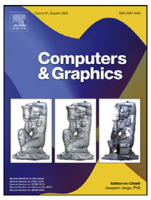
Computers & Graphics 91 (2020) 232–242

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag



Special Section on 3DOR 2020

SFINGE 3D: A novel benchmark for online detection and recognition of heterogeneous hand gestures from 3D fingers' trajectories

Ariel Caputo[a,*], Andrea Giachetti[a], Franca Giannini[b], Katia Lupinetti[b], Marina Monti[b], Marco Pegoraro[a], Andrea Ranieri[b]

[a] *Department of Computer Science, University of Verona, Strada le Grazie 15, Verona 37134, Italy*
[b] *Institute of Applied Mathematics and Information Technologies Enrico Magenes National Research Council Genoa Branch, Via De Marini 6, Genova 16149, Italy*

A R T I C L E I N F O

Article history:
Received 13 May 2020
Revised 21 July 2020
Accepted 27 July 2020
Available online 7 August 2020

Keywords:
Computers and graphics
Formatting
Guidelines

A B S T R A C T

In recent years gesture recognition has become an increasingly interesting topic for both research and industry. While interaction with a device through a gestural interface is a promising idea in several applications especially in the industrial field, some of the issues related to the task are still considered a challenge. In the scientific literature, a relevant amount of work has been recently presented on the problem of detecting and classifying gestures from 3D hands' joints trajectories that can be captured by cheap devices installed on head-mounted displays and desktop computers. The methods proposed so far can achieve very good results on benchmarks requiring the offline supervised classification of segmented gestures of a particular kind but are not usually tested on the more realistic task of finding gestures execution within a continuous hand tracking session.

In this paper, we present a novel benchmark, SFINGE 3D, aimed at evaluating online gesture detection and recognition. The dataset is composed of a dictionary of 13 segmented gestures used as a training set and 72 trajectories each containing 3–5 of the 13 gestures, performed in continuous tracking, padded with random hand movements acting as noise. The presented dataset, captured with a head-mounted Leap Motion device, is particularly suitable to evaluate gesture detection methods in a realistic use-case scenario, as it allows the analysis of online detection performance on heterogeneous gestures, characterized by static hand pose, global hand motions, and finger articulation.

We exploited SFINGE 3D to compare two different approaches for the online detection and classification, one based on visual rendering and Convolutional Neural Networks and the other based on geometry-based handcrafted features and dissimilarity-based classifiers. We discuss the results, analyzing strengths and weaknesses of the methods, and deriving useful hints for their improvement.

1. Introduction

Gesture recognition techniques are expected to play an important role in future interactive experiences. In many Virtual and Augmented Reality applications or in the control of robots and smart objects, gestural interaction could provide a natural way to interact. Low-cost hand tracking tools like the Leap Motion can be mounted on the headset or embedded in desktop interfaces, and software pipelines able to provide hand pose estimation from RGB and RGBD data are available [1,2]. For this reason, hand gesture recognition can be treated as a 3D shape analysis method and has become an interesting topic for the geometry processing community.

Several researchers tried to develop gesture-recognition methods based on the analysis of the finger joints trajectories and a few benchmarks to compare the performance of these methods have been released [3–8]. The major limitations of most of the existing benchmarks are related to the fact that they measure the performance of classifiers working on segmented trajectories and do not test the methods in a realistic online detection scenario. The difference is relevant, as, in principle, the methods should be able to process streams of skeletal data continuously also avoiding false detection and being robust against different types of "non-gesture" movements not included in training data (noise).

An example of unsegmented online detection is proposed in [9], where the dataset does not include static gestures or typical

* Corresponding author.
E-mail addresses: cptfmr71@univr.it (A. Caputo), andrea.giachetti@univr.it (A. Giachetti), franca.giannini@ge.imati.cnr.it (F. Giannini), katia.lupinetti@ge.imati.cnr.it (K. Lupinetti), marina.monti@ge.imati.cnr.it (M. Monti), marco.pegoraro_01@studenti.univr.it (M. Pegoraro), andrea.ranieri@ge.imati.cnr.it (A. Ranieri).

https://doi.org/10.1016/j.cag.2020.07.014
0097-8493/© 2020 Elsevier Ltd. All rights reserved.

Table 1
Summary of the benchmarks for 3D hand gesture recognition including finger trajectories data. Static gestures are characterized by a constant hand pose, **Coarse** by a global hand trajectory and **Fine** by fingers' articulation. **Samp.** is the total number of gestures avalilabe in the training and test sets, **Seq.** is the number of recorded sequences including single or multiple gestures and non gesture movements. **Hands** indicates the number of hands used to perform the gestures and if the gesture is always performed with the right hand (R). SFINGE 3D is the first dataset including static, fine and coarse gestures in an online detection task with dictionary-based training.

Dataset	Data included	Online task	Gesture classes			Training set		Test set		Users	Hands
			Static	Fine	Coarse	Samp.	Seq.	Samp.	Seq.		
Shrec 2017 [3]	22 Joints + Depth images	No	0	5	9	1960	NA	840	NA	28	1 (R)
DHG 14/28 [11]	22 Joints + Depth images	No	0	5	9	2800	NA	NA	NA	20	1 (R)
Handicraft [12]	22 Joints	No	0	9	1	210	NA	90	NA	10	2
ASL [4]	Joints, Angles	No	18	12	0	780	NA	420	NA	20	1 (R)
Montalbano [13]	Audio, RGBD images, 3D skeleton	Yes	0	0	20	7754	393	2742	276	27	2
LMDHG [9]	46 Joints	Yes	0	0	13	~455	35	~182	14	21	2
LMHGIf3DVMI [14]	23 Joint	Yes	0	5	2	NA	~881	NA	~98	10	1 (R)
Shrec 2019 [10]	16 Joints + Quaternions	Yes	0	0	5	60	60	135	135	13	1
SFINGE 3D	20 Joints + Quaternions	Yes	5	4	4	468	NA	278	72	5	1

single-handed manipulation gestures and the evaluation protocol does not measure false positives. Another one is the SHREC 2019 online gesture detection track [10], where the gesture classes to be detected are limited to coarse ones, characterized by a single trajectory.

This fact motivated us to propose a novel benchmark especially designed for the testing of gesture classifiers that specifically address a realistic online scenario. This goal is achieved by providing in each sequence of the dataset an unsegmented collection of gestures and adopting a specific evaluation protocol for online detection. In addition, to make the dataset the more general as possible we include different kinds of gesture templates (i.e. static, fine and course) in an equal proportion. We also present two different approaches to cope with the proposed task, one based on image rendering and Convolutional Neural Networks, the other exploiting a dissimilarity based approach and handcrafted features. The results of these methods are promising, show the feasibility of a trajectory-based online recognizer able to detect different kinds of gestures, but also demonstrate the necessity of further investigation to make the methods more robust.

The remaining of this paper is organized as follow: Section 2 provides an overview on the existing 3D hand dataset suitable for online gesture recognition; Section 3 illustrates the proposed collection of data and how it was acquired; while Section 4 describes two methods applied on the proposed dataset. Section 5 shows how to use the proposed dataset to evaluate and compare online 3D hand gesture recognition methods and Section 6 discusses the obtained results. Finally, Section 7 concludes the paper.

2. Related work

Several works addressing online hand gesture recognition propose their own dataset (e.g. [15–17]), which not always are public and in some cases have a small number of samples. Anyhow, to evaluate and compare the effectiveness of online 3D hand gesture recognition algorithms, it is essential investing in community benchmarks. For this reason, in the following, we focus our attention on reviewing benchmarks for 3D hand gesture recognition including fingers trajectories data that is available to the research community. A summary and comparison of the discussed methods is proposed in Table 1.

