Continuous Finger Gesture Spotting and Recognition Based on Similarities Between Start and End Frames

Gibran Benitez-Garcia, *Member, IEEE*, Muhammad Haris, Yoshiyuki Tsuda, and Norimichi Ukita, *Member, IEEE*

***Abstract*— Touchless in-car devices controlled by single and continuous finger gestures can provide comfort and safety on driving while manipulating secondary devices. Recognition of finger gestures is a challenging task due to (i) similarities between gesture and non-gesture frames, and (ii) the difficulty in identifying the temporal boundaries of continuous gestures. In addition, (iii) the intraclass variability of gestures' duration is a critical issue for recognizing finger gestures intended to control in-car devices. To address difficulties (i) and (ii), we propose a gesture spotting method where continuous gestures are segmented by detecting boundary frames and evaluating hand similarities between the *start* and *end* boundaries of each gesture. Subsequently, we introduce a gesture recognition based on a temporal normalization of features extracted from the set of spotted frames, which overcomes difficulty (iii). This normalization enables the representation of any gesture with the same limited number of features. We ensure real-time performance by proposing an approach based on compact deep neural networks. Moreover, we demonstrate the effectiveness of our proposal with a second approach based on hand-crafted features performing in real-time, even without GPU requirements. Furthermore, we present a realistic driving setup to capture a dataset of continuous finger gestures, which includes more than 2,800 instances on untrimmed videos covering safety driving requirements. With this dataset, our both approaches can run at 53 fps and 28 fps on GPU and CPU, respectively, around 13 fps faster than previous works, while achieving better performance (at least 5% higher mean tIoU).**

***Index Terms*— Hand gesture recognition, gesture spotting, human-computer interaction, automotive user interfaces.**

I. Introduction

HAND gesture recognition (HGR) is an essential part of human-computer interaction. In particular, touchless automotive user interfaces (AUI) controlled by hand and finger gestures can provide comfort and safety on driving while manipulating secondary devices like audio and navigation systems [1]–[3]. Besides, some essential in-car devices can also be controlled with finger gestures. For example, the wipers can be activated by detecting a denial gesture performed with one finger. Furthermore, continuous finger gestures are convenient for AUI, since they allow multiple commands on different functions. For instance, the audio control of 'rewind' can be activated with an isolated 'flicking-left' gesture, while the 'skip to the start' function can be assigned to a combination of two continuous 'flicking-left' gestures. The main advantages of AUI controlled by finger gestures include lower visual load, nonintrusive performance, and high-level user acceptability [4]–[7].

Finger gestures can be dissected on three motion states: preparation, nucleus, and retraction [8]. The message in a finger gesture is mainly contained in the nucleus state, which presents the most representative appearance and motion attributes of the gesture [9]. Transitions between motion states can be considered as boundaries of the gesture's nucleus. These boundaries exist even between continuous nucleus states, as shown in Figure 1. Thus, it is possible to rely on boundary detection for spotting the nucleus of finger gestures. Still, it is not easy (i) to detect boundary frames because they are similar to other gesture frames, and (ii) to identify a pair of boundary frames that correspond to the *start* and *end* frames of continuous gesture. These problems may also be observed in other gesture recognition applications. However, these are critical for continuous finger gesture recognition intended to control AUI, as illustrated in Figure 1. In addition, a frequent gesture recognition difficulty (iii) is the intraclass variability of gestures' duration, as can be seen in Figure 1. This issue is crucial for finger gesture recognition due to there are gestures significantly longer than others. Besides, preparation and retraction states might share similarities with the nucleus of some flicking gestures. For instance, flicking-down gestures look partially similar to the retraction state.

In this paper, we propose a continuous finger gesture spotting and recognition method. Our gesture spotting method resolves difficulties (i) and (ii) by detecting boundary frames of gesture's nucleus when gestures are performed continuously, as shown in the top row of Figure 1. We evaluate the hand similarities between the *start* and *end* frames of each gesture under the assumption that a gesture's nucleus starts and ends with a similar position and pose of the hand. Note that this assumption is particularly valid for finger gestures intended to control AUI, such as flicking gestures illustrated in Figure 1. However, it is not applicable in all the wide range of possible hand gestures. We name *Similarity Check* to the evaluation of

Manuscript received August 30, 2019; revised April 21, 2020; accepted July 15, 2020. The Associate Editor for this article was S. S. Nedevschi. *(Corresponding author: Gibran Benitez-Garcia.)*

Gibran Benitez-Garcia, Muhammad Haris, and Norimichi Ukita are with the Department of Advanced Science and Technology, Toyota Technological Institute, Nagoya 468-8511, Japan (e-mail: gibran@toyota-ti.ac.jp; mharis@toyota-ti.ac.jp; ukita@toyota-ti.ac.jp).

Yoshiyuki Tsuda is with DENSO Corporation, Kariya 448-8661, Japan.

Digital Object Identifier 10.1109/TITS.2020.3010306

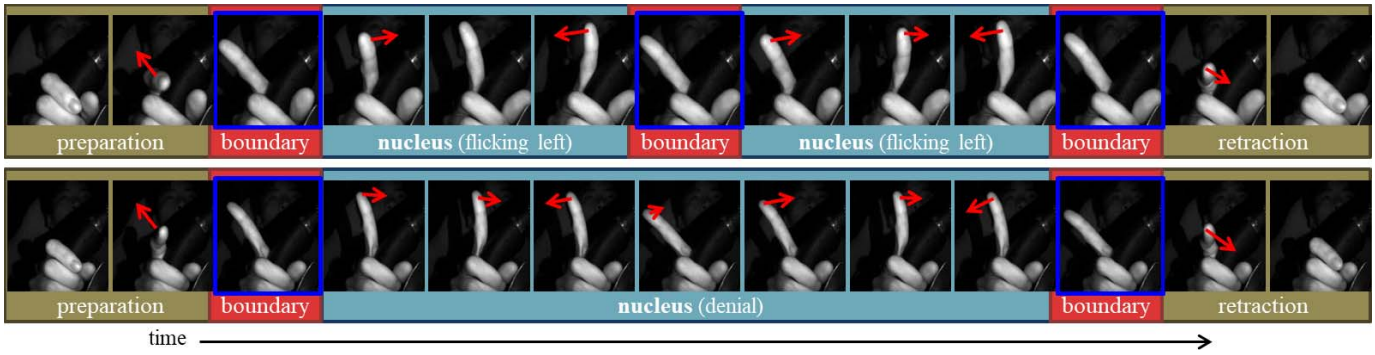
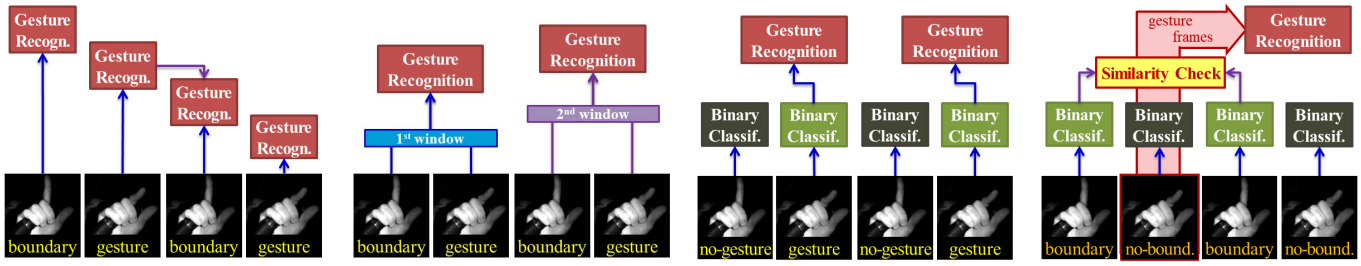


Fig. 1. Example of the properties of continuous and isolated finger gestures. The top row shows a sequence of two continuous gestures (flicking-left). The bottom row shows a sequence of an isolated gesture (denial). The red arrows illustrate the finger motion. Frames that share strong similarities are highlighted in blue. Note the duration inconsistency of both gestures; denial takes roughly twice as the time of flicking-left.



