its application to hand gesture recognition

Myung-Cheol Roh, Siamac Fazli & Seong-Whan Lee

Applied Intelligence
The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies

ISSN 0924-669X
Volume 45
Number 2

Appl Intell (2016) 45:255-264
DOI 10.1007/s10489-015-0757-8



Springer

Appl Intell (2016) 45:255–264
DOI 10.1007/s10489-015-0757-8

Selective temporal filtering and its application to hand gesture recognition

Myung-Cheol Roh[1] · **Siamac Fazli**[1] · **Seong-Whan Lee**[1]

Published online: 22 February 2016

Abstract In temporal data analysis, noisy data is inevitable in both testing and training. This noise can seriously influence the performance of the temporal data analysis. To address this problem, we propose a novel method, termed Selective Temporal Filtering that builds a noise-free model for classification during training and identifies key-feature vectors that are noise-filtered data from the input sequence during testing. The use of these key-feature vectors makes the classifier robust to noise within the input space. The proposed method is validated on a synthetic-dataset and a database of American Sign Language. Using key-feature vectors results in robust performance with respect to the noise content. Futhermore, we are able to show that the proposed method not only outperforms Conditional Random Fields and Hidden Markov Models in noisy environments, but also in a well-controlled environment where we assume no significant noise vectors exist.

Keywords Key-feature vector · Selective Temporal Filtering · Gesture recognition

Myung-cheol Roh is currently with S-1 Corporation.

✉ Seong-Whan Lee
sw.lee@korea.ac.kr

Myung-Cheol Roh
myung.roh@gmail.com

Siamac Fazli
fazli@korea.ac.kr

[1] Department of Brain and Cognitive Engineering, Korea University, Anam-dong, Seongbuk-ku, Seoul 02841, Korea

1 Introduction

Temporal data analysis has played an important role in numerous research fields and applications including human-robot interface, gesture recognition, and smart-phone interfaces. Many useful tools have been proposed to manage temporal data. One of the most well-known and successful methods are HMMs (Hidden Markov Models) [1]. HMMs provide an efficient method to model temporal data and has been implemented in various applications such as speech, gesture, and face analysis [2, 3]. Recently, DBNs (Dynamic Bayesian Networks) have been highlighted to address spatial and temporal data effectively [4, 5]. DBNs are directed acyclic graphs and are convenient tools for modeling casual relationships between features. HMMs are a special case of DBNs [4]. CRFs (Conditional Random Fields) and extended CRFs are also widely used methods [6, 7]. CRF is a discriminative undirected probabilistic graphical model. It has been widely used in language processing and gesture recognition.

The above approaches are mainly concerned with the dependency of data within the temporal domain. The majority of temporal data classification methods have focused on data modeling in environments where the noise proportion is assumed to be small and it is generally assumed, implicitly or explicitly, that the noise content can be absorbed by the models. However, noise appears in unexpected forms; thus, it cannot be modeled in general. For example in gesture recognition, impulse noise can be added in a sequence of feature vectors owing to tracking failure; this occurs frequently in real environments. Noise seriously influences the accuracy of temporal data analysis in practical environments. Moreover, noisy measurements occur not only in testing environments but also in training environments, which makes noise-free training difficult.

Noise filtering has been extensively studied in *static* data analysis and there are many previous approaches regarding filtering noise and unnecessary features. The most popular method is PCA (Principal Component Analysis) [8]. PCA finds the principal axes such that the variance of the data projected onto the axes is maximized under a Gaussian distribution assumption. Using those principal axes, the dimensionality of a dataset can be reduced and/or noise sources can be suppressed. One of the conventional, yet powerful frameworks for filtering noisy data is the RANSAC (RANdom SAmple Consensus) method [9]. It provides a framework of filtering outliers that are noisy or removing non-helpful features in static data analysis. There are many methods based on RANSAC, such as stereo image matching, image stitching, and face shape alignment [10–12]. Adaboost is widely used to select useful features for classification [13]. Adaboost is a method of building a set of weak classifiers and is used to select informative features from a large feature set. These feature-filtering approaches in static data are useful in filtering out noise and non-informative data and can facilitate reliable analysis even in noisy environments. However, these tools cannot be utilized for temporal data.

The determination of whether a given data point is noisy can only be made when a corresponding correct model is provided and the model is noise-free. Thus, we define noise as outliers that do not fit well within a given noise-free model. To estimate a noise-free model and select noise-free data, we propose a novel method, termed STF (Selective Temporal Filtering), a temporal data filtering and classification method. In training, it determines a noise-free model and in testing, it identifies key-feature vectors that are noise-free data for a given noise-free model and classifies using these key-feature vectors. To determine a noise-free model, noise-free data must be provided during the training phase. However, without a noise-free model, noise-free data cannot be identified. To solve this chained problem, RANSAC is used. The basic processes are hypothesizing candidate models, evaluating the models, and selecting the best model by measuring its agreement with the observations. The best hypothesis is the noise-free model. Given a noise-free model, key-feature vectors are identified based on a dynamic programming approach. According to our experiments on a synthetic-dataset and an American Sign Language database, the proposed approach demonstrates stable performance against various noise levels.

We would like to mention several related approaches of hand gesture detection: K. Hu and L. Yin proposed a scale-invariant feature descriptor based on multi-layer geometric shapes to classify sixteen types of hand postures [14]. V.-T. Nguyen et al. proposed a hand representation method that is invariant to scale, rotation, and differences in the object structures [15]. The above methods are robust to background complexity and object variances. However, they are directed at analyzing postures rather than gestures.

H.-D. Yang et al. proposed a threshold CRF to recognize American Sign Language [6]. The threshold CRF distinguishes between in-vocabulary signs and out-of-vocabulary non-signs. In [16], H.-D. Yang and S.-W. Lee proposed a combining method of manual and non-manual features for American sign language recognition. Manual signs were detected using CRF and non-manual signals and facial expressions were recognized using SVM.