So far, one of the most popular benchmarks is the one from the SHREC 2017 track on 3D Hand Gesture Recognition [3], which is derived from the DHG 14-28 dataset [11]. Both these datasets feature a dictionary of 14 gestures performed in two ways (using just two fingers or the whole hand). The dictionary includes both coarse (characterized by a global hand trajectory) and fine (characterized by fingers' articulation) gestures, without any static gestures. The task proposed is the classification of segmented gestures from the temporal evolution of a 22 joints hand skeleton. Wei et al. [12] proposed the Handicraft-Gesture dataset, which has been used in various works. It includes 300 sequences of depth data corresponding to 10 gestures, each one performed three times by 10 subjects. The considered gestures originate from pottery skills and are: *Poke*, *Pinch*, *Pull*, *Scrape*, *Slap*, *Press*, *Cut*, *Circle*, *Key Tap*, and *Mow*. Among them, 8 include either finger's position or movement detection, and only one movement involves two hands. The depth data is captured with 60 frames per second. For the evaluation the authors used the dataset derived by 7 subjects and the data of the other 3 subjects for the testing, such that no gesture was already present in the training set. Therefore this dataset includes only the dictionary of gestures. Avola et al. [4] acquired a dataset of 12 dynamic gestures and 18 static gestures based on the ASL in order to evaluate the behavior of their proposed method. The gestures were chosen to represent much of the variations in joint angles and finger positions that occur when the hand performs a gesture. The dataset is composed of 1200 sequences, coming from 20 different people (15 males and 5 females). The sequences from 14 people were used to create the training set while sequences of the remaining 6 people were used to form the test set. Unfortunately, the aforementioned datasets are not ideal for evaluating online classification tasks since the collected data is segmented.

To support the evaluation and comparison of 3D hand gesture recognition methods performing online, Caputo et al. provide a set of 60 trajectories annotated with temporal locations in the SHREC 2019 [10]. The task of the dataset is to detect the gesture label and its end on a test set of 135 unlabelled trajectories with an online method. The data were collected from 13 users with a Leap Motion for a total of 195 sequences. For each sequence, users had to perform one and only one of the following gestures with their index fingertip: *Cross*, *Circle*, *V-mark*, *Caret*, *Square*. The dataset includes the 3D coordinates and quaternion of 22 joints of a hand. Focusing on the recognition of gestures linked to a vocabulary, Montalbano gesture dataset [13] combines 13,858 Italian sign gestures in video sequences, considering 20 different classes. Gestures were performed by 27 different participants and data were acquired through a Kinect 360 sensor with multi-modal information present, e.g. RGB, depth, skeleton and audio. The dataset contains 393 sequences for the training set (7.754 gestures), 287 validation sequences (3.362 gestures), and 276 test sequences (2.742 gestures). Each sequence lasts between 1 and 2 minutes and contains between 8 and 20 gesture samples. Specializing on a domain-specific set of gestures, the Leap Motion Hand Gestures for Interaction with 3D Virtual Music Instruments (LMHGIf3DVMI) dataset [14] has been designed specifically to recognize gestures for the control of the performance of a virtual 3D musical instrument. With this aim, the dataset considers eight gesture classes (*Palm*

Table 2
The set of 13 gestures of the proposed dictionary, including static gestures, dynamic gestures characterized by palm trajectory only (coarse) and dynamic gestures characterized by fingers' articulation (fine).

#	Name	Type	Description
1	One	Static	Thumb is raised while all the other fingers are closed
2	Two	Static	The index and the middle finger are raised, other fingers are closed
3	Three	Static	Index, middle and thumb raised, other fingers closed
4	Four	Static	Thumb closed and others fingers raised
5	OK	Static	All the fingers are raised, while the index and thumb are closed in order to shape a circle
6	Pinch	Fine	The index touches the thumb while the others fingers are closed
7	Grab	Fine	All the fingers close together like if they were holding an object
8	Expand	Fine	All the fingers raise together
9	Tap	Fine	The index moves like if it was pushing a button
10	Swipe left	Coarse	The hand moves to left
11	Swipe right	Coarse	The hand moves to right
12	Swipe V	Coarse	The hand draw a letter v
13	Swipe O	Coarse	The hand draw a letter o

tapping, *Index finger tapping*, *Thumb finger plucking*, *Index finger plucking*, *Middle finger plucking*, *Ring finger plucking*, *Pinky finger plucking* and *Unknown*). Data was collected from 10 participants (5 females and 5 males) using the LM sensor for the eight gesture classes (of the right hand) and is made by a set of 186 features including (i) 3D positions of all finger joints, palm center, wrist and elbow; (ii) 3D velocity vectors for palm center and finger tips; (iii) 3D vectors of directions for hand and each finger, along with the 3D vector of the palm normal. Finally, among the available benchmarks suitable for online purposes, the Leap Motion Dynamic Hand Gesture (LMDHG) database [9] contains unsegmented sequences of hand gestures performed with either one hand (*Point to*, *Catch*, *Shake down*, *Shake*, *Draw C*, *Scroll*, *Draw Line*, *Slice*, *Rotate*) or both hands (*Shake with two hands*, *Catch with two hands*, *Point to with two hands*, *Zoom*). Data was acquired through a Leap Motion sensor encoding the 3D coordinates of 46 joint positions (23 for each hand) and involving 21 participants. At the end of each gesture, the involved participants were asked to perform a “rest” gesture, i.e. keeping the hands as still as possible in a flat position for a few seconds, thus providing a kind of null gesture that can be used to recognize the ending of a certain movement. This dataset contains 608 “active” plus 526 “inactive” (i.e. classified as resting) gesture samples, corresponding to a total of 1134 gestures. The main limitation of this dataset is, on one hand, that the training gestures, being performed by many different participants, are quite noisy and often misinterpreted and wrongly performed by the participants, in particular with respect to the scant amount of samples provided. On the other hand, each gesture is interleaved with a *Resting* gesture, making the online recognition task less realistic. Moreover, most classes have roughly 50 samples, except *Point to with hand raised* that presents only 24 samples and *Rest*, as previously said, that presents 526 samples.

Looking at the related work, it is possible to see that the benchmarks proposed are mostly used to test methods for classification of segmented gesture. Most of the methods proposed in the literature and focused on gesture recognition from fingers trajectories are evaluated on this kind of benchmarks, especially on the Shrec 2017 dataset (e.g. [4,6,18]). However, the methods cannot be applied for online gesture recognition, as they are not trained to discriminate gestures from non-gesture sequences. The few benchmarks designed also for online gesture detection are not very popular and presents some limitations, not including different gesture types.

This motivated us to create our novel benchmark, SFINGE 3D, with a dictionary of gestures that are typical of mid-air interactions, including static and dynamic (both fine and coarse) gestures, and featuring a specifically designed online detection evaluation procedure.

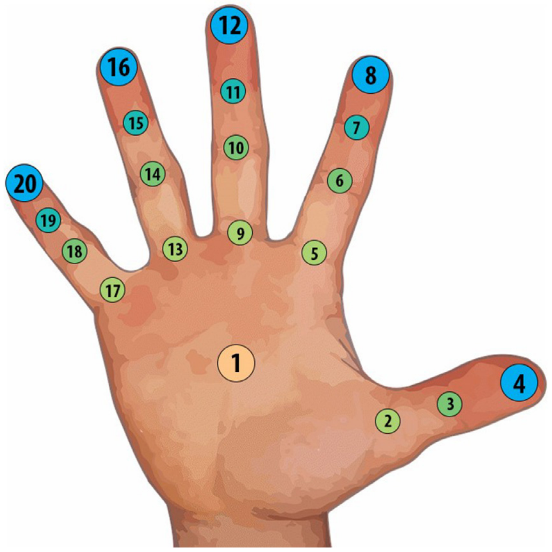


Fig. 1. Finger joints captured in the data stream.