(a) Linked frame-wise classification (b) Sliding windowing (c) Binary classification (d) Our approach

Fig. 2. Comparison of continuous HGR based on their gesture spotting approach. (a) Gesture recognition is applied frame by frame. Gesture spotting is based on a linking process using confidential scores [16], [17]. (b) Gesture spotting and recognition based on temporal sliding windows [18], [19]. (c) Classification between gesture and non-gesture frames. Gesture recognition is performed on gesture frames only [20]–[22]. (d) Our proposal is based on boundary/non-boundary detection. Similarity check evaluation is required for defining target frames employed for gesture recognition.

hand similarities that employs features extracted from detected boundary frames, as highlighted in Figure 2 (d). For gesture recognition, we overcome difficulty (iii) by using a temporal normalization of features extracted from the set of spotted frames. We merge these frames into a limited number of features able to represent the gesture's nucleus. Thus, we can also overcome the duration inconsistency that may appear between different gestures.

Based on our core idea, we present two different alternatives for reaching real-time performance: 1) a deep-network based approach oriented to exploit a GPU if enabled; 2) a traditional approach with hand-crafted features which can perform when only the CPU is available. For (1), we propose a framework employing compact deep neural networks trained with a triplet-loss function [10]. This function is suitable for discriminating boundary frames, since it learns to decrease the Euclidean distance of true boundaries embeddings while increasing that between false-positive boundary frames. For (2), we use histogram of oriented gradients (HOG) [11] and histogram of optical flow (HoOF) [12] features classified with Random Forest (RF) [13] to detect boundary frames.

To validate our method, we propose a realistic setup for finger gesture recognition, which considers the driving distractions identified by the National Highway Traffic Safety Administration (NHTSA) [14]. We define eight gestures which are natural to perform by keeping both hands on the steering wheel, and demand a low cognitive load. With this setup, we capture a dataset of continuous finger gestures with more than 2,800 instances on untrimmed videos.

In summary, the main contributions of our work include:

- A novel finger gesture spotting method based on boundary detection, which introduces a *Similarity Check* between detected boundaries (Section III-B).
- A gesture recognition approach based on a temporal normalization of features extracted from the set of spotted frames (Section III-C).
- A realistic setup for finger gesture recognition designed to capture a dataset of continuous gestures on untrimmed videos.

This work is an extension of the conference paper presented on MVA2019 [15], which introduces our deep-network approach (Section IV). The novel contributions of this paper are threefold:

- A real-time finger gesture spotting and recognition based on boundary frame similarities using hand-crafted features capable of working on CPU (Section V).
- A detailed description of the setup designed to cover safe driving requirements when controlling touchless AUI (Section VI).
- Performance comparisons of our proposals with previous methods for real-time gesture recognition (Section VII).

II. Related Work

Most of the HGR state-of-the-art approaches are applied to isolated gestures, assuming that only one gesture is included in each video of the dataset [23]–[26]. Based on the method for obtaining characteristics of gestures, there exist two

groups of HGR approaches: hand-crafted and deep-network based [27], [28]. Hand-crafted methods primarily employ HOG and HoOF features [3], [20], [27], [29]. For instance, Ohn-Bar and Trivedi [3] evaluated several variations of HOG and classifiers to recognize hand gestures for automotive interfaces. Joshi *et al.* [20] used HOG features for classifying upper body gestures with Random Forest (RF). Borghi *et al.* [29] employed HoOF and support vector machines (SVM) for modeling the gestures in a spatio-temporal feature space. Similarly, for our hand-crafted based approach, we propose to merge a combination of HOG and HoOF features from limited temporal sections to model the motion and appearance properties of finger gestures.

Previous works focused on learned feature extraction methods tend to use deep convolutional neural networks (CNN) and 3D CNN, as presented on the ChaLearn Look At People (LAP) gesture recognition challenge [30]. ChaLearn LAP is based on two datasets: Isolated and Continuous Gesture Dataset (IsoGD and ConGD) [31]. The winners of the last IsoGD challenge, Miao *et al.* [24], employ a well-known 3D CNN model called C3D [32] to extract features from RGB, depth, and flow fields. They propose a feature level fusion within each modality, and use SVM for classifying the fused features. One year after that challenge, Narayana *et al.* [25] overcome the results by proposing a late-fusion approach from 12 different channels, comprising focus regions of global, left and right hand, including modalities of RGB, depth, and flow. In contrast to those methods, we focus on rendering the network compact while maintaining acceptable accuracy. Therefore, we build our approach on a compact CNN model originally designed to perform on mobile devices. For gesture recognition, we include a few extra convolutional layers to reuse feature maps used in the gesture spotting.

On the other hand, ConGD challenge participants [30] and recent continuous HGR works also use CNN and 3D CNN with multi-modal data inputs [16], [21], [22], [33], [34]. Based on the gesture spotting approach, these methods can be primarily divided into three types, as shown in Figure 2 (a), (b), and (c).

A. Linked Frame-Wise Classification

Proposed by Singh *et al.* [16], this approach classifies gestures frame by frame. Subsequently, gesture spotting is based on a process of linking the frame's confidential scores to limitate continuously detected gestures, as shown in Figure 2 (a). Similarly, Kalogeiton *et al.* [17] proposed to use the SSD (Single Shot MultiBox object Detector) [35] for constructing 'temporal action tubes' from the spatial frame-level detections. One critical issue of this approach comes when boundary frames are similar to gesture frames. In that case, continuous gestures might be misrecognized as a single isolated gesture. Figure 1 illustrates this issue, where two continuous flicking-left gestures (top row) might be misrecognized as a single denial gesture (bottom row).

B. Sliding Windowing

Hand gestures are directly recognized with temporal sliding windows [18], [19]. In this approach, gesture recognition and spotting are achieved simultaneously, since the temporal window length determines the gesture location. For instance, C3D [32] can be used to recognize gestures using non-overlapped sliding windows, as the example shown in Figure 2 (b). Thus, a gesture is spotted by the temporal window that recognizes it correctly. Recently, Zolfaghari *et al.* [18] proposed a 3D CNN method with temporal pooling using sliding windows with 50% overlap. They propose to update the prediction of 3D CNN every window, by using a working memory. A drawback of this approach is the fixed-length windows, which limit its performance when gestures with significant length variations continuously appear.

C. Binary Classification

This approach aims to binarize each frame to gesture or non-gesture frames in order to extract the frame sequence used for gesture recognition [20]–[22]. As shown in Figure 2 (c), consecutive gesture frames are feed to the recognition step, while non-gesture frames are discarded. For instance, Joshi *et al.* [20] proposed Random Forest to distinguish between gesture and non-gesture frames using 3D joint-based and appearance features. In a recent approach, Kopuklu *et al.* [22] propose a hierarchical structure of 3D CNN architectures to detect and recognize continuous hand gestures. The detection discriminates between gestures and non-gestures using a shallow 3D CNN model, while the recognition is carried out by a deep 3D CNN using weighted average filtering to take a single-time activation per gesture. A variation of the *binary classification* evaluates specific properties of boundary frames for performing the gesture spotting [36], [37]. For instance, in [37]–[39], the quantity of movement (QOM) [40] is employed to determine gesture boundaries by detecting motionless from depth frames.

Binary classification is a simple yet effective approach commonly used on several works, including the winners of the ChaLearn LAP ConGD challenge [21], [22], [36]–[39], [41]. However, the binarization process might extract a sequence, including multiple gestures because of any non-gesture frames might not be detected between continuous gestures. Contrastively, false-positive errors of boundary frame detection may result in isolated gestures unintentionally segmented in several parts. Therefore, we overcome these issues by proposing a *Similarity Check*, which evaluates the similarities between the *start* and *end* boundary frames of the gesture's nucleus, as shown in Figure 2 (d).