Recently, gesture analysis methods based on 3-dimensional (3D) information have been discussed. H.-D. Yang proposed a hierarchical threshold CRF to recognize sign language gestures [17]. The method used 3D data, which was obtained by a TOF (time of flight) camera and used a hierarchical CRF to recognize American sign language. F. Ghaleb et al. proposed a hand gesture recognition method based on CRF in conjunction with SVMs [18]. They used a depth map to detect and segment the hands. Using shape features, which they proposed, an SVM verified whether a gesture was meaningful or not.

We use CRF and HMM as the baseline algorithms in this paper. HMMs have been one of the most successful models in sequential data analysis and CRF is the one of the most popular models in hand gesture recognition, as indicated above. The proposed method as well as the baseline algorithms are tested on the American Sign Language data and the synthetic-dataset. The comparative results are presented in the Experiments and Analysis section.

2 Problem definition

Let $\mathbf{Y} = \{Y_1, Y_2, \cdots Y_T\}$ be an observed input sequence and Y_i be the D-dimensional feature vector, where $1 \leq i \leq T$ and T is the length of $\mathbf{Y}$ over the time axis. In temporal data, a classification problem is defined to determine c such that

$$\arg\max_c \mathcal{E}(\mathbf{Y}, \Theta_c) \tag{1}$$

where Θ_c is the parameter set of the cth class model. $\mathcal{E}$ is an evaluation function of the input sequence $\mathbf{Y}$ for a given Θ_c. $\mathcal{E}$ measures the probability that an observation $\mathbf{Y}$ occurs in the cth class model. For example, in HMM, $\mathcal{E}$ is defined as follows:

$$\mathcal{E}(\mathbf{Y}, \Theta_c) = P(\mathbf{Y}|\Theta_c), \tag{2}$$

and $\mathcal{E}$ is calculated by:

$$\mathcal{E}_{HMM}(\mathbf{Y}, \Theta_c) = P(\mathbf{Y}|\Theta_c) = \sum_{all\ Q_\mathbf{Y}} P(\mathbf{Y}|Q_\mathbf{Y}, \Theta_c) P(Q_\mathbf{Y}|\Theta_c) \tag{3}$$

where Q is the state sequence for the observations $\mathbf{Y}$. This approach uses every input feature vector in an observed

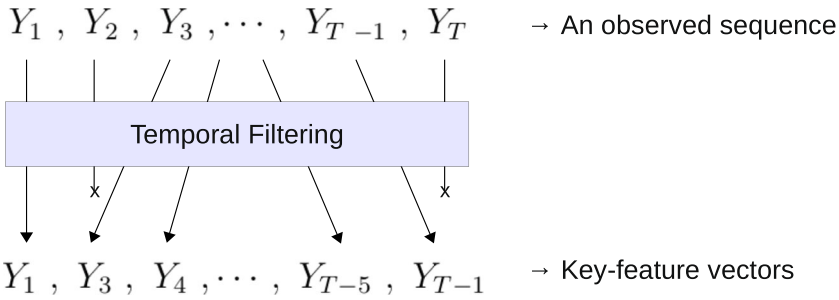


Fig. 1 Temporal filtering identifies useful feature vectors according to a given model. Feature vectors found through temporal filtering are key-feature vectors

sequence, which may include noise and non-informative feature vectors that do not provide any useful information for classification.

Noise and non-informative information are typically inherent in the data we obtain. Even in training data that are collected within a controlled environment, unnecessary feature vectors and noise are inevitable. Previous approaches, including HMM, all feature vectors of an observed sequence are used for classifier estimation. This makes the classifier particularly sensitive to noise patterns, which were not presented during training data collection.

If we can identify and use only informative observations $\mathcal{K}$, the classification accuracy can be more stable and robust to noise. Figure 1 presents the concept of filtering. The selected feature vectors are a subset of a sequence of observed feature vectors and are called *key-feature vectors* in this paper. For example, in HMM, given a subset of noise-free feature vectors $\mathcal{K}_c$ for a class c, the input observation can be evaluated as follows:

$$\mathcal{E}_{HMM_c} = \alpha_c \sum_{all\ Q_{\mathcal{K}_c}} P(\mathcal{K}_c|Q_{\mathcal{K}_c}, \Theta_c) P(Q_{\mathcal{K}_c}|\Theta_c) \tag{4}$$

where $\mathcal{K}$ is a key-feature vector that is a subset of $\mathbf{Y}$ and α_c is a normalization constant. For a given model and an observation sequence, key-feature vector sequences are defined as the minimal subset of the observation sequence that is required to classify the observation sequence correctly. In this paper, we propose STF (Selective Temporal Filtering) to identify $\mathcal{K}$ and classify using $\mathcal{K}$ instead of $\mathbf{Y}$, simultaneously.

Note that the conventional feature-filtering approach processes temporal data over a spatial domain and STF processes the data over a *temporal domain*.