3. Dataset and proposed task

Data was captured through a Leap Motion sensor mounted on the head of the performing subjects, connected to a PC running a simple Unity application. The 3D position in the space and the quaternion associated with the spatial rotation of the palm and of 19 finger joints, shown in Fig. 1, are captured at the rate of 20 frames per second and stored in text files using the Leap Motion API. Users had to perform gestures with the right hand, keeping the palm in view.

For our task, we recorded separately a dictionary with the template of the segmented gestures to be detected and a set of sequences with the gestures to be found. The gestures have been performed by 5 subjects after a short training. Each sequence was manually annotated during recording. The start was set when users start performing a gesture and the end when users decided to end the execution. The dictionary is composed of 468 gestures, 36 executions for each of the 13 classes defined. The classes are described in Table 2 and depicted in Fig. 2.

The gesture classes can be divided in three macro categories: static, coarse and fine. In the static gestures the hand reaches a meaningful pose and remains in that pose for at least one second. Typical examples, included in our collection, are numbers performed by raising selected fingers. In the coarse gestures, the whole hand trajectory characterizes the classes. The movement can

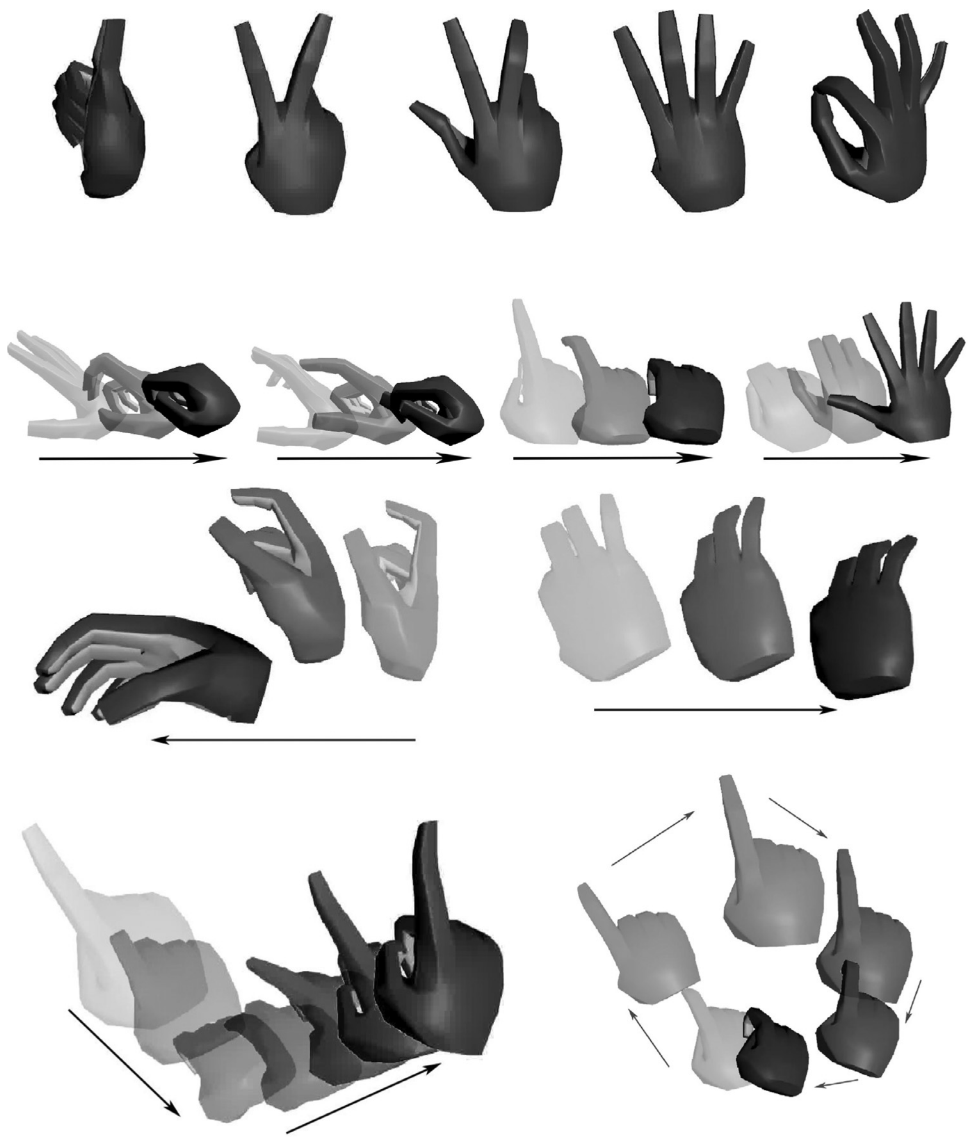


Fig. 2. Examples of the gestures included in the dataset, rendered based on the captured skeletons. First row: static gestures (1, 2, 3, 4, ok). Second row: fine gestures (grab, pinch, tap, expand). Third and fourth rows: coarse gestures (swipe left, swipe right, swipe V, swipe O).

be simple like from right to left or represent a symbol written in the air like the letter “O”. Finally, in the fine gestures, the action is determined by the fingers’ movements. For example, in the grab class, the fingers close together as if they were holding an object. With these three macro-categories, the dataset includes examples of different actions that may be useful to detect and recognize in interactive interfaces.

Test data are 72 sequences including 3 to 5 gestures belonging to the classes of the dictionary possibly interleaved with random finger motion. The task is to recognize each gesture in a sequence given the dictionary of classes that can be performed. The algorithms we want to benchmark should be able to detect the hidden gestures with a limited delay, indicating the starting frame and avoiding false detection. Not having a non-gesture class, the task can also be seen as the training of a set of one class classifiers aimed to separate the gestures belonging to each single class from the non-relevant gestures.

We designed a specific evaluation estimating the number of gestures correctly detected, the number of the gestures missed, the number of false positives and the resulting F_{score} [19] defined as:

$$F_{score} = 2\frac{precision \cdot recall}{precision + recall} \tag{1}$$

where *precision* is the percentage of gestures correctly detected in the sequences and *recall* is percentage of relevant gestures in the sequences correctly classified.

A gesture is considered as *correctly detected* if (i) the gesture start is identified within the time window of the ground truth gesture with a margin of 20 frames at the beginning to allow slightly early detections to still be considered correct[1] and (ii) the gesture is labelled in accordance with the ground truth. If a gesture is detected in the proper frame window but with a different label, then we mark the gesture as *mislabelled* and consider it as a false positive, like gestures detected outside the correct gesture frames. When a gesture is not detected in the corresponding time window, we label it as *missed* (i.e. false negative).

[1] Especially in the case in which the algorithm object of the benchmark needs a wide time window and therefore can only attribute the beginning of the current detection to a $t - \tau_{win}$ time.