III. Proposed Method

The overview of our proposed framework is illustrated in Figure 3. Let I_t be the current frame and also the *end* boundary of a gesture. The gesture consists of a sequence of $S+1$ frames $\{I_{t-S}, \ldots, I_{t-1}, I_t\}$, where I_{t-S} represents the *start* boundary frame. The goal of our proposal is to recognize the gesture that takes place between I_{t-S} and I_t from a sequence of an arbitrary number of frames, which includes continuous gestures. Our proposal can be divided into three stages: feature extraction, gesture spotting, and gesture recognition. The basic process flow consisting of these three stages is shared by our deep-network and hand-crafted based approaches. Nevertheless, the differences among these

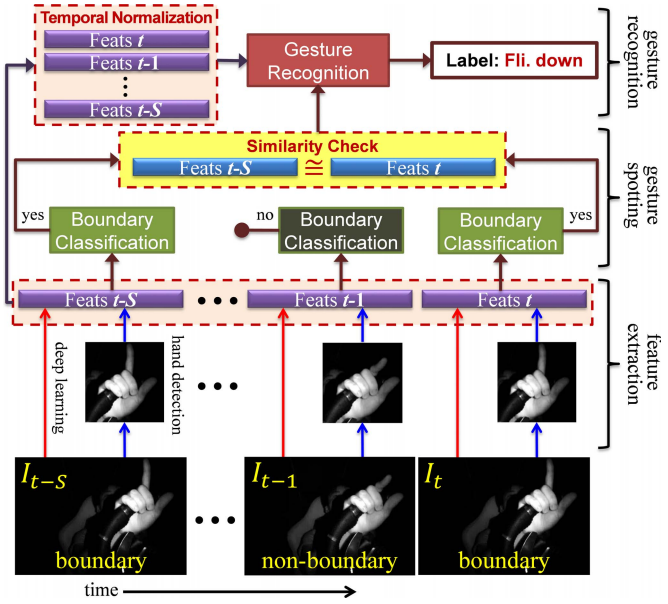


Fig. 3. Overview of our proposal. Based on the feature extraction method, the framework has two approaches. For the deep-network approach (path shown with the red arrows), the hand location is regressed altogether with the boundary classification, while for the hand-crafted features approach, the hand is detected before feature extraction (path shown with the blue arrows). With these features, we classify between boundary and non-boundary each frame of the input sequence. For gesture spotting, we propose to evaluate similarities between two detected boundaries. Hence, the gesture is spotted if the similarity check between the *start* and *end* boundaries is successful. The gesture class is recognized based on a temporal normalization of the features extracted from the set of spotted frames.

TABLE I
DIFFERENCES AMONG OUR BOTH APPORACHES

	Feature extraction	**Gesture spotting**	**Gesture recognition**
Deep-network based	SSD lite with MobilenetV2	*Similarity Check* using features learned with a triplet-loss function	Temporal stacking of features merged with a temporal pooling, and classified with softmax
Hand-crafted based	HOG and HoOF	*Similarity Check* using HOG features only	Features merged with a blending function, and classified with RF

approaches are resumed in Table I and detailed in the following subsections.

A. Feature Extraction

We handle two different types of hand features for gesture spotting and recognition, respectively. Similarity features (X_{sm}) are used for gesture spotting, while gesture features (X_{gs}) for gesture recognition. The extraction process of both hand features is carried out differently for each of our two approaches, as shown in Table I. The deep-network approach can simultaneously detect the hand and extract its gesture features, as proposed in SSD [35]. Besides, we use a triplet-loss function [10] for learning the similarity features, path shown with the red arrows in Figure 3. For the hand-crafted based approach, we apply a simple hand detection process based on the hand motion. Subsequently, the appearance and motion characteristics of the hand are defined by HOG and HoOF features, path shown with the blue arrows in Figure 3.

Algorithm 1 Continuous Gesture Spotting Process

Input: Boundary classification score (p), similarity (X_{sm}) and gesture features (X_{gs}) of each frame (I_i) of the input sequence
Output: Gesture features from the set of spotted frames

1 **Initialize** a boundary clip ($R \leftarrow \emptyset$), a boundary clip's working memory ($W_R \leftarrow \emptyset$), and a features' working memory ($W_X \leftarrow \emptyset$)
2 **if** $p > 0.5$ **then**
3 $R \leftarrow (p, X_{sm})$
4 **else**
5 **if** $length(R) \geq 2$ **then**
6 $W_R \leftarrow R$
7 **if** $length(W_R) > 0$ **then**
8 $W_X \leftarrow X_{gs}$
9 **if** $length(W_R) \geq 2$ **then**
10 **for** $m = 1$ *to* $length(W_R) - 1$ **do**
11 $start \leftarrow X_{sm}$ of $max(p \in W_R(m))$
12 **for** $n = m + 1$ *to* $length(W_R)$ **do**
13 $end \leftarrow X_{sm}$ of $max(p \in W_R(n))$
14 **if** $Dist(start, end) > Th$ **then**
15 CLEAR: $W_R(1 : n - 1)$
16 OUTPUT: $W_X(i_{start} : i_{end})$
17 CLEAR: R, W_R & W_X **if**$\{length(W_X) > T\}$

B. Gesture Spotting

This process is achieved by two steps: boundary classification and *Similarity Check*. As shown in Figure 3, the boundary classification is performed frame-wise, while the similarity check is only applied when two potential boundaries are detected. We define the *Similarity Check* as the Euclidean distance between similarity features. Hence, the gesture is spotted if the Euclidean distance between the *start* and *end* boundary frame's features is greater than a similarity threshold (Th). The complete process of our continuous gesture spotting proposal is detailed in Algorithm 1. The inputs are the previously extracted features (X_{sm} and X_{gs}), and the probability score (p) of the boundary classification from each frame (I_i). Firstly, to avoid isolated false-positive detections, we define a boundary just if at least two consecutive boundary frames are detected (lines 2-3 of Algorithm 1), we called this a *boundary clip* (R). After detecting the first boundary clip (lines 5-6), we store X_{gs} of consecutive non-boundary frames in a working memory (lines 7-8). Subsequently, when a second boundary clip is detected (line 9), the *Similarity Check* is applied using the similarity features corresponding to the boundary frames that have the highest probability score (lines 11 and 13, respectively). Finally, we spot a gesture if $Dist(start, end) > Th$, where $start$ and end represent

TABLE II
INPUTS FOR ALGORITHM 1 FROM EACH APPROACH

	Similarity features	**Gesture features**	**Boundary classification score**
Deep-network based	Learned using triplet-loss (ε from Eq. 2)	Learned with MobilenetV2 (X in Eq. 2)	Output of softmax from SSD lite
Hand-crafted based	HOG features (V_{hog} from Eq. 4)	HOG + HoOF features (B from Eq. 6)	Fusion of RF outputs using HOG and HoOF features (as in Eq. 5)

the X_{sm} of frames with $max(p)$ from the first and second boundary clips, respectively. The output will be the gesture features corresponding to all frames between the *start* and *end* temporal boundaries (line 16). Otherwise, the boundary clips are kept in the working memory (W_R) until the duration between the first boundary clip and the current frame is no longer than T frames (line 17). We have set $T = 45$ based on the longest gesture from our dataset. Note that since we use different features for our both approaches, the inputs of Algorithm 1 also differ. Table II presents the inputs for each method, which are further detailed in the following sections.

C. Gesture Recognition

As shown in the upper part of Figure 3, we adopt a temporal normalization of features extracted from the set of spotted frames, which is used to represent each spotted gesture. This normalization consists of summarizing the whole gesture into a limited number of temporal sections. Given that, gestures with different duration can be defined with the same amount of features while maintaining its temporal order. Hence, all the spotted frames ($S + 1$) are merged into K temporal sections, which are further concatenated to represent the whole gesture. As shown in Table I, the way to merge the extracted features differs based on the approach. For the deep-network based approach, we apply a temporal stacking of features before a temporal pooling on K sections. Afterwards, we use softmax for gesture recognition. For the hand-crafted based approach, we concatenate HOG and HoOF features from each frame before applying a blending function on K sections. The concatenated sections are finally classified using RF. Note that K is a tunable parameter which defines the ideal length for representing all gestures from the dataset.

In the next two sections, we detail specific information for each of our two approaches.

IV. DEEP-NETWORK BASED APPROACH

A. Feature Extraction

The overview of the architecture for feature extraction and boundary classification is shown in Figure 4. We build our proposal on the SSD detector [35], which classifies boundary frames and regresses the bounding box of the hand in a single-stage architecture. To ensure real-time performance, we use a faster version named SSD lite [42], which implements a compact CNN architecture called MobilenetV2 as a base network. This architecture can perform on real-time even on

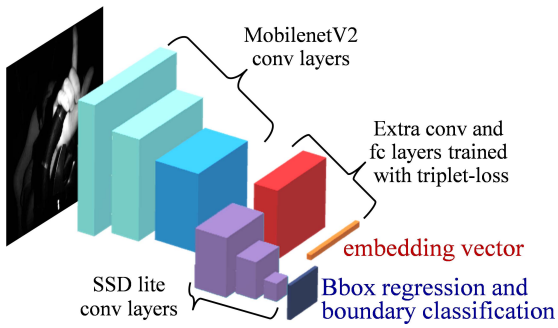


Fig. 4. Overview of the proposed architecture for feature extraction and boundary classification from the deep-network approach. We use the SSD lite architecture [42], and we propose a few extra layers trained with the triplet-loss function [10]. From the current frame, we extract convolutional feature maps using MobilenetV2. Boundary classification and hand detections are achieved by SSD lite. If a boundary frame is detected, we feed the extracted feature maps into the extra layers to define an embedding vector.

mobile devices, since it exploits the depth-wise convolution and introduces an inverted residual block of convolutions. SSD lite includes 17 inverted residual convolution layers (*MobilenetV2 conv. layers* colored in blue in Figure 4) and three depth-wise convolution layers (*SSD lite conv. layers* colored in purple in Figure 4) for bounding box regression and boundary classification.