3 Selective temporal filtering

In the proposed STF, the classification definition differs from the conventional definition. $\mathcal{E}(\mathbf{Y}, \Theta_c)$ in (1) is redefined as follows:

$$\mathcal{E}(\mathbf{Y}, \Theta_c) = \max_{Y \subset \mathbf{Y}} \mathcal{F}(Y, \Theta_c) \mathcal{L}(Y, \Theta_c) \tag{5}$$

where Y and Θ_c are a subset of an observation $\mathbf{Y}$ and the cth class model parameter. $\mathcal{F}$ is an evaluation function that measures the agreement of Y with the cth class model. $\mathcal{L}$ is a function that measures the sampling distribution of Y to allow it to address the problems that may be caused by using only a subset of the original observation set (e.g., sub-gesture problems [19]). The complexity to solve (5) is extremely high. Thus, in this paper, we use a simplified version as follows:

$$\mathcal{E}(\mathbf{Y}, \Theta_c) = \left(\max_{Y' \subset \mathbf{Y}} \mathcal{F}(Y', \Theta_c) \right) \mathcal{L}(\mathcal{K}', \Theta_c) \tag{6}$$

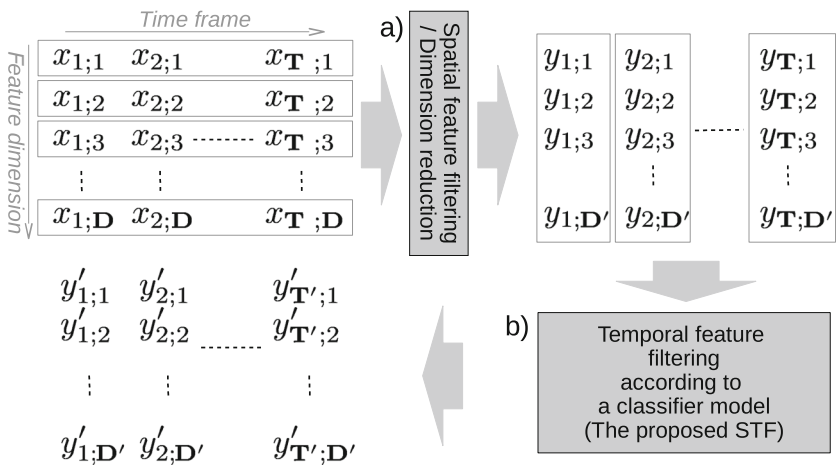


Fig. 2 Illustration demonstrating the key difference between conventional spatial feature filtering and the proposed STF (Selective Temporal Filtering). In general, conventional (spatial) filtering reduces the feature dimension from $\mathbf{D}$ to $\mathbf{D}'$ where $\mathbf{D}' \leq \mathbf{D}$. The proposed STF method identifies a key-feature vector sequence. Consequently, the number of time-frames is reduced from $\mathbf{T}$ to $\mathbf{T}'$ where $\mathbf{T}' \leq \mathbf{T}$

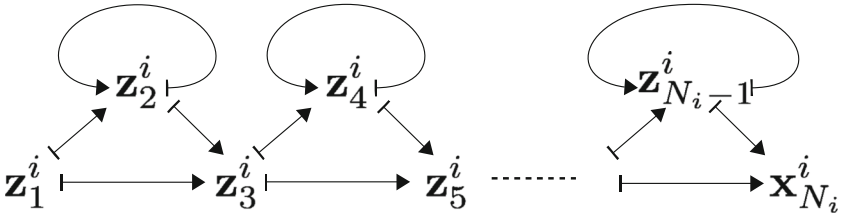


Fig. 3 Example of Finite-State Machine (FSM) used to describe the proposed ith classifier model

where $\mathcal{K}_c$ is a key-feature vector set. For the cth model and an observed feature sequence $\mathbf{Y} = \{Y_1, Y_2, \cdots, Y_{\mathbf{T}}\}$, the key-feature vector set is $\mathcal{K}_c \subset \mathbf{Y}$ which maximizes its agreement with the model as follows:

$$\mathcal{K}_c = \arg\max_{Y' \subset \mathbf{Y}} \mathcal{F}(Y', \Theta_c) \tag{7}$$

where $Y' = \{Y'_1, Y'_2, \cdots, Y'_{\mathbf{T}'}\}$ and for $Y_p = Y'_{p'}$ and $Y_q = Y'_{q'}$, if $p < q$ then $p' < q'$ (the orders are preserved). The lengths of the key-feature vector sequences $\mathbf{T}'$ in the model represents the number of minimum and necessary feature vectors.

Figure 2 illustrates the concept of the proposed STF. x, y, and y' are elements of feature vectors X, Y, and Y', respectively, where X, Y, and Y' are the observed (input) feature vectors, spatial filtered feature vectors, and temporal filtered feature vectors (key-feature vectors). There are two kinds of filtering in Fig. 2: spatial filtering and temporal filtering (STF). The spatial filtering filters a multi-dimensional feature vector $\{x_{t;1}, x_{t;2}, \cdots, x_{t;\mathbf{D}}\}$ on the spatial axis to reduce the dimension of a given feature vector or to combine the weight on each dimension [8, 13]. After spatial filtering, a new vector $Y_t = \{y_{t;1}, y_{t;2}, \cdots, y_{t;\mathbf{D}'}\}$ is obtained where $1 \le t \le \mathbf{T}$ and $\mathbf{D}' \le \mathbf{D}$. Spatial filtering is a widely used method (e.g., PCA). In STF, the spatial filtering has the role of reducing the dimensionality of the features space however, this is not mandatory.

Whereas a spatial filter manages spatial information at a given time t, a temporal filter addresses the temporal information. Given a sequence of feature vectors, $\mathbf{Y} = \{Y_1, Y_2, \cdots, Y_{\mathbf{T}}\}$, temporal filtering identifies a subset of $\mathbf{Y}$ according to a given classifier model. As a result of temporal feature filtering, a new set of vectors $\mathcal{K} = \{Y_1, Y_2, \cdots, Y_{\mathbf{T}'}\}$ is obtained from $\mathbf{Y}$, where in general $\mathbf{T}' \le \mathbf{T}$. $\mathbf{T}'$, the length of the key-feature vector sequence for a model, is determined according to the class model. For example, the $\mathbf{T}'$ of a complex model is greater than that of a simple model.