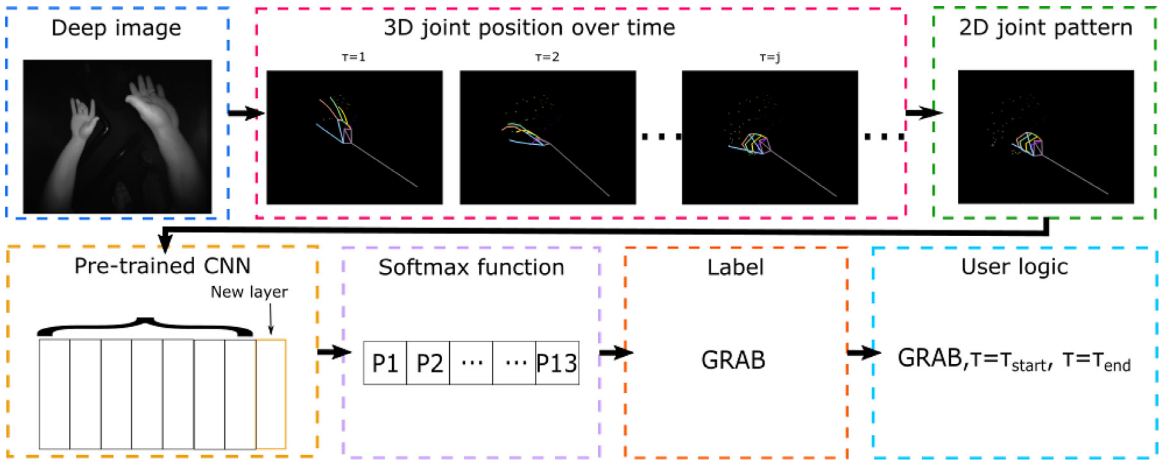


Fig. 3. Pipeline of the view-based hand gesture recognition.

The dataset with the evaluation code are publicly available at this website: https://github.com/SFINGE3D/DatasetV1

4. Proposed methods

The proposed SFINGE 3D bemchmark has been used to evaluate and compare two methods for the online gesture recognition. The first method is derived from a recent view-based approach exploiting Convolutional Neural Networks and adapted for online recognition (see Section 4.1). The other is based on the idea of training single class classifiers based on dissimilarity features evaluated on the provided dictionary (see Section 4.2).

4.1. View-based method

The first method applied on the SFINGE 3D benchmark adopts and improves the view-based approach presented in [8], which is able to detect gesture movements over time and whose pipeline is illustrated in Fig. 3. First, hand gesture data is recorded by using a Leap Motion sensor (blue box). A 3D gesture visualization containing temporal information is created by using the joint positions of the 3D skeleton of the hands (magenta box) as obtained from the Leap Motion data. Then, from the 3D visualizer, a 2D image is generated by projecting the obtained 3D points on a view plane (green box). The created image represents the input of the pre-trained Convolutional Neural Network (CNN) (yellow box), resulting in a single softmax layer that directly generates the probability distribution for the reference classes (purple box). The gesture is then labeled with the class that obtains the maximum probability value (orange box). Finally, the “user logic” block validates the label only if its probability is above a certain percentage of confidence (98% for the SFINGE 3D dataset) and it concatenates consecutive gestures with same labels in a single gesture, also determining the starting and ending frames (azure box).

4.1.1. From SFINGE 3D data to 3D joint visualization

The method proposed in [8] has been developed on the 3D hand skeleton with 46 joints (23 joints for each hand) created by exploiting the positions of each finger of the hand (20 joints per hand), the palm center (1 joint per hand), the wrist (1 joint per hand) and the elbow (1 joint per hand). To apply the method proposed in [8] on the SFINGE 3D dataset, which encodes 20 joints out of the 23 necessary for the 3D hand skeleton visualization, it is necessary to first establish the positions of the missing joints, i.e. the *metacarpal thumb*, the *wrist* and *elbow*. Note that the method proposed in [8] uses just the 3D positions of the joints without considering their orientation, then here the method exploits just a subset of the data present in the SFINGE 3D dataset.

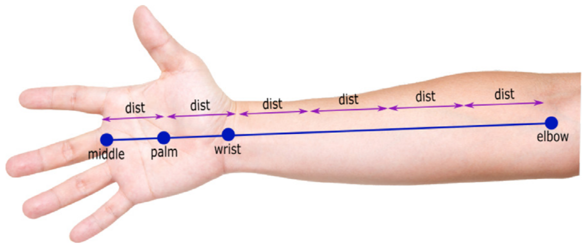


Fig. 4. Missing joints reconstruction based on average arm vs. hand proportions.

These joint positions are approximated assuming that the elbow, the wrist, the palm and the middle intermediate joints are aligned. In addition, we assume that the wrist, the palm and the middle intermediate joints are equally distant (Fig. 4); while the elbow and the wrist joints double the wrist and the middle intermediate joints distance. Then, with these assumptions and defining the distance between the palm and the middle intermediate joints as *palm_middle_distance*, we have defined the missing joints positions as follows:

$$\text{elbow} = \text{palm} - 4 \cdot \text{palm\_middle\_distance} \tag{2}$$

$$\text{wrist} = \text{palm} - \text{palm\_middle\_distance} \tag{3}$$

Finally, for simplicity, at this stage we set the last lacking joint, i.e. the *metacarpal thumb*, equal to the *intermediate thumb*.

Once all the necessary joint positions are computed, they are used to visualize a 3D skeleton into a programmable 3D environment obtained by using the VisPy library [20].

Spatial and temporal information of each gesture movement are encoded by creating a 3D gesture image, where 3D points and edges are depicted in the virtual space for each finger. The temporal information is encoded by changing the color intensity of the joints at different time instants: recent positions have more intense colors, while earlier positions have more transparent colors.

4.1.2. From 3D joints to 2D pattern

Finally, we create a 2D image every 20 frames (1 s) by projecting the 3D points and edges as they are at the last instant of the training gesture on a view plane corresponding to the top view, which represents hands in a “natural” way as a human usually see them. Fig. 5 shows examples of the 2D patterns obtained for the

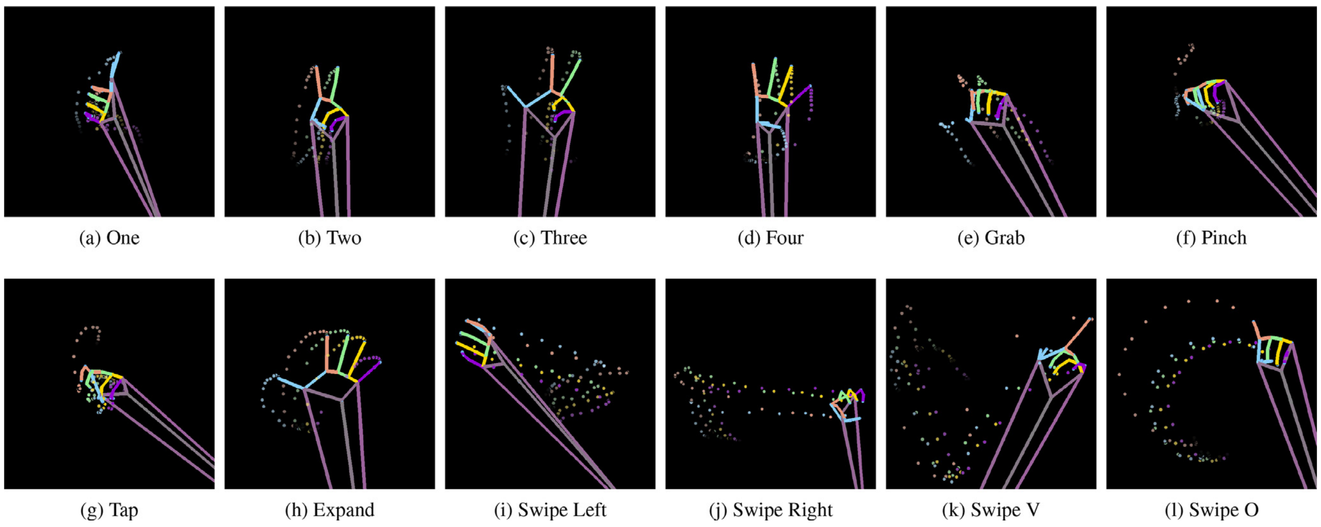


Fig. 5. Example gestures as displayed by the 3D visualizer. Fingertips traces shows that some gestures are much quicker than others (e.g *Tap* vs. *Swipe V*).

different gestures of the SFINGE 3D benchmark. Although this view does not contain all the information available in the 3D representation of the hands, we have found that it is sufficient for a CNN to classify the set of dynamic gestures under study very accurately.