After boundary classification, we use a triplet-loss function to learn features that can define the temporal boundaries. The triplet-loss function [10] employs triplets of samples (anchor, positive, negative) for training the network. Anchor and positive samples are chosen from boundary frames, while non-boundary frames refer to the negative samples. This loss function aims to minimize the Euclidean distance between the anchor and "difficult positive" samples, while penalizing that between the anchor and "difficult negatives." A difficult positive has a large Euclidean distance with the anchor. Conversely, difficult negatives present a short distance, although they belong from different classes. This phenomenon often occurs when detecting boundary frames of finger gestures. Therefore, we add a depth-wise convolution and a fully-connected layer for learning embeddings capable of representing similarities of boundary frames. Figure 4 shows these extra layers, colored in red and orange, respectively. The triplet-loss is defined by:

$$L = \sum_{i}^{O} \left[||f(X_i^a) - f(X_i^p)||_2^2 - ||f(X_i^a) - f(X_i^n)||_2^2 - \alpha \right] \quad (1)$$

where α is a margin that is enforced between positive and negative pairs from X^a (anchor), X^p (positive), and X^n (negative) samples' features, $f(X)$ is the embedding of features X in to a Q-dimensional Euclidean space, and O is the number of samples per class in the batch. In our proposal, embeddings are defined as:

$$\varepsilon = f(\psi(X)) \quad (2)$$

where X refers to the feature maps obtained from the last convolutional layer of MobileNetV2 (colored in electric blue in Figure 4), and $\psi(X)$ represents the two added extra layers designed to learn the similarity features.

B. Gesture Spotting

The gesture spotting is based on the similarity features learned with the triplet-loss function, and the gesture features obtained from the base network (in our case feature maps from MobilenetV2). Therefore, as shown in Table II, the inputs of Algorithm 1 are $X_{sm} = \varepsilon$, $X_{gs} = X$, and p as the output of the softmax function used in the SSD lite architecture.

C. Gesture Recognition

Inspired by Zolfaghari *et al.* [18], we adopt a temporal stacking of feature maps as the first step of gesture recognition. The intuition behind the temporal stacking is that feature maps may contain similar information in the same temporal order, since the base network with shared weights is applied frame-wise. In this way, we keep the temporal structure of the features before merging them by a temporal pooling.

Let X_j^{ϕ} be the j-th feature map, $j \in \{1, 2, \ldots, C\}$, from the ϕ-th frame, $\phi \in \{t - S, \ldots, t - 1, t\}$, where C is the number of channels at the last convolutional layer of MobilenetV2, and t is the current frame (the detected *end* boundary). Thus, the temporal stacking is defined as a $(M, N, (S+1)C)$ tensor:

$$X' = \left[X_1^{t-S}, X_1^{t-S+1}, \ldots, X_C^{t-1}, X_C^{t}\right] \tag{3}$$

Subsequently, we apply adaptive average temporal pooling to reduce the size of X' to (M, N, KC), where $M = N$ is the spatial size of the feature maps, and KC is the temporal size, obtained by averaging features from non-ovelapped groups of $((S + 1)C)/KC$ channels. In other words, we merge the $S + 1$ spotted frames into K temporal sections. Temporal pooled features are feed into two depth-wise convolutional layers with batch normalization, ReLu6 non-linearity, and dropout. Finally, the last fully-connected layer uses softmax to determine the class of the spotted gesture.

V. Hand-Crafted Based Approach

A. Feature Extraction

In order to ensure real-time performance, we define a simple hand detection process based on the hand motion detected around the steering wheel. Firstly, we predefine a steering wheel template, and compute a dense optical flow (OF) frame-wise. Subsequently, we define the hand location by finding the region within the steering wheel template that concentrates the hand motion uniformly during n frames. The results of hand detection are approximately 99% with $IOU@0.5$ and $n = 5$. Even with our simple approach, we can perform hand detection correctly. Still, the few detection errors are attributed to issues of motion registration by the optical flow approach.

After hand detection, we classify boundary frames based on appearance and motion features, as shown in Figure 5. To include additional temporal information to the hand features (HOG and HoOF, respectively), we compose a feature vector assembled from $L + 1$ per-frame features, given by:

$$V = \left[X_{t-L}, \ldots, X_{t-1}, X_t\right] \tag{4}$$

where V is the assembled feature vector of the current frame (t), and X represents either the HOG or HoOF features.

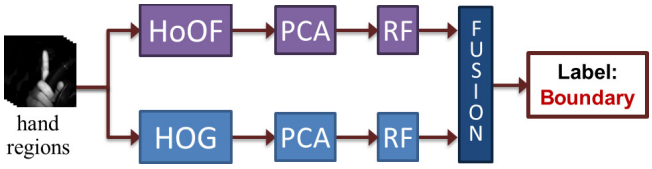


Fig. 5. Diagram of our boundary classification proposal based on hand-crafted features. Principal component analysis (PCA) and Random Forest (RF) are applied to the HOG and HoOF hand features. The final classification is made by fusing confidential scores from both features.

Subsequently, we apply principal component analysis (PCA) for dimensionality reduction. Thus, vectors from each type of features (V_{hog} and V_{hof}) are independently classified by RF. Note that the independent feature vector extraction and classification of both features can be run in parallel, as shown in Figure 5.

The final classification is made by a late fusion, where the confidential scores from HOG and HoOF features are fused to take the final decision. This fusion is defined by:

$$p = \beta * p_{hog} + (1 - \beta) * p_{hof} \tag{5}$$

where p is the merged score probability of a sample, which is obtained from the probabilities p_{hog} and p_{hof}, corresponding to the RF outputs using appearance and motion features, respectively. Finally, the contribution of each feature is regulated by the parameter β.

B. Gesture Spotting

The gesture spotting takes only the hand's appearance features as similarity features. Contrastively, gesture features are defined by the combination of appearance and motion features, given by:

$$B = [V_{hof}, V_{hog}] \tag{6}$$

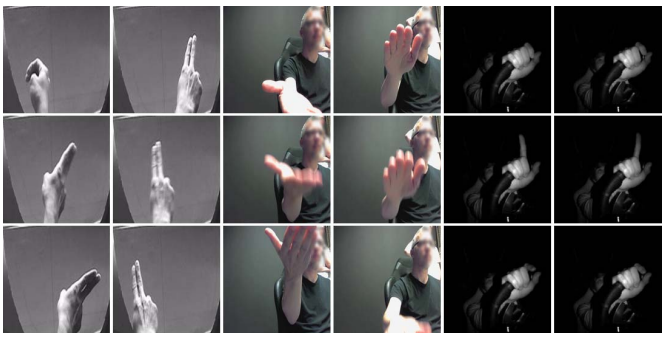
where B is the concatenation of HoOF and HOG assembled feature vectors (from Eq. 4), respectively. As shown in Table II, the inputs of Algorithm 1 are $X_{sm} = V_{hog}$, $X_{gs} = B$, and p which is obtained with Eq. 5. Note that, as for similarity features, we do not consider the motion features (V_{hof}) because motion from the *start* and *end* boundary frames are different given that the finger is moving towards opposite directions at these two moments.

C. Gesture Recognition

Inspired by the work of Joshi *et al.* [20], we apply a temporal normalization on the set of $[B_1, B_2, \ldots, B_{S+1}]$ feature vectors from the $S+1$ spotted frames. Thus, the whole gesture is represented by a final feature vector, given by:

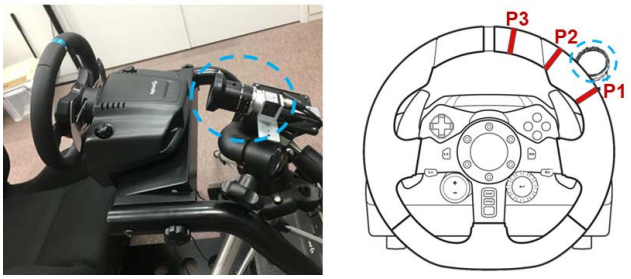
$$H' = \left[H_1, H_2, \ldots, H_K\right] \tag{7}$$

where K is the number of the defined temporal sections, and H_i represents the features of the i-th temporal section, which are calculated by $\varphi([B_1, \ldots, B_P])$, where φ is a blending function, and $P = (S + 1)/K$ is the number of feature vectors corresponding to the temporal section sub-set. Given that, all spotted frames are represented by a concatenation



(a) VIVA dataset. (b) Nvidia dataset. (c) Our dataset.