1: **for** $\forall i \in \{j|$ for jth feature vector $\mathbf{Y}_j, \mathbf{Y}_j \in \mathcal{K}\}$ **do**
2: $\quad s = \frac{N_s}{T'}$
3: $\quad B_c[$ Round $(s \times i)] \leftarrow 1$
4: **end for**
5: $\mathcal{L} = \sum_{l=1}^{N_s} B_c[l]$

Fig. 4 Algorithm for calculating the sampling distribution in (6)

Notice that spatial filtering reduces the dimension of a feature vector and temporal filtering reduces the length of the sequence in general.

Selecting the subset that consists of only key-feature vectors with respect to a classifier model can improve the invariance of a given classifier against noise. Further, because it always finds the same key-feature vector sequence length for a given model, the problem of the score or probability of a longer sequence being less than that of a shorter sequence is prevented. In the next section, we describe the methodology of modelling and training the STF.

3.1 Classifier model for STF

In the STF framework, the classification model has the important role of not only classifying the input sequence, but also selecting the key-feature vectors.

Figure 3 illustrates the STF model for the ith class having $\mathbf{T}'_c = (N_i - 1)/2$ key-feature vectors using a Finite-State Machine (FSM). The transition probabilities are non-zero and equal within the model, where e and o are even and odd numbers, respectively.

$$p(o+1|o) = p(o+2|o) = p(e|e) = p(e+1|e) \neq 0 \tag{8}$$

In STF, unlike in the majority of state-space models, the transition does not have an important role [1, 4]. The transition merely builds the structure. The even states, z_e^c, where e is an even number, are *null* states. The other states are *key-feature* states. In the *null* states, the observation probabilities are zero. For an input feature vector $\mathbf{y}$ and the null state z_e^c,

$$p(\mathbf{y}|z_e^c) \approx 0. \tag{9}$$

In this paper, the observation probability for each key-feature state is modeled by multiple Gaussian distributions. The observation probability $p(\mathbf{y}|\mathbf{z}_j^c)$ in the key-feature state is defined as follows:

$$p(\mathbf{y}|\mathbf{z}_j^c) = \max(\mathcal{N}(\mathbf{y}|\mu_{j,1}^c, \Sigma_{j,1}^c), \cdots, \mathcal{N}(\mathbf{y}|\mu_{j,G}^c, \Sigma_{j,G}^c)) \tag{10}$$

where G is the number of clusters of the jth node in the cth model. An observation is distributed by one of the multiple Gaussian models rather than by the weighted sum of multiple Gaussian models.

Using this model, only $\mathbf{T}'_c$ vectors remain in the key-feature states and they are considered the key-feature vector

$\mathbf{T}' \leftarrow (\lambda \times T'_s)$
$itr \leftarrow 0$
for $\forall\psi \in \Psi$ **do**
  for $\forall\varphi \subset \psi$ **do** ▷ Initialize a hypothesis
    $l \leftarrow 0$
    $\mu_{t,c} \leftarrow \mathcal{N}(\varphi(t), \sigma)$ for $1 \le t \le \mathbf{T}'$ and $1 \le c \le G$
    for rept $= 0$ to R **do**
      for $\forall\psi' \in \mathcal{D}$ **do** ▷ Update the parameters
        $l \leftarrow l + 1$
        $[\mathcal{K}_l, \epsilon_l] \leftarrow \mathcal{E}(\psi', \Theta_{itr})$
        $\Theta_{itr} \leftarrow k$-means with $\{\mathcal{K}_1, \cdots, \mathcal{K}_l\}$
      end for

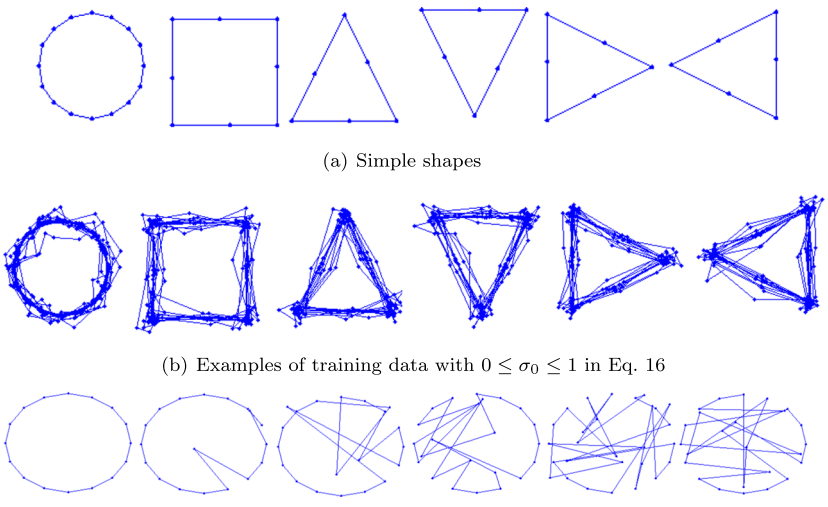
      $l \leftarrow 0$
    end for
    for $\forall\psi' \in \mathcal{D}$ **do** ▷ Evaluate the hypothesis
      $l \leftarrow l + 1$
      $[\mathcal{K}_l, \epsilon_l] \leftarrow \mathcal{E}(\psi', \Theta_{itr})$
    end for
    $\overline{\epsilon_{itr}} \leftarrow$ average $(\epsilon_1, \cdots, \epsilon_l)$
    $itr \leftarrow itr + 1$
  end for
end for
$\Theta^* = \text{argmax}_{\Theta_{itr}} \overline{\epsilon_{itr}}$ ▷ Select a best hypothesis

Fig. 5 Algorithm for the learning parameter $\Theta = \{T', \theta\}$ where $\theta = \{(\mu_{1,1}, \Sigma_{1,1}), \cdots (\mu_{\mathbf{T}',G}, \Sigma_{\mathbf{T}',G})\}$. λ controls the length of the key-feature vector as in (11). $\mathcal{D}$ is a training dataset. N_s and G are the shortest lengths (time frames) of the sample data and the number of clusters for each state, respectively. Ψ and φ are sets of training data having the length of T'_s among the training data and a subset of ψ, having the length of $\mathbf{T}' = 0.8T_s$ elements, respectively. $\mathcal{E}$ is the evaluation function that returns the key-feature vectors $\mathcal{K}$ and the error for the given sequence ψ' in (12) and (13)

$\mathcal{K}_c$ for the cth classifier model. The non-key-feature vectors and noisy feature vectors remain in the null states.