4.1.3. *Offline data augmentation*

With the aim of improving the robustness of the method proposed in [8], additional offline data augmentation through the 3D visualizer was performed before feeding the network with the images.

This need arose from the fact that, in order to be able to detect gestures in a continuous flow of non-segmented hand poses, we had to drastically decrease the inference time from the 5 seconds of the initial method down to the 20 frames (1 s) used to classify the SFINGE 3D dataset. Having to classify images so frequently also forced us to vary the cancellation policies of the history of fingertips, which now no longer occurs at constant intervals but when the classification accuracy exceeds 75%.

Using these inference criteria we noticed that, using the original training method based only on the online data augmentation performed by *Fast.ai* (see Section 4.1.5), the classifier was not able to detect many short gestures in time (such as *expand*), frequently incomplete gestures (such as *swipe V* or *swipe O*) or gestures with a very long and noisy history, mainly due to a failure in detecting the previous gesture.

To deal with these issues the training set was enriched with the images obtained by sampling the gestures more frequently and adding noise directly in the 3D visualizer (Fig. 6). Gestures were sampled at constant intervals of 25 frames throughout their progress and with variable history length from 50 to 400 frames in 50 frame increments. Furthermore, we added uniform random noise on the history of the gesture by inserting a random number of points (between 50 and 400 in increments of 50) to the true history of the gesture. The noise is added on a frame by frame basis, thus it is unrelated with the gesture or with the previous or next frame in the sequence, much like noise on an analog TV broadcast correlates with the image or the previous/next frames. Uniform random noise makes fingertip traces very difficult for a classifier (or for a human being) to see. The classifier therefore becomes very good at distinguishing the noise coming from old gestures (structurally easier to eliminate than uniform random noise) from the trace of the current gesture, therefore boosting overall classification accuracy.

4.1.4. *Classification architecture*

The proposed method leverages a pre-trained ResNet-50 [21], a state-of-the-art 2D CNN that has been modified and fine-tuned to classify the images produced by the 3D visualizer. The ResNet-50 is very well suited for this kind of problem because this kind of architecture is pre-trained on ImageNet [22] and it is one of the fastest at making inference on new images, having one of the lowest FLOPS count among all the architectures available today [23]. Unfortunately, given the modest size of the original LMDHG dataset, it would not have been possible or useful to train from scratch a 3D CNN model capable of classifying all the available information coming from the LM sensor. Therefore transfer learning was performed twice, by first loading the pretrained ImageNet weights, then training on the LMDHG database and finally fine-tuning on the SFINGE 3D dataset.

4.1.5. *The training method*

The training took place using Python code inside Jupyter Notebook, leveraging the popular *Fast.ai* [24] deep learning library based on PyTorch. The hardware used was a GPU node of the high-performance EOS cluster located within the University of Pavia. Each of those GPU nodes has a dual Intel Xeon Gold 6130 processor (16 cores, 32 threads each) with 128 GB RAM and 2 Nvidia V100 GPUs with 32 GB of video RAM.

The training was performed on 1920x1080 resolution images rendered by our 3D visualizer, properly sorted in directories according to the labels of the training set of the SFINGE 3D dataset and then split randomly into training and validation sets with a 70/30 proportion.

As previously mentioned, the model used for training was a ResNet-50 architecture already pre-trained on ImageNet first and then fine-tuned on the LMDHG dataset. *Fast.ai* convenient APIs allow to download popular pre-trained architectures and weights in a very simple and automatic way. *Fast.ai* also automatically modifies the model so that the number of neurons in the output layer corresponds to the number of classes of the current problem, initializing the new layer with random weights. However, to perform transfer learning from another previously trained model on the LMDHG database, a small manual intervention was necessary to mimic what *Fast.ai* does under the hood: we then loaded the LMDHG model, erased the output layer with 16 neurons and replaced it with a new one with 13 untrained neurons to adapt it to the current problem.

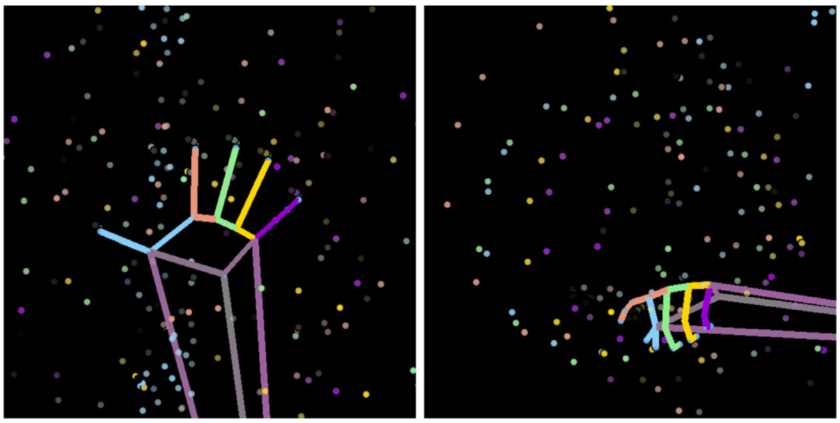


Fig. 6. Example of gestures (*Expand* and *Swipe O*) with noise added as data augmentation directly in the 3D visualizer.

The training was performed using the progressive resizing technique, i.e. performing several rounds of training using the images in the dataset at increasing resolutions to speed up the early training phases, have immediate feedback on the potential of the approach, and to make the model resistant to images at different resolutions (i.e. the model generalizes better on the problem and on hands of different sizes). The specific section in [25] explains very well the concept of progressive resizing. For our particular problem, we have chosen the resolutions of 192, 384, 576, and 960 px (i.e. 1, 2, 3, and 5 tenths of the original 1920×1080 px resolution). Raising the resolution above 1/2 of the original resolution provides no further improvement in classification accuracy, has high training costs and makes the network slower at inference time.[2] We also hooked a few callback functions through the *Fast.ai* callback APIs to automatically save the model if the accuracy on the validation set improved with respect to the previous epoch and to stop the training if the accuracy was above a predefined threshold, in this case 0.999, equal to 99.90% accuracy.

When performing transfer learning from a pre-trained network, each training round at a given image resolution is usually divided into two phases, $a =$ convolutional layers are frozen and $b =$ all layers unfrozen, each made by one or more epochs. For each epoch, all the training set is fed into the network, one batch at a time, with each image in the batch being altered by a random number of online data augmentation transformations of different intensity or magnitude, performed in real-time by *Fast.ai* on the CPU. The online data augmentation transformations that we chose to apply to the images were: crop, flip left/right, symmetric warp, rotate, zoom, brightness and contrast. For our problem, we found that there was no significant improvement in *unfreezing* the convolutional layers of the ResNet architecture, so we performed only the training of the last uninitialized layer with fewer epochs and with larger learning rates.