Fig. 6. Comparison of opposite gestures from three datasets. (a) swipe-right, and swipe-left; (b) hand-up, and hand-down; (c) flicking-down, and flicking-up. Each column illustrates a single hand gesture. The first, second and third rows show the starting, middle and ending frames of each gesture motion, respectively.



(a) Data collection setup. (b) Hand positions.

Fig. 7. Environment for data collection. The fixed camera location is highlighted in blue.

of K temporal sections. We tested (i) mean average filtering, (ii) median filtering, and (iii) median vector as the blending function. We experimentally found the best performance by using (ii) median filtering as φ, which is defined as follows:

$$\varphi(B_i) = \left[med([B_1^1, \ldots, B_P^1]), \ldots, med([B_1^D, \ldots, B_P^D])\right] \quad (8)$$

where med represents the median value of each feature from the sub-set, and $D = length(B)$ is the dimension of the P vectors corresponding to the i-th temporal section.

The gesture recognition is performed by classifying the final vectors (H') using RF. The ideal number of trees and features to consider when looking for the best split in our random forest models were determined using the Out-of-Bag (OOB) error rate in a combination of training and validation sets.

VI. Dataset

Most of the publicly available driver's hand gesture datasets are impractical for real-world applications due to 1) not considering possible driving distractions induced by proposed gestures; 2) including a huge set of different gestures making difficult to memorize; 3) presenting a lack of continuous hand gestures in untrimmed videos.

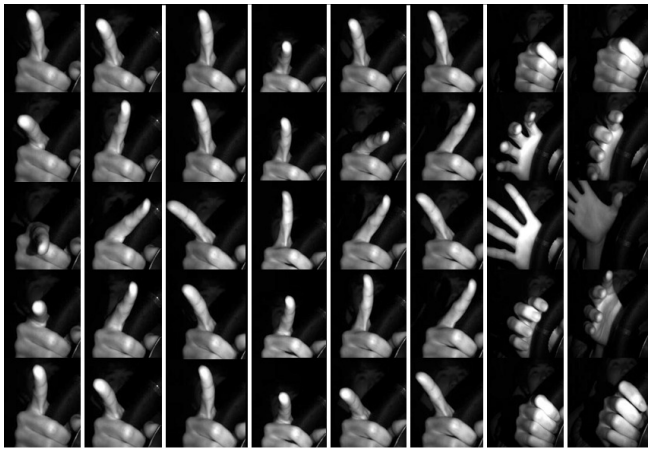
For instance, VIVA [3] and Nvidia [43] datasets consist of videos with isolated gestures, including more than 20 different gestures. Both present two different points of view of gestures captured from a similar region of action, located close to the center dashboard. Thus, the driver has to release his hand from the steering wheel, be aware of the position where the sensor will capture the motion, and try to remember a gesture from the huge set. Moreover, these datasets lack real-world continuous hand gestures. These issues arise a need for suitable databases to evaluate real-world performance under optimal safety conditions.

Gestures performed while handling the steering wheel may be the solution of problems presented in previous datasets. However, these gestures are limited to finger gestures making the automatic temporal segmentation and classification more challenging. For example, from Figure 6 (a-b) we can note that opposed gestures clearly show differences in appearance and motion. On the other hand, flicking gestures with one finger present the opposite situation, as shown in Figure 6 (c).

Given the limitations of existing datasets, we propose a realistic setup designed to capture a dataset considering the driving distractions identified by the National Highway Traffic Safety Administration [14]. Thus, the proposed gestures demand a low cognitive load, and are natural to perform by keeping both hands on the steering wheel. In addition, we choose a near-infrared camera (NIR) to register the finger gestures. This camera type allows developing vision-based systems that can work in different light conditions.

We use an acA2040-90umNIR Basler ace USB 3.0 camera to acquire NIR videos at 30 fps. The location of the camera is close to the center dashboard, 30cm behind the steering wheel with a positive angle of 16 degrees. We simulate two driving environments differing only on the steering wheel dimensions, employing a 278 mm (shown in Figure 7 (a)) and 380 mm wheel diameters, respectively. Twenty subjects participated in data collection over three different sessions. Subjects recorded gestures with their right hand starting and ending from a neutral position (full steering wheel grip) located in three different spots marked in the steering wheel, as shown in Figure 7 (b). Participants were instructed to perform continuous gestures one right after the other before returning to the neutral position, as shown in the top row of Figure 1. As shown in Figure 8, we define eight finger gestures that can be divided into two groups: flicking (*down, left, right & up*) and indications (*circle, denial, open & release*). The size of each frame was normalized to 640 × 480 pixels, 8-bit depth.

In total, our dataset comprises more than 2,800 instances. We randomly split the data into training, validation, and testing sets, with 1536 (192 per class), 432, and 928 instances, respectively. The validation set consists of 336 and 96 instances per flicking (84 × 4) and indications (24 × 4) classes, respectively. Meanwhile, the testing set includes 704 (176 × 4) and 224 instances (56 × 4), respectively. Note that the amount of instances per class group is balanced only in the training set, being about 60% more cases of flicking gestures than those of indications. The complete dataset was frame-wise annotated by specifying four motion states: rest, preparation/retraction, boundary, and nucleus. Besides, bounding boxes of driver's hand were annotated for the training set only.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 8. Example of the eight classes of our dataset at first position (P1). The temporal order of each gesture progresses from the top to bottom rows. (a) Flicking-Down (FD), (b) Flicking-Left (FL), (c) Flicking-Right (FR), (d) Flicking-Up (FU), (e) Circle (Ci), (f) Denial (De), (g) Open (Op), and (h) Release (Re). Note that only the nucleus of each gesture is shown.

VII. Experimental Results

A. Implementation and Training Details

1) Deep-Network Based Approach: We implement all deep-network models using PyTorch 0.4 on a server with i7-6850K CPU @ 3.60GHz, and a single Nvidia GeForce GTX 1080. We train the SSD lite model using mini-batch SGD with Nesterov momentum of 0.9, and mini-batches of size 64. The initial learning rate was 0.01 and decayed by 0.98 every epoch. The MobileNetV2 architecture was reduced 25% from its original size. The triplet-loss network was trained using Adam optimizer with a learning rate of 0.0001 with exponential decay after 150 epochs, $O = 32$, and $\alpha = 0.2$. The extra depth-wise convolutional layer uses 120 1×1 filters, and the fully connected layer includes 60 hidden neurons (i.e., $Q = 60$). We experimentally set $Th = 0.55$. We vary the value of K between the length of the shortest and the longest gesture (in terms of the number of frames), we get the best results with $K = 8$. The last two depth-wise convolutional layers for gesture recognition, include 960 and 480 1×1 filters, respectively. Given that, the stacked feature maps are pooled to $10 \times 10 \times 1920$. For all models, we train with image crops of 300×300 pixels using random crop augmentation, as proposed by the original SSD method [35].

2) Hand-Crafted Based Approach: We conduct our experiments on a PC with Intel Core i7-7820HK CPU @ 2.90GHz, and 16GB of RAM. DIS optical flow [44] was used for hand detection and HoOF calculation with parameters: $\theta_{sf} = 1$, $\theta_{it} = 12$, $\theta_{ps} = 8$, and $\theta_{ov} = 3$ (pixels). The input frame was downscaled by a factor of two before the OF application. The hand region was resized to 24×24 pixels before HOG application. Note that different RF models were trained based on the three positions of the hand in the database. Thus, the hand detection process was also in charge of locating the hand within three specified zones. The rest of the parameters were experimentally set as $L = 2$, $\beta = 0.6$, $K = 8$, and $Th = 0.52$.

TABLE III

Comparison of Gesture Spotting Results Between Our Proposal Variations. Red Here and in the Following Tables Indicates the Best Performance

Method	tIoU@0.3	tIoU@0.5	tIoU@0.8
HC baseline	0.56	0.52	0.39
HC simil	0.68	0.63	0.54
DN baseline	0.73	0.68	0.59
DN simil	0.76	0.71	0.61
DN trip-simil	0.82	0.77	0.67

B. Ablation Studies

We evaluate the gesture spotting performance based on the class-aware temporal Intersection-over-Union score (tIoU). The score counts when the gesture label is correctly predicted, and the IoU between predicted and ground-truth temporal boundaries (starting and ending frames) is higher than a detection threshold. To specifically analyze the gesture recognition performance, we present confusion matrices evaluated with a class-agnostic tIoU using a threshold of 0.5 (tIoU@0.5).