To implement STF, a dynamic programming approach is implemented as it is the most simple and intuitive method. In a dynamic programming table, each row corresponds to each state of the FSM: odd rows and even rows represent null and key-feature states, respectively. Notice that each key-feature state is modeled by a multi-modal distribution. The score $(\max_{Y' \subset \mathbf{Y}} \mathcal{F}(Y', \Theta_c))$ in (6) of a given input vector $\mathbf{Y}$ is calculated in the same conventional manner as the dynamic programming. The key-feature vector set $\mathcal{K}_i$ is determined by the back-tracking method of dynamic programming. Feature vectors that correspond to key-feature nodes are key-feature vectors.

In some cases, data can be represented by a sub-set of other data; this is known as a sub-gesture problem. For example, a trajectory of drawing the character "o" is a subset of number "8". Because STF uses a subset of the observation sequence, rather than all, there is a possibility that "8" could be classified as "o". We assume, that if the input data are classified correctly, the key-feature vectors will be selected evenly in the input data. Thus, for a given input sequence of the trajectory of "o", the key-feature vectors would be selected more evenly in the "o" class than in the



(a) Simple shapes

(b) Examples of training data with $0 \le \sigma_0 \le 1$ in Eq. 16

(c) Examples of testing data of circles with $0 \le \sigma_0 \le 2$ in Eq. 16 and 0 to 12 noise feature vectors inserted.

Fig. 6 Illustration of the simple shape data used

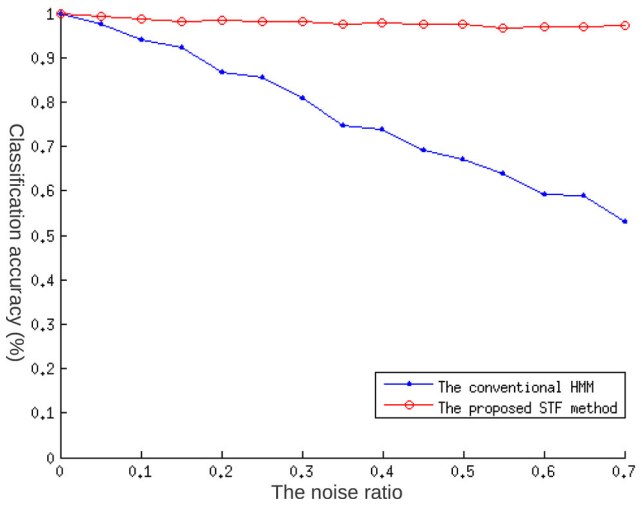


Fig. 7 Experimental results on simple shapes

"8" class. The $\mathcal{L}$ in (6) measures the sampling distribution. It weights the scores according to the distributions. Figure 4 presents the calculation of the sampling distribution we implemented.

3.2 Training

In this section, we will omit c to simplify the notation (e.g., $\mathbf{T}'$ means $\mathbf{T}'_c$). All classifiers can be trained separately.

In STF, during training the following parameters are estimated: the length of the key-feature vector sequence $\mathbf{T}'$ and the distribution model of the observations for each node z_t where $1 \leq t \leq \mathbf{T}'$ in Fig. 3.

Determining the optimal $\mathbf{T}'$ for a class is difficult without any prior knowledge of the minimum length of a sequence. In this paper, we assume that the sequences having the shortest length among the training samples are close to an ideal model of the key-feature vectors. $\mathbf{T}'$ is set as follows:

$$\mathbf{T}' = \text{ceil}(\lambda T'_s) \tag{11}$$

where $0 < \lambda \leq 1$ and T'_s is the shortest sequence length of the sequences in the training data.

To estimate the distribution of the observations, we use the k-means algorithm. The distribution of each cluster is represented by a Gaussian model. Estimating the distribution of the observations for z_t is an NP problem in STF. However, from the assumption that the shortest samples are close to the ideal model, we initialize the distribution from one of the feature vectors of the shortest samples.

Figure 5 shows the algorithm, which was used to learn the parameters of the classifier model. The algorithm consists of four parts: initializing a hypothesis of parameters, updating the initialized parameters, evaluating the hypothesis, and selecting the best hypothesis.

While "initializing" a hypothesis process, the initial parameters of G Gaussians on each key-feature node are given from the random subset $\varphi \in \psi$ where ψ is one of the training sequences having the shortest length T_s. φ is a sequence with a length of λT_s.

In the updating process, a key-feature vector sequence is identified for each training data according to a given correct model and the G Gaussian models are updated according to the corresponding key-feature vectors to key-feature nodes. The key-feature vectors are determined by an evaluation function. For a given sequence ψ', the evaluation function $\mathcal{E}$ returns the key-feature vectors $\mathcal{K}$ and the error ϵ as follows:

$$\epsilon = \mathcal{E}_\epsilon(\psi', \Theta) = \text{median}(e_1, \cdots, e_N) \tag{12}$$