All the code and jupyter notebooks described in this section are available at the following URL.[3]

4.1.6. Inference on the test set

The test set of the SFINGE 3D dataset is composed of the same classes as the training set but presenting (i) a continuous stream of gestures, and (ii) noise between one gesture and the other. For these reasons, it cannot be processed with a "traditional" inference approach, since it is very difficult to establish a priori where a gesture begins and ends. In order to apply our method on the SFINGE 3D test sequences, we added a "user logic" block at the end of the pipeline proposed in [8]. This block allows to set thresholds to make the recognition more robust and to vary the amount of previous history that is fed to the model in the inference phase. The "user logic" block also performs other useful functions, such as preparing the CSV file for performance evaluation and concatenation of consecutive predictions in a single prediction with a frame interval which is the sum of the intervals of the individual predictions.

4.2. Dissimilarity-based method

The second method we applied to the SFINGE 3D dataset relies on the dissimilarity-based classification approach [26] based on binary classifiers and a sliding window approach for online detection. The idea of dissimilarity-based classification is to characterize each gesture with a set of distances from selected dictionary elements. This method has shown to be successful in different domains, improving the classification performance when the dissimilarities do not separate classes well.

The vector including all these dissimilarity measurements is taken as the gesture descriptor, and descriptors of the dictionary elements have been used to train 13 single-class classifiers able to discriminate each of the gesture from non-gestures and other classes.

To compute the dissimilarity descriptor for any query gesture, we proceed as shown in Fig. 7. First we take a subset of the dictionary as the representation set used to build the dissimilarity features. We used a dense set selecting half of the dictionary elements. Both the dictionary gestures and the query gesture are resampled with spline interpolation to a fixed number of samples (20) equally spaced in time. Furthermore, gesture coordinates are translated to have the starting point of the palm in *(0,0,0)*.

Different dissimilarity scores between the query gesture and the elements of the representation set are then estimated and concatenated to create the novel gesture descriptor:

- palm trajectory dissimilarity: for each gesture of the representation set, we estimate a dissimilarity value equal to the sum of the Euclidean distances of the corresponding points in the query gesture. Given N gestures in the representation set, we have N features for the query gesture descriptor.
- hand articulation dissimilarity: for the query gesture and each gesture of the representation set, we estimated the evolution of the 9 distances between fingertips keypoints and between fingertips and the palm keypoint (Fig. 8). We then estimate 9 dissimilarity components as the sums over the corresponding time samples of the differences between the values of the 9

[2] Because we then have to feed the model with higher resolution images to keep the best accuracy possible.

[3] https://github.com/dynamic-hand-gestures-classification/dynamic-hand-gestures-classification

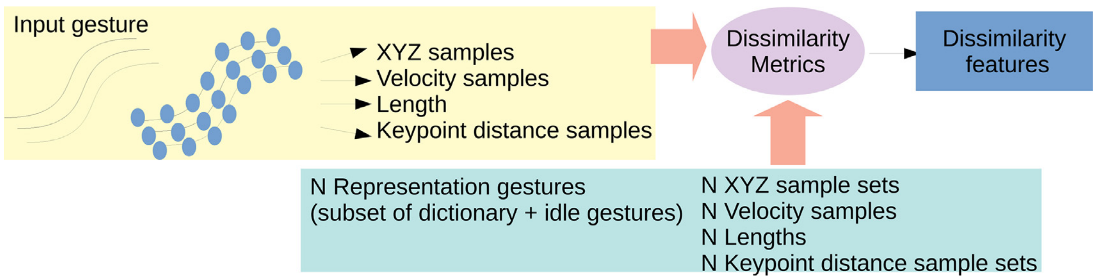


Fig. 7. Pipeline for the extraction of gesture representation and dissimilarity wrt the representation set selected from the dictionary.

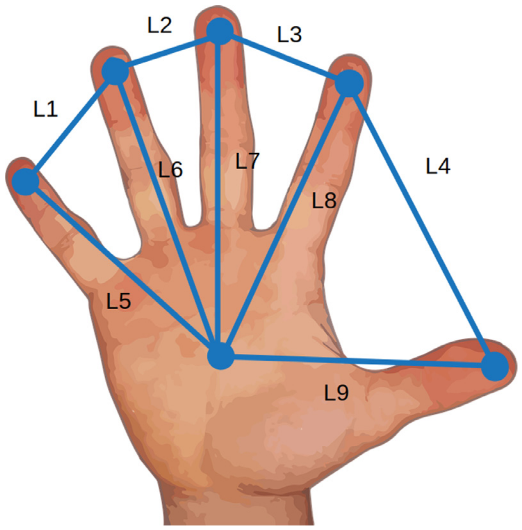


Fig. 8. Keypoints and the derived 9 distances used to evaluate the hand articulation dissimilarity component used in the method (see text).

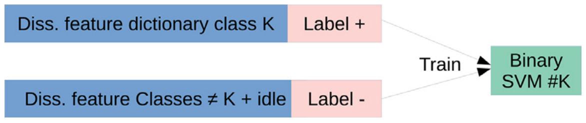

Diss. feature dictionary class K
Label +
Diss. feature Classes ≠ K + idle
Label -
Train
Binary SVM #K

Fig. 9. We train a specific SVM classifier to detect the different dictionary gestures against non-gestures (or other gestures).

distances in query and representation set gestures. Given N gestures in the representation set, we have thus 9N features for the query gesture descriptor.

- palm trajectory length dissimilarity: we added to the set the difference in length between the query gesture and the representation set gestures.
- palm velocity dissimilarity: for the resampled gestures, we estimated the module of the displacement between each sample and the next one and the sum of the differences of the corresponding displacements between the query gesture and the representation set gestures is the estimated dissimilarity.

The final dissimilarity vector has therefore 12N elements where N is the number of elements in the representation set. In our tests, we set N=18 and the size of the resulting descriptor is 3024.

For each gesture, we then train class-specific linear SVM classifiers using all the dictionary gestures and an additional set of “idle” gestures synthetically generated from some initial frames of the dictionary gestures and keeping then the hand approximately fixed along time. Each classifier is trained using binary labels (gesture vs non-gesture), e.g. we cast the problem as multiple single gesture detection, not requiring a complete non-gesture training set to cope with non-gesture sequences and false detection. In principle, we could have trained one-class classifiers for the task, but we found that the use of binary SVM classifiers trained in this way led to better results.

During the online gesture recognition, we use a sliding window approach to select fixed-size windows along the test sequence. We use three values of the windows size, chosen based on the distribution of the lengths of the dictionary gestures. These values are 38, 44 and 50 frames in order to deal with potential differences in dynamic gestures speed. Windows are moved along the sequence with steps of 5 frames; gestures in the windows are resampled in 20 steps, dissimilarity descriptors are estimated and fed into the single class classifiers. For each time step tested, the outputs of the classifiers for each window size are collected into an array of votes. Finally, sequences of frames with non-zero votes for a gesture class and longer than 1 are detected as gestures of the corresponding class.

The method has been implemented using Matlab R2019b. The sliding window classification with multiple window size has, in the current implementation a latency of 0.05 sec on a laptop with an Intel®Core^{TM}i7-9750H CPU and 16 Gb RAM, meaning that the method is suitable for real-time applications.

5. Evaluation

In this section, we present the results of the two methods whose outputs has been processed with the same evaluation criteria. For each of the 72 sequences, the raw results of the methods feature a list of gesture detections with the corresponding classes and starting frames. Classification results described in Section 3 have been then automatically estimated with a Matlab script that will be available together with the benchmark data.