We name *DN trip-simil* and *HC simil* to our deep-network (Section IV) and hand-crafted (Section V) based proposals, respectively. We also present results of *DN simil*, a variation of our deep-network approach which excludes the triplet-loss network. Hence, the *Similarity Check* is based on a hot-vector extracted from the last 1×1 convolutional layer of the SSD lite. Since this approach does not learn specific features for evaluating the similarity, it is comparable to our *HC simil* proposal, which uses only HOG features for this task.

We consider two baselines regarding deep-network and hand-crafted based approaches: *DN baseline* and *HC baseline*. *DN baseline* does not include the *Similarity Check* nor the triplet-loss network. Thus, the gesture spotting is based on the consecutive non-boundary frames detected between boundary frames, which were classified by the SSD lite detector focused on hands. Still, the gesture recognition process is as described in Section IV-C, the same as that of *DN simil* and *DN trip-simil*. On the other hand, the gesture spotting of *HC baseline* is based only on non-boundary classification as described in Section V-A, while the gesture recognition process is the same as that of *HC simil* (Section V-C).

Table III presents the results of baselines and variations of our approaches. Our intuition suggests, and the results confirm, that the baselines would be weaker than our proposals, since they cannot exclude false-positive boundary detections that *Similarity Check* can. This ability is further improved by learning the similarities of gesture boundaries, as proposed in *DN trip-simil*. Deep-network based methods perform better than those based on hand-crafted features. Besides, *HC simil* significantly outperforms the results of *HC baseline*.

The gesture spotting performance of each class is also evaluated. Table IV presents average and per class tIoU of all variations using a tIoU threshold of 0.5. The high tIoU score obtained by circle and denial gestures shows that our approaches can overcome the challenging problem of dividing a long gesture into several small parts. This issue is illustrated

TABLE IV

AVERAGE AND PER-CLASS TIOU@0.5 RESULTS OF OUR APPROACH VARIATIONS

Class	HC baseline	HC simil	DN baseline	DN simil	DN trip-simil
FD	0.39	0.49	0.71	0.72	0.76
FL	0.42	0.56	0.77	0.77	0.85
FR	0.58	0.74	0.63	0.64	0.65
FU	0.11	0.34	0.34	0.37	0.39
Ci	0.39	0.50	0.62	0.67	0.78
De	0.49	0.57	0.60	0.69	0.73
Op	0.89	0.97	0.93	0.93	0.98
Re	0.85	0.89	0.88	0.88	0.99
Avg	***0.52***	***0.63***	***0.68***	***0.71***	***0.77***

	FD	FL	FR	FU	Ci	De	Op	Re
FD	71	18	3	0	2	6	0	0
FL	21	71	0	1	1	5	0	1
FR	7	3	70	10	5	5	0	0
FU	0	12	28	43	5	12	0	0
Ci	3	26	0	3	58	10	0	0
De	0	3	12	3	6	76	0	0
Op	0	0	0	0	0	0	95	5
Re	0	0	1	1	0	0	25	73

(a) HC simil (acc. 69%)

	FD	FL	FR	FU	Ci	De	Op	Re
FD	90	9	0	1	0	0	0	0
FL	3	95	0	0	2	0	0	0
FR	2	3	67	27	1	0	0	0
FU	3	5	10	78	3	1	0	0
Ci	13	2	7	2	67	9	0	0
De	0	0	2	2	4	92	0	0
Op	0	0	0	0	0	0	100	0
Re	0	0	0	0	0	0	6	94

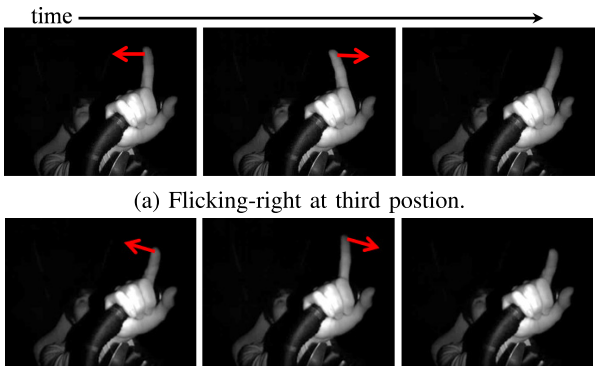
(b) DN trip-simil (acc. 84%)

Fig. 9. Confusion matrices of our both proposals (hand-crafted and deep-network).

in Figure 1, where a denial gesture could be misrecognized as two flicking-left gesture if the fourth nucleus frame is false-detected as a boundary frame.

From Table IV, we can also notice that the flicking-up class obtains the lowest results. This issue occurs mainly for two factors: 1) the appearance of each gesture frame is more similar than other gestures, as shown in Figure 8 (d); 2) the *end* boundary frame of this gesture often presents apparent differences in the finger position with respect to the *start* boundary; this issue mainly occurs when the gesture is performed continuously. On the other hand, our *DN trip-simil* approach outperforms other variations in almost all classes, except for flicking-right, which is surprisingly better recognized by our *HC simil* approach. This issue might be related to misrecognition errors of the gestures recognition process due to the lack of motion information.

Confusion matrices of our both approaches are shown in Figure 9. The average accuracy achieved by *HC simil* and *DN trip-simil* is 69% and 84%, respectively. The most relevant misrecognition errors from each approach are flicking-up → flicking-right (28%), and flicking-right → flicking-up (27%) from *HC simil* and *DN trip-simil*, respectively. These errors are produced due to the appearance similarities between these gestures, which are generated by the camera perspective bias. As illustrated in Figure 10, flicking-right and flicking-up look similar when the hand is at the farthest point from the camera (location P3). Another recurrent error of both approaches is related to the circle gesture. Since this gesture has the longest duration, this error reflects that more temporal sections are needed to define the circle gesture. Besides, this gesture clearly



(a) Flicking-right at third postion.

(b) Flicking-up at third postion.

Fig. 10. Example of a common misrecognition error from two different gestures (FR ↔ FU), which is produced by the hand perspective (located at P3). The red arrows indicate the motion direction of the finger.

TABLE V

GESTURE SPOTTING RESULTS USING DIFFERENT TIOU THRESHOLDS

Method	tIoU@0.3	tIoU@0.5	tIoU@0.8
HC simil	0.68	0.63	0.54
C3D [32]	0.77	0.69	0.48
ROAD [16]	0.74	0.70	0.57
RHGDC [22]	0.79	0.72	0.57
DN simil	0.76	0.71	0.61
DN trip-simil	0.82	0.77	0.67

presents intraclass appearance changes due to differences in finger position and motion.

C. Comparison With Previous Works

We compare our proposals with three previous approaches: C3D [32], ROAD [16], and RHGDC [22]. Real-time Hand Gesture Detection and Classification (RHGDC) is a recent approach which focuses on recognizing gestures in real-time. RHGDC employs two 3D CNN models which affect the memory storage requirements. We implemented RHGDC with the standard parameters defined in its publicly available code on Pytorch. Real-time Online Action Detection (ROAD) is one of the few state-of-the-art approaches that can perform online without comprising memory storage with huge models (such as 3DCNN-based approaches). We trained ROAD with a reduced VGG-16 network and the conventional SSD detector, by using their publicly available code on Pytorch. The results of these methods are compared with our proposals in Table V. Although QOM [40] is commonly applied in the state-of-the-art approaches of the ChaLearn LAP ConGD challenge [30], [39], [41], we did not include it in this comparison because we experimentally got poor results when using optical flow from IR images instead of depth images.

The benefit of our proposals can be easily seen when using tIoU@0.8. The results of *DN trip-simil* and *DN simil* outperform all previous works, showing that the similarity check process helps to precisely determine the boundaries of continuous gestures. *HC simil* achieves competitive results compared with those of deep learning methods, and it is even better than C3D when precise detections of boundaries are required (tIoU@0.8).