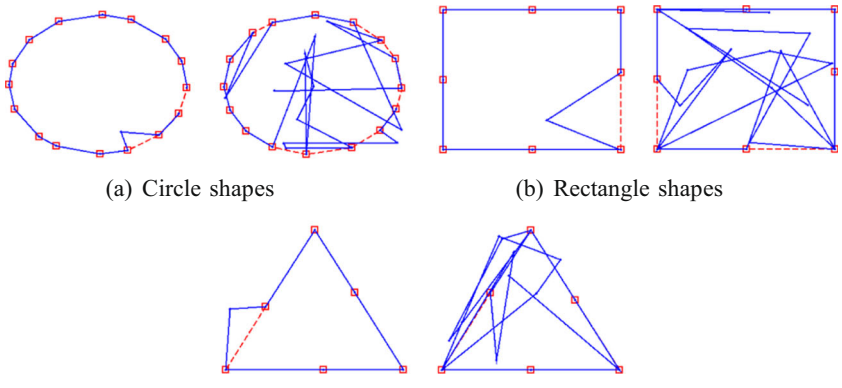
$$\mathcal{K} = \mathcal{E}_\mathcal{K}(\psi', \Theta) = \arg\max_{\varphi^q} \sum_{t=1}^{N} p(\varphi_t^q|\theta_t) \tag{13}$$

where

$$p(\varphi_t^q|\theta_t) = \max(\mathcal{N}(\varphi_t^q|\theta_{t,1}), \cdots, \mathcal{N}(\varphi_t^q|\theta_{t,G})), \tag{14}$$

$\theta_t = \{\theta_{t,1}, \cdots, \theta_{t,c}\}$ and G are the numbers of clusters for each state. The evaluation function is implemented using a dynamic programming technique.

In the "evaluating the hypothesis" process, all the training data are evaluated with respect to the updated parameter



(a) Circle shapes (b) Rectangle shapes

(c) Triangle shapes

Fig. 8 Selected key-feature points in our experiments for the low noise data (noise ratio to noise-free data, $nr = 0.05$) and extremely high noise data($nr = 0.7$). Input data are represented by blue curves and selected key-feature points using the STF method are represented red curves with rectangles

Fig. 9 Some frames of the ASL database. The colored curves represent hands trajectories tracked by an appearance-based tracking method

and the total error is calculated. The above processes are repeated until all the hypotheses are tested. Finally, in the "selecting a hypothesis" process, the hypothesis with the lowest error of all the training data is selected.

4 Experiments and analysis

To evaluate the proposed method, we conducted experiments on two different datasets. One was a dataset of synthetic simple shapes. Using the simple shape data, we demonstrated how the STF method functioned and investigated its robustness properties to noise. The second was an ASL (American Sign Language) database, which is the one of the most representative hand gesture databases. The ASL database was created in a controlled room where noise was assumed non-existent or significantly low. In our experiments, even though the model used in the proposed STF is simple, it outperformed the HMMs and CRF with and without noise.

4.1 STF on synthetic data

In the simple shape dataset experiment, we used six primitive shapes: square, circle, and four different triangles where the circumscribed rectangles were squares having a length of two. Figure 6a shows the six primitive shapes.

Let a primitive feature vector at time t be $\mathbf{v}_t = \{v_1, v_2, \cdots, v_D\}$. Data (noisy data) are generated by adding Gaussian noise to the primitive feature vector as follows:

$$\mathbf{y}_t = \{y_1, y_2, \cdots, y_D\} \tag{15}$$

where

$$y_i = v_u + \mathcal{N}(\mu, \sigma). \tag{16}$$

and $0 \leq \sigma \leq \sigma_0$. For training data, σ_0 was set to 1. Figure 6b illustrates examples of training data. Ten examples for each shape were drawn. For testing data, σ_0 was set to 2 to assume considerably greater variations than in training data. In addition to the higher variability within the testing data, also insertion noise was added. $nr \times D$ random vectors were inserted at random positions in the sequences, where $0 \leq nr \leq 0.7$. nr represents a noise ratio of the number of noise feature vectors to the number of noise-free feature vectors in a sequence. The testing data included considerably greater variance in the feature vectors and contained up to 70% of erroneous (noise) feature vectors. Figure 6c presents examples of the testing data. For each shape, 550 feature vectors were used for training and 8250($= 550 \times 15$ levels of the nr) feature vectors were used for testing. We used the positions of the points as feature vectors.

Figure 7 presents the experimental results on the simple shapes, where the red line with circles and the blue line with small dots represent the proposed STF method and conventional HMM, respectively. This experiment demonstrates the robustness to noise feature vectors of the proposed STF. As shown in Fig. 7, STF outperformed HMM. When the nr was 0, the accuracies of both methods were virtually perfect, 99.9 % and 99.7 %, for the testing data that were generated with a large variance, σ_0.

The accuracy of HMM declined significantly as noise feature vectors are added. However, STF could filter out non-key-feature vectors and select only key-feature vectors. Thus, the noise ratio did not significantly influence the classification accuracy in the proposed STF. Figure 8 presents the selected key-feature vectors that were identified by STF. The input data are represented by blue lines and the key-feature vectors are represented by red broken-lines with red rectangles. As can be seen, even under serious noise, STF determined the key-features successfully.

4.2 STF on the american sign language data

The second experiment was conducted on the ASL database, which was collected at Boston University.[1] Figure 9 exhibits some sample videos with hand trajectories. The video data we used included 48 sign words: And, Born, Arrive, Boy, Bicycle, Butter, Big, Day, Black, Car, Decide, Cold, Different, Here, Farm, Interpret, Inform, Funny, Finish, Library, Good, Magazine, Hot, Know, Many, Maybe, Lie, Name, Like, Night, Man, Rain, Now, Read, Out, Shoes, Past, Sorry, Sit, Take-off, Strange, What, Want, Work, Tell, Yesterday, Together, and Wow.

Ten video sequences per word were used for training. While capturing training data, the subjects were asked to wear colored gloves: a green glove on the left hand and a purple glove on the right hand. Another ten video sequences per sign were collected with bare hands for testing.