5.1. View-based method results

Fig. 10 (left) shows the classification results for all the detected gestures, like the dissimilarity-based one, not counting false positives which are instead shown in Table 3. This method performs substantially better compared to the dissimilarity-based one on the *Swipe Left* and *Swipe Right* gestures with 0.87 correct classification ratio for both gestures. *Pinch* and *Tap* remain critical gestures at least in terms of correct classifications. This can be explained by the fact that the *Pinch* and *Tap* gestures have a strong movement component on the z axis and far less movement on the x and y axes, those best represented by the view captured by the 3D viewer. Providing also a side view to the classifier could improve its classification on these two gestures. The processed data is presented in Table 3. The ratio of false positives over the total number of ground truth gestures is lower compared to the dissimilarity-based with a total score of 0.18. The most critical gestures in terms of false positives for this method are *Expand* and *Swipe Right* with 0.95 and 1.53 ratios respectively, and they are the major contribution in raising the method's false positive average.

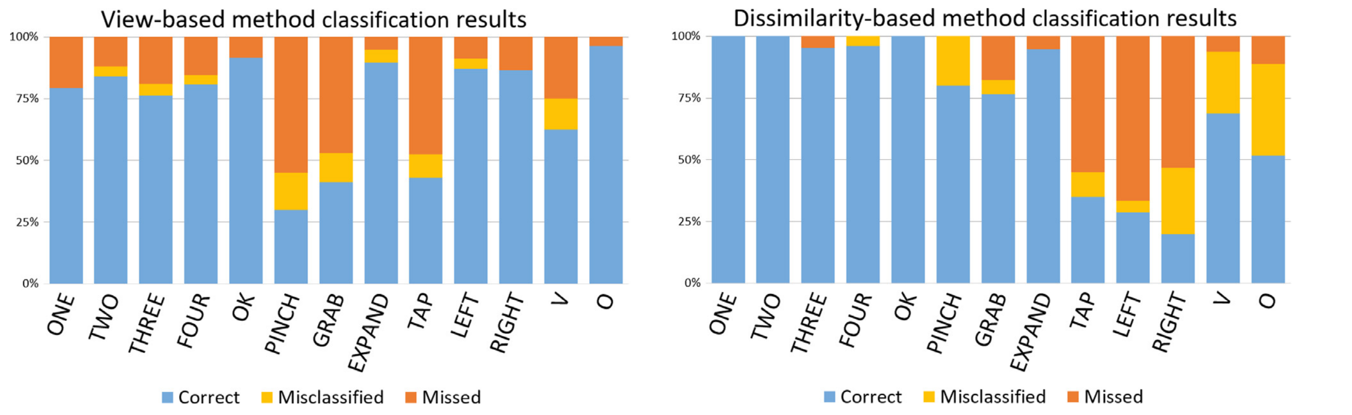


Fig. 10. Stacked bars chart showing the proportions between correct, misclassified and missed gestures for the view-based method (left) and the dissimilarity-based method (right).

Table 3
Classification results for the view-based method reporting different ratios and the F-score both class-by-class and overall.

View-based method results					
Class	Corr./tot	Miscl./tot	Missed/tot	FP/tot	F score
ONE	0.79	0.00	0.21	0.08	0.84
TWO	0.84	0.04	0.12	0.00	0.91
THREE	0.76	0.05	0.19	0.00	0.86
FOUR	0.81	0.04	0.15	0.00	0.89
OK	0.92	0	0.08	0.00	0.96
PINCH	0.3	0.15	0.55	0.00	0.46
GRAB	0.41	0.12	0.47	0.24	0.50
EXPAND	0.9	0.05	0.05	0.95	0.63
TAP	0.43	0.09	0.48	0.05	0.58
LEFT	0.87	0.04	0.09	0.09	0.89
RIGHT	0.87	0.00	0.13	1.53	0.51
V	0.63	0.12	0.25	0	0.77
O	0.96	0.00	0.04	0.04	0.97
Average	**0.74**	**0.05**	**0.20**	**0.18**	**0.77**

Table 4
Classification results for the dissimilarity-based method reporting different ratios and the F-score both class-by-class and overall.

Dissimilarity-based method results					
Class	Corr./tot	Miscl./tot	Missed/tot	FP/tot	F score
ONE	1.00	0.00	0.00	0.08	0.96
TWO	1.00	0.00	0.00	0.00	1.00
THREE	0.95	0.00	0.05	0.05	0.95
FOUR	0.96	0.04	0.00	0.23	0.88
OK	1.00	0.00	0.00	0.04	0.98
PINCH	0.80	0.20	0.00	0.25	0.78
GRAB	0.76	0.06	0.18	0.00	0.87
EXPAND	0.95	0.00	0.05	0.74	0.71
TAP	0.33	0.10	0.57	0.00	0.50
LEFT	0.26	0.04	0.70	0.09	0.39
RIGHT	0.20	0.27	0.53	0.00	0.33
V	0.69	0.25	0.06	0.69	0.58
O	0.52	0.37	0.11	0.37	0.55
Average	**0.75**	**0.10**	**0.15**	**0.19**	**0.77**

5.2. Dissimilarity-based method results

Fig. 10 (right), shows the classification results for all the detected gestures, not counting false positives which are instead shown in Table 4. This method presents some relevant issues with few gestures. *Tap*, *Swipe Left* and *Swipe Right* present an elevate number of missed detections, with a missed ratio of 0.55, 0.66 and 0.35 respectively. More data, including other metrics such as the number of false positives, the F-score and the average delay of the detected gesture beginning from the ground truth mark are reported in Table 4. The ratio of false positives considering all the gestures scores 0.19. The most critical gestures in terms of false positives are *Grab*, *Swipe V* and *Swipe O*. *Tap*, *Swipe Left* and *Swipe Right* have both a false positive ratio and a correct classification ratio below the average possibly indicating that overall, with this method, detections for these classes rarely trigger.

6. Discussion

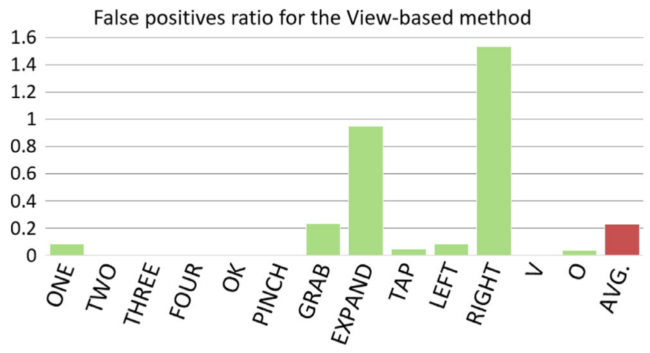
Our benchmark allows for a quick test of gesture recognition by working on finger trajectories, including different types of gestures and evaluation of a realistic online task.

The results obtained with the two methods proposed are promising, but show several issues that need to be addressed in future work:

- the difficulty of handling different types of gestures simultaneously, e.g. static/dynamic, coarse/fine;
- the difficulty in avoiding a significant number of false positives;
- the existence, albeit contained, of thresholding mechanisms within the two algorithms (therefore not an optimal end-to-end training of classifiers).

It is interesting to note that, while the average performance of the two methods are quite close, in particular no significant difference was found with a one-way ANOVA analysis on both corrected classifications and F-score averages ($p \gg 0.05$), they have quite different results on specific gestures: the view-based method has problems in the classification of gestures in which the main contribution to movement occurs along the z-axis. The dissimilarity-based method has problems in the detection of coarse gestures (*Left/Right Swipe*) and the *Tap* gesture, and seems in general more accurate on gestures characterized by gesture articulations. As the method is based on single class classifiers, this suggest that different metrics could be applied to compare hand trajectories and orientation.