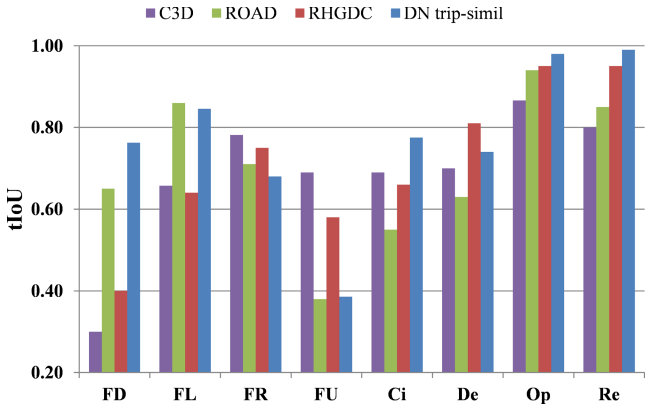


Fig. 11. Comparison of tIoU per class using a threshold of 0.5.

	FD	FL	FR	FU	Ci	De	Op	Re
FD	68	0	0	32	0	0	0	0
FL	16	57	5	16	3	3	0	0
FR	7	0	58	5	16	14	0	0
FU	21	0	0	79	0	0	0	0
Ci	26	0	0	17	57	0	0	0
De	6	24	2	2	2	64	0	0
Op	0	0	0	0	0	0	100	0
Re	0	2	0	0	0	0	0	98

(a) RHGDC (acc. 74%)

	FD	FL	FR	FU	Ci	De	Op	Re
FD	74	0	0	22	0	0	4	0
FL	0	25	2	14	39	20	0	0
FR	0	0	42	2	37	19	0	0
FU	16	0	0	84	0	0	0	0
Ci	0	0	10	10	75	5	0	0
De	0	0	1	4	17	78	0	0
Op	0	0	0	0	0	0	96	4
Re	0	0	0	0	2	0	0	98

(b) C3D (acc. 72%)

	FD	FL	FR	FU	Ci	De	Op	Re
FD	88	7	0	5	0	0	0	0
FL	0	85	0	0	2	13	0	0
FR	0	0	81	11	0	8	0	0
FU	6	0	9	85	0	0	0	0
Ci	0	12	12	0	76	0	0	0
De	0	13	3	6	0	78	0	0
Op	0	0	0	0	0	0	100	0
Re	0	0	0	0	0	0	11	89

(c) ROAD (acc. 86%)

Fig. 12. Confusion matrices of previous works.

Figure 11 shows a comparison of per class gesture spotting performance between *DN trip-simil* and all previous deep learning approaches. The results of flicking-up suggest that the motion information learned by 3D CNN models benefits the gesture spotting, since the hand motion of this class is distinctive even if the shape is very similar to boundary frames. That is why *DN trip-simil* and ROAD have problems to spot this class, since they only learn appearance features with 2D CNN models. On the other hand, our proposal significantly overcomes the results of flicking-down gestures. 3D CNN based approaches have several problems spotting this class because flicking-down gestures share motion and appearance similarities with the retraction state, as shown in Figure 1.

Confusion matrices of previous works are shown in Figure 12. Flicking-up and flicking-right results from C3D and RHGDC indicate that the misrecognition errors presented by *DN trip-simil* can be solved by including motion information, which is the distinctive feature among these classes (as illustrated in Figure 10). Moreover, a critical issue of these methods is related to the fixed-length input of 3D CNN models. Gestures shorter than the required input size are forced to loop frames as many times as necessary to satisfy fixed-length. Thus, the misrecognition errors of flicking-down ↔ flicking-up might be produced due to the described problem combined with the imprecise spotting, since they may include frames from preparation or retraction states. ROAD does not present these issues because it does not employ 3D CNN models. However, denial gesture recognition is affected by the frame-wise classification, due to the hand shape look similar to different gestures.

TABLE VI
COMPARISON OF GESTURE RECOGNITION RESULTS

Method	Recall	Precision	f1-score	Accuracy
HC simil	0.71	0.30	0.43	0.69
C3D [32]	0.74	0.15	0.25	0.72
RHGDC [22]	0.68	0.17	0.27	0.74
ROAD [16]	0.85	0.31	0.46	0.86
DN trip-simil	0.90	0.55	0.68	0.84

ROAD presents the highest gesture recognition accuracy (86%). However, as shown in Table VI, *DN trip-simil* presents better performance, due to the imbalance on the amount of instances per class. Our proposal presents the best results of recall, precision, and f1-score. Indeed, *DN trip-simil* overcomes more than 20% the precision and f1-score of ROAD. These results suggest that our gesture recognition proposal is more effective than previous works even when not perfect gesture spotting is achieved (as evaluated by tIoU@0.5).

In general, the experimental results confirm that our proposals have certain robustness to scale and orientation changes. In the case of our DN approach, the random crop augmentation and the training data with different hand positions may alleviate these problems. However, due to the lack of explicitly motion information, an orientation issue is still present, as illustrated in Figure 10. On the other hand, the different RF models trained based on the three hand positions may alleviate the related problems of our HC approach.

D. Testing Time Evaluation

To demonstrate the real-time capability of our both proposals, we present detection speeds of each involved process in Table VII and Table VIII, respectively. *HC simil* can run at 28 fps, even if all processes are sequentially run on CPU. Note that the bottleneck of the performance relies on the optical flow computation. Thus, we can further speed-up this approach by choosing a more efficient optical flow process. *DN trip-simil* method runs at 53 fps on a single GTX 1080 GPU. In this case, the SSD lite process presents the bottleneck, which can be improved by any faster object detector approach.

Finally, Table IX presents a comparison of all evaluated works based on the inference time (speed) and the model size in terms of storage memory consumption. We can notice that the fastest method is *HC baseline* (running on CPU), but it presents deficiencies on gesture spotting and recognition (showing the lowest tIoU score). Conversely, *HC simil* approach can perform at 28 fps, and higher accuracy

TABLE VII
COMPUTATIONAL TIME OF EACH PROCESS INVOLVED IN HC SIMIL

Process	Time
Optical flow	12 ms
Hand detection	3 ms
Features (HOG and HoOF)	4 ms
RF (HOG and HoOF)	8 ms
Gesture spotting (Algorithm 1)	4 ms
RF (gesture classification)	5 ms
Total	*36 ms (28 fps)*

TABLE VIII
COMPUTATIONAL TIME OF EACH PROCESS INVOLVED IN DN TRIP-SIMIL

Process	Time
SSD lite	13 ms
Triplet-loss network	1 ms
Gesture spotting (Algorithm 1)	4 ms
Gesture classification	1 ms
Total	*19 ms (53 fps)*

TABLE IX
COMPARISON OF INFERENCE TIME AND MODEL SIZE

Method	Speed	Model Size	tIoU@0.5
HC baseline	32 ms (31 fps)*	27 MB**	0.52
HC simil	36 ms (28 fps)*	27 MB**	0.63
DN baseline	15 ms (67 fps)	15 MB	0.68
DN simil	17 ms (59 fps)	15 MB	0.71
DN trip-simil	19 ms (53 fps)	19.2 MB	0.77
ROAD [16]	25 ms (40 fps)	95.1 MB	0.70
C3D [32]	37 ms (27 fps)	387 MB	0.69
RHGDC [22]	46 ms (22 fps)	370 MB***	0.72

* running on CPU

** including all RF models and PCA matrices

*** including detection and classification models

(more than 10% higher than the baseline). On the other hand, our deep-network approaches present the smallest model size, while being significantly faster than previous works. In resume, our proposals present the most efficient performance taking into account the tradeoffs between inference time and accuracy.

VIII. CONCLUSION

In this paper, we have proposed a continuous finger gesture spotting and recognition method intended to control in-car devices. Gesture spotting was achieved by taking advantage of the boundary similarities of target gestures. Gesture recognition was based on a temporal normalization of features extracted from the set of spotted frames. We have presented two proposals capable of performing on real-time based on hand-crafted and deep-learned features. In the experiments, we assess our two approaches for recognizing driver's finger gestures, and demonstrate that, both methods obtain better performance than previous works. As future work, we plan to implement an end to end optimization, and to include specific temporal features for our deep-network approach. We also would like to evaluate our approach in the presence of natural hand movements that may look like target gestures, as well as in extreme illumination conditions, such as night driving in a real-world environment.

REFERENCES

[1] M. Alpern and K. Minardo, "Developing a car gesture interface for use as a secondary task," in *Proc. CHI Extended Abstr. Human Factors Comput. Syst. (CHI)*, 2003, pp. 932–933.

[2] F. Parada-Loira, E. Gonzalez-Agulla, and J. L. Alba-Castro, "Hand gestures to control infotainment equipment in cars," in *Proc. IEEE Intell. Vehicles Symp. Proc.*, Jun. 2014, pp. 1–6.

[3] E. Ohn-Bar and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2368–2377, Dec. 2014.