[1] American Sign Language Database, http://www.bu.edu/asllrp/ncslgr.html

The features used in this experiment were the same as those in [6]. A gesture was described by a sequence of six features that were the relative positions of each hand with respect to the face, vertical symmetry of the two hands, occlusion state of the two hands, and motion directions of the two hands between two consecutive frames. The relative positions of the left and right hands were modeled by mixtures of 35 and 33 Gaussians, respectively. The motion directions were coded by 8 directional codewords [20] plus "no movement" codeword. The occlusion state and the symmetry of the two hands had binary values, on and off. The face was detected by [21] and the hands were tracked by an appearance-based tracking method [22].

Let T_s be the shortest length of a sequence for a class. To train a model, we set the length of the key-feature vectors for each class as $0.8T_s$ ($\lambda = 0.8$ in (11)).

Figure 10 presents the experimental results for the ASL database. We compared STF with HMM and CRF. The CRFs were the same as used in [6] and the data we used in this experiment were isolated data. When the noise ratio was 0 ($nr = 0$), the accuracies of STF, HMM, and CRF were 92.1 %, 75.4 %, and 85.7 %, respectively. Even for *noise-free* data ($nr = 0$), the STF method outperformed HMM and CRF. This demonstrates that even on relatively well-captured data, unnecessary and non-informative feature vectors exist that may seriously influence the performance. As the nr was increased, the performance declined to 62.2 %, 48.8 %, and 48.4 % for the STF, HMM, and CRF methods, respectively.

In addition to the absolute accuracy, we demonstrated how STF is robust to noise. Figure 11 indicates the accuracy decline as the noise ratio increases. For each method, the accuracy reduction was calculated as follows:

$$v_{method}(i) = Acc_{method}(0) - Acc_{method}(i) \tag{17}$$

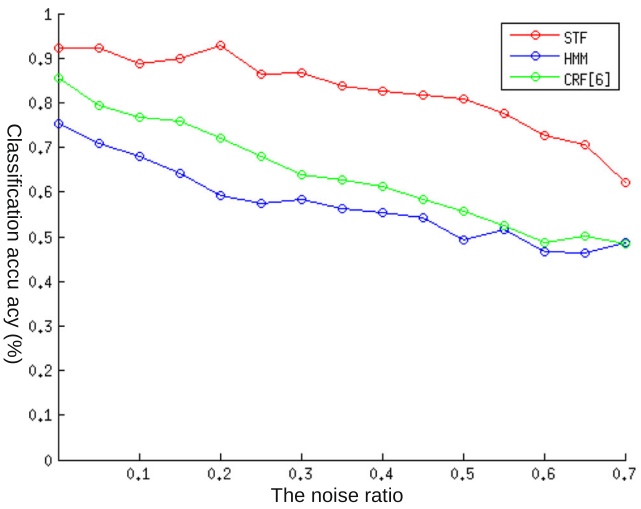


Fig. 10 Experimental results on the American Sign Language database

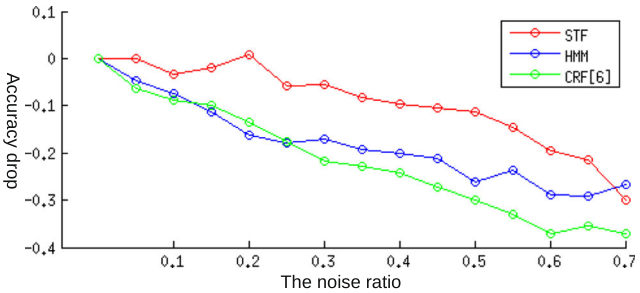


Fig. 11 Accuracy declines as noise ratio increases. The y-axis is the gradient of accuracy decline. It indicates how rapidly or slowly the accuracy diminishes as noise increases

where $method = \{STF, HMM, CRF\}$, i is the noise ratio, and Acc represents the accuracy that is indicated in Fig. 10. This confirms that the accuracy of STF did not decline significantly until $nr = 0.2$, whereas the accuracies of HMM and CRF declined constantly from $nr = 0.05$. This illustrates that the STF method does not only outperforms the previous methods, but is also resistant to noise.

Selecting a helpful feature vector set that is a subset of the input feature vectors over a time axis facilitates considerably higher and more stable classification accuracy.

5 Conclusion

We proposed an STF (Selective Temporal Filtering) method for temporal data analysis that is robust to noise. STF identifies a sequence of key-feature vectors for each classifier by filtering noise and removing non-informative feature vectors. To show its abilities, we applied STF to two gesture recognition datasets. We presented experimental results for a synthetic-dataset and an American Sign Language database. For the synthetic-dataset experiment, the proposed method, employing a simple model using multiple Gaussians, was extremely robust to noise feature vectors. The classification accuracy did not decline as the noise ratio was increased up to 70 %, whereas HMM diminished constantly as the noise ratio increased. The American Sign Language classification experiment was evaluated under various noise conditions. In ideal experimental settings, where the assumption of minimal noise is justifiable, STF achieved an accuracy of 92.1 %, whereas HMM and CRF achieved 75.4 % and 85.64 %, respectively. As the noise ratio was increased up to 70 %, the performance of STF declined more slowly than that of HMM and CRF. Overall, the key-feature vector sequence, identified by the proposed STF, provided reliable performance under significant noise.

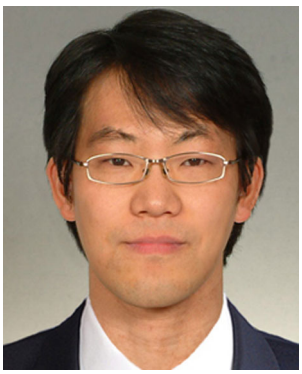
The proposed method demonstrated effective performance in isolated word recognition of the ASL. However, it is not appropriate for *sentence* recognition of the ASL as it cannot distinguish words from sentences. In order to apply

STF to continuous gesture recognition, a spotting method must be considered. In our future research we are planning to extend the proposed method such that it can separate words from sentences.