[4] C. A. Pickering, K. J. Burnham, and M. J. Richardson, "A research study of hand gesture recognition technologies and applications for human vehicle interaction," in *Proc. 3rd Inst. Eng. Technol. Conf. Automot. Electron.*, Jun. 2007, pp. 1–15.

[5] G. Jahn, J. F. Krems, and C. Gelau, "Skill acquisition while operating in-vehicle information systems: Interface design determines the level of safety-relevant distractions," *Hum. Factors: J. Hum. Factors Ergonom. Soc.*, vol. 51, no. 2, pp. 136–151, Apr. 2009.

[6] A. Rangesh, E. Ohn-Bar, and M. M. Trivedi, "Long-term multi-cue tracking of hands in vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1483–1492, May 2016.

[7] A. Rangesh, E. Ohn-Bar, and M. M. Trivedi, "Driver hand localization and grasp analysis: A vision-based real-time approach," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 2545–2550.

[8] A. Kendon, "Gesticulation and speech: Two aspects of the process of utterance," *Relationship Verbal Nonverbal Commun.*, vol. 25, no. 1980, pp. 207–227, 1980.

[9] A. Kenndon, "Current issues in the study of gesture," in *The Biological Foundations of Gestures: Motor and Semiotic Aspects*, J. L. Nespoulous, P. Perron, and A. R. Lecours, Eds. Hillsday, NJ, USA: Lawrence Erlbaum Associates, 1986, pp. 23–47.

[10] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2005, pp. 886–893.

[12] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal, "Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1932–1939.

[13] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[14] National Traffic Law Center, "Investigation and prosecution of distracted driving cases," Nat. Traffic Law Center, NHTSA, Washington, DC, USA, Rep. no. DOT-HS812407, 2017.

[15] G. Benitez-Garcia, M. Haris, Y. Tsuda, and N. Ukita, "Similar finger gesture recognition using triplet-loss networks," in *Proc. 16th Int. Conf. Mach. Vis. Appl. (MVA)*, May 2019, pp. 1–6.

[16] G. Singh, S. Saha, M. Sapienza, P. Torr, and F. Cuzzolin, "Online real-time multiple spatiotemporal action localisation and prediction," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3657–3666.

[17] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid, "Action tubelet detector for spatio-temporal action localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4415–4423.

[18] M. Zolfaghari, K. Singh, and T. Brox, "ECO: Efficient convolutional network for online video understanding," in *Proc. 15th Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 713–730.

[19] O. Kopuklu, N. Kose, and G. Rigoll, "Motion fused frames: Data level fusion strategy for hand gesture recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018.

[20] A. Joshi, C. Monnier, M. Betke, and S. Sclaroff, "Comparing random forest approaches to segmenting and classifying gestures," *Image Vis. Comput.*, vol. 58, pp. 86–95, Feb. 2017.

[21] G. Zhu, L. Zhang, P. Shen, J. Song, S. A. A. Shah, and M. Bennamoun, "Continuous gesture segmentation and recognition using 3DCNN and convolutional LSTM," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 1011–1021, Apr. 2019.

[22] O. Kopuklu, A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," in *Proc. 14th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, May 2019.
[23] E. Tsironi, P. Barros, C. Weber, and S. Wermter, "An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition," *Neurocomputing*, vol. 268, pp. 76–86, Dec. 2017.
[24] Q. Miao *et al.*, "Multimodal gesture recognition based on the ResC3D network," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 3047–3055.
[25] P. Narayana, J. R. Beveridge, and B. A. Draper, "Gesture recognition: Focus on the hands," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5235–5244.
[26] Z. Hu, Y. Hu, J. Liu, B. Wu, D. Han, and T. Kurfess, "3D separable convolutional neural network for dynamic hand gesture recognition," *Neurocomputing*, vol. 318, pp. 151–161, Nov. 2018.
[27] P. K. Pisharady and M. Saerbeck, "Recent methods and databases in vision-based hand gesture recognition: A review," *Comput. Vis. Image Understand.*, vol. 141, pp. 152–165, Dec. 2015.
[28] M. Asadi-Aghbolaghi *et al.*, "Deep learning for action and gesture recognition in image sequences: A survey," in *Gesture Recognition*. Cham, Switzerland: Springer, 2017, pp. 539–578.
[29] G. Borghi, E. Frigieri, R. Vezzani, and R. Cucchiara, "Hands on the wheel: A dataset for driver hand detection and tracking," in *Proc. 13th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, May 2018, pp. 564–570.
[30] J. Wan *et al.*, "Results and analysis of ChaLearn LAP multi-modal isolated and continuous gesture recognition, and real versus fake expressed emotions challenges," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 3189–3197.
[31] J. Wan, S. Z. Li, Y. Zhao, S. Zhou, I. Guyon, and S. Escalera, "ChaLearn looking at people RGB-D isolated and continuous datasets for gesture recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016.
[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4489–4497.
[33] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles, "SST: Single-stream temporal action proposals," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6373–6382.
[34] S. Buch, V. Escorcia, B. Ghanem, and J. C. Niebles, "End-to-end, single-stream temporal action detection in untrimmed videos," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–12.
[35] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 21–37.
[36] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Multi-sensor system for driver's hand-gesture recognition," in *Proc. 11th IEEE Int. Conf. Workshops Autom. Face Gesture Recognit. (FG)*, May 2015, pp. 1–8.
[37] P. Wang, W. Li, S. Liu, Z. Gao, C. Tang, and P. Ogunbona, "Large-scale isolated gesture recognition using convolutional neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 7–12.
[38] H. Wang, P. Wang, Z. Song, and W. Li, "Large-scale multimodal gesture segmentation and recognition based on convolutional neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 3138–3146.
[39] P. Wang, W. Li, Z. Gao, C. Tang, and P. O. Ogunbona, "Depth pooling based large-scale 3-D action recognition with convolutional neural networks," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1051–1061, May 2018.
[40] F. Jiang, S. Zhang, S. Wu, Y. Gao, and D. Zhao, "Multi-layered gesture recognition with Kinect," *J. Mach. Learn. Res.*, vol. 16, pp. 227–254, Feb. 2015.
[41] Z. Liu, X. Chai, Z. Liu, and X. Chen, "Continuous gesture recognition with hand-oriented spatiotemporal feature," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 3056–3064.
[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[43] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4207–4215.
[44] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast optical flow using dense inverse search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 471–488.

Gibran Benitez-Garcia (Member, IEEE) received the B.S. and M.S. degrees from the Mechanical Engineering School, National Polytechnic Institute, Mexico City, in 2011 and 2014, respectively, and the Ph.D. degree from the University of Electro-Communications, Japan, in 2017. He is currently a Post-Doctoral Fellow with the Intelligent Information Media Laboratory, Toyota Technological Institute, Nagoya, Japan. His main research interests include face/facial expression recognition, and human–computer interaction.

Muhammad Haris received the S.Kom. (Bachelor of Computer Science) degree from the Faculty of Computer Science, University of Indonesia, Depok, Indonesia, in 2009, and the M.Eng. and Dr. Eng. degrees from the Department of Intelligent Interaction Technologies, University of Tsukuba, Japan, in 2014 and 2017, respectively, under the supervision of Dr. H. Nobuhara. He is currently a Post-Doctoral Fellow with the Intelligent Information Media Laboratory, Toyota Technological Institute, with Prof. N. Ukita. His main research interests include low-level vision and image/video processing.

Yoshiyuki Tsuda received the B.E. and M.S. degrees in informatics from Kyoto University, Japan, in 2007 and 2009, respectively. He is currently with DENSO Corporation. His main research interests include automotive human–machine interface and image enhancement.

Norimichi Ukita (Member, IEEE) received the B.E. and M.E. degrees in information engineering from Okayama University, Japan, in 1996 and 1998, respectively, and the Ph.D. degree in informatics from Kyoto University, Japan, in 2001. He was a Research Scientist of Precursory Research for Embryonic Science and Technology, Japan Science and Technology Agency (JST), from 2002 to 2006. He was a Visiting Research Scientist at Carnegie Mellon University from 2007 to 2009. After working for five years as an Assistant Professor at NAIST, he became an Associate Professor in 2007 and moved to the Toyota Technological Institute, Japan (TTIJ), in 2016. He is currently a Professor with the Graduate School of Engineering, TTI-J. He is also with the Cybermedia Center, Osaka University, as a Guest Professor. His main research interests include object detection/tracking and human pose/shape estimation.