Acknowledgments This work was partly supported by the ICT R&D program of MSIP/IITP [B0101-15-0552 , Development of Predictive Visual Intelligence Technology] and also supported by the Implementation of Technologies for Identification, Behavior, and Location of Human based on Sensor Network Fusion Program through the Ministry of Trade, Industry and Energy (Grant No. 10041629).

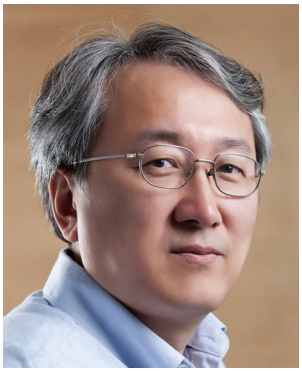
References

1. Rabiner LR (1989) A tutorial on hidden markov models and selected applications in speech recognition. Proc IEEE 77:257–286
2. Starner T, Weaver J, Pentland A (1998) Real-time american sign language recognition using desk and wearable computer based video. IEEE Trans Pattern Anal Mach Intell 20(12):1371–1375
3. Ahmad M, Lee S-W (2008) Human action recognition using shape and CLG-motion flow from multi-view image sequences. Pattern Recognit 41(7):2237–2252
4. Murphy K (2002) Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, University of California, Berkeley
5. Suk H-I, Shin B-K, Lee S-W (2010) Hand gesture recognition based on dynamic bayesian network framework. Pattern Recognit 43(9):3059–3072
6. Yang H-D, Sclaroff S, Lee S-W (2009) Sign language spotting with a threshold model based on conditional random fields. IEEE Trans Pattern Anal Mach Intell 31(7):1264–1277
7. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of Int. Conf. on Machine Learning, pp 282–289
8. Pearson K (1901) On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2(11):559–572
9. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24:381–395
10. Irani M, Anandan P, Hsu S (1995) Mosaic based representations of video sequences and their applications. In: Proceedings of Fifth International Conference on Computer Vision, pp 605–611
11. Lacey AJ, Pinitkarn N, Thacker NA (2000) An evaluation of the performance of RANSAC algorithms for stereo camera calibration. In: Proceedings of the British Machine Vision Conference
12. Roh M-C, Oguri T, Kanade T (2011) Face alignment robust to occlusion. In: Proceedings of IEEE International Conference on Automatic Face Gesture Recognition and Workshops, pp 239–244
13. Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and an application to boosting. Proceedings of the Second European Conference on Computational Learning Theory:23–37
14. Hu K, Yin L (2015) Multiple feature representations from multi-layer geometric shape for hand gesture analysis. In: Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on, vol 1, pp 1–7
15. Nguyen V-T, Le T-L, Tran T-H, Mullot R, Courboulay V (2015) A new hand representation based on kernels for hand posture recognition. In: Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on, vol 1, pp 1–6
16. Yang H-D, Lee S-W (2013) Robust sign language recognition by combining manual and non-manual features based on conditional random field and support vector machine. Pattern Recogn Lett 34:2051–2056
17. Yang H-D (2014) Sign language recognition with the kinect sensor based on conditional random fields. Sensors 15(1):135–147
18. Ghaleb F, Youness E, Elmezain M, Dewdar F (2015) Vision-based hand gesture spotting and recognition using CRF and SVM. J Softw Eng Appl 8:313–323
19. Alon J, Athitsos V, Sclaroff S (2005) Accurate and efficient gesture spotting via pruning and subgesture reasoning. In: Proceedings of International Conference on Computer Vision Workshop on Human Computer Interaction, pp 189–198
20. Lee H-K, Kim J-H (1999) An HMM-based threshold model approach for gesture recognition. IEEE Trans. on Pattern Analysis and Machine Recognition 21(10):961–973
21. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. Proc IEEE Conf Comput Vis Pattern Recognit 1:511–519
22. Yang H-D, Lee S-W, Lee S-W (2006) Multiple human detection and tracking based on weighted temporal texture features. Int J Pattern Recognit Artif Intell 20(3):377–391



Myung-Cheol Roh received his B.S. degree in Computer Engineering from Kangwon University, Chun-Choen, Korea, in 2001, and the MS and PhD degrees in Computer Science and Engineering from Korea University, Seoul, Korea, in 2003 and 2008. Currently, he is working as a managing researcher at S1, Seoul, Korea. He won the best paper award of the 25th annual paper competition which is supervised by the Korea Information Science Society and is sponsored by Microsoft in 2006. He worked at the Center for Vision, Speech and Signal Processing in the University of Surrey, UK, as a collaborate researcher in 2004 and at the Robotics Institue in Carnegie Mellon University, US, as a researcher from 2008 to 2012. His present research interests include face alignment, face and gesture recognition, robot vision and the pattern recognition related fields.

Siamac Fazli received his B.Sc. Physics degree from the University of Exeter in 2002, his M.Sc. in Medical Neurosciences from the Humboldt University Berlin in 2004 and his Ph.D. from the Berlin Institute of Technology in 2011. From 2011-2013 he worked as a Postdoc researcher at the Berlin Institute of Technology for the Bernstein Focus Neurotechnology. Since 2013 he works as an Assistant Professor at Korea University. His current research interests include neuroscience, machine learning, multi-modal neuroimaging and brain-computer interfacing.



Seong-Whan Lee received the B.S. degree in computer science and statistics from Seoul National University, Korea, in 1984, and the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea, in 1986 and 1989, respectively. Currently, he is the Hyundai-Kia Motor Chair Professor at Korea University, Seoul, where he is the Head of the Department of Brain and Cognitive Engineering. His research interests include artificial intelligence, pattern recognition, and brain engineering. He is a Fellow of the IEEE, the IAPR, and the Korean Academy of Science and Technology.