In addition, the view-based method reports a large number of false positives on the *Expand* and *Swipe Right* gestures (see Fig. 11, left). While false positives on the former are easily explained by the fact that, in all respects *Expand* is an open hand with all five fingers visible (basically the most common resting position for any hand), on the latter it is more difficult to understand what triggers false detections. Probably the network has learned to classify a 45° counterclockwise rotation as the start of a *Swipe Right* gesture which, in reality, may or may not be the prelude to the actual gesture.



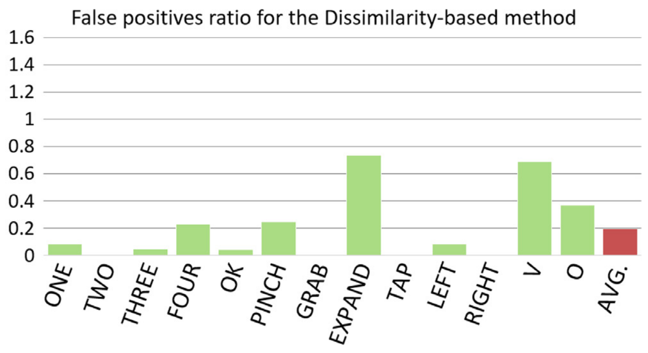


Fig. 11. Bar charts showing the ratios between false positives and ground truth gestures for the view-based method (left) and the dissimilarity-based method (right).

The dissimilarity-based method reports a large number of false positives for the coarse *Swipe V* and *Swipe O* gestures (see Fig. 11, right), that can be explained by a poor coarse comparison metric as the missed detection, and for the *Expand* gesture, that has been previously discussed. This may suggest to exclude this gesture from command gestures dictionaries for general purpose gestural interfaces, or to adopt specific rejection strategies.

There are many strategies that could be applied to both the view-based and the dissimilarity-based method to improve results.

For the view-based method, there is ample room for improvement:

- adding two more classes such as “noise” and “blank” carrying information about “non-gestures” could help in decreasing the number of false positives;
- as previously said, introducing a second side view to feed the network with double “composite” images of the 3D scene would greatly help the classifier to recognize gestures with movement mostly developing along the z-axis such as pinch and grab;
- performing further data augmentation on the training set to feed the network with the same sequences but rotated around x and y axes with slightly different angles. At the moment, the only rotations provided by *Fast.ai*’s standard data augmentation are only those applied on the 2D image, therefore around the z-axis;
- adding an LSTM layer at the end of the convolutional layers of the ResNet-50 model, to make the network learn along the time dimension.

For the dissimilarity-based method, several ideas to improve the results can be identified as well:

- as the method is based on class-specific classifiers, we could differentiate the features or even the labeling rules for the specific gesture type or class. In particular, improved metrics should be employed to estimate coarse trajectory dissimilarity. Furthermore, the current method does not exploit information related to the evolution of the hand orientation.
- Handcrafted features could be replaced by features derived from the raw dictionary data, for example by training dimensionality reduction methods like discriminative mappings or autoencoders.
- Currently we have no rejection strategies for gestures with similar parts (e.g. *Swipe V*, *Swipe O*, *Swipe Right*) that may be confused. By adopting specific strategies it may be possible increase the performance on them.
- False positives could be reduced by including more complex, synthetically generated patterns in the non-gesture class of the training set.

7. Conclusion

In this paper, a novel benchmark for online gesture detection and recognition, SFINGE 3D, has been presented. The main novelty of the benchmark is its suitability for the detection and classification of gestures in online realistic use cases where gestures are sequentially performed and not a priori segmented.

We also proposed and tested on the novel benchmark two different approaches for the online gesture recognition, one based on visual rendering and Convolutional Neural Networks and the other based on geometry-based handcrafted features and dissimilarity-based classifiers. The results obtained are promising: both the methods are suitable for real time online application and provide a similar detection rate of about 75% on average. The main weaknesses of both the methods are related to a low accuracy and a high number of false positives for a limited number of the dictionary gestures. This means that, on restricted dictionaries (different for each method), the quality of the gesture detection methods would be already rather good. Moreover, adding the non gesture class can further improve the recognition capabilities by reducing the number of false positives.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Ariel Caputo: Conceptualization, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Andrea Giachetti:** Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Franca Giannini:** Conceptualization, Writing - original draft, Writing - review & editing. **Katia Lupinetti:** Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing, Visualization. **Marina Monti:** Conceptualization, Writing - original draft, Writing - review & editing. **Marco Pegoraro:** Software, Investigation, Validation, Writing - review & editing. **Andrea Ranieri:** Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization.

References

[1] Lugaresi C, Tang J, Nash H, McClanahan C, Uboweja E, Hays M, et al. Mediapipe: A framework for building perception pipelines. 2019. arXiv:1906.08172.
[2] Bazarevsky V., Zhang F.. On-device, real-time hand tracking with mediapipe. 2019. https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html.

[3] De Smedt Q, Wannous H, Vandeborre J-P, Guerry J, Le Saux B, Filliat D. Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset; 2017.
[4] Avola D, Bernardi M, Cinque L, Foresti GL, Massaroni C. Exploiting recurrent neural networks and leap motion controller for the recognition of sign language and semaphoric hand gestures. IEEE Trans Multimed 2018;21(1):234–45.
[5] Tao W, Leu MC, Yin Z. American sign language alphabet recognition using convolutional neural networks with multiview augmentation and inference fusion. Eng Appl Artif Intell 2018;76:202–13.
[6] Maghoumi M, LaViola Jr JJ. Deepgru: deep gesture recognition utility. In: Proceedings of the international symposium on visual computing. Springer; 2019. p. 16–31.
[7] Mazzini L, Franco A, Maltoni D. Gesture recognition by leap motion controller and lstm networks for cad-oriented interfaces. In: Proceedings of the international conference on image analysis and processing. Springer; 2019. p. 185–95.
[8] Lupinetti K., Ranieri A., Giannini F., Monti M.. 3d dynamic hand gestures recognition using the leap motion sensor and convolutional neural networks. 2020. arXiv:2003.01450.
[9] Boulahia SY, Anquetil E, Multon F, Kulpa R. Dynamic hand gesture recognition based on 3d pattern assembled trajectories. In: Proceedings of the 2017 seventh international conference on image processing theory, tools and applications (IPTA). IEEE; 2017. p. 1–6.
[10] Caputo F, Burato S, Pavan G, Giachetti A, Voillemin T, Wannous H, et al. Shrec 2019 track: online gesture recognition. In: Proceedings of the eurographics workshop on 3D object retrieval; 2019.
[11] De Smedt Q, Wannous H, Vandeborre J-P. Skeleton-based dynamic hand gesture recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops; 2016. p. 1–9.
[12] Wei L, Zheng T, Jinghui C. Dynamic hand gesture recognition with leap motion controller. IEEE Signal Process. Lett. 2016;23(9):1188–92.
[13] Escalera S, Gonzàlez J, Baró X, Reyes M, Lopes O, Guyon I, et al. Multi-modal gesture recognition challenge 2013: dataset and results. In: Proceedings of the fifteenth ACM on international conference on multimodal interaction; 2013. p. 445–52.
[14] Deployment of LSTMs for Real-Time Hand Gesture Interaction of 3D Virtual Music Instruments with a Leap Motion Sensor. Zenodo; 2018. 10.5281/zenodo.1258041
[15] Ameur S, Khalifa AB, Bouhlel MS. A comprehensive leap motion database for hand gesture recognition. In: Proceedings of the 2016 seventh international conference on sciences of electronics, technologies of information and telecommunications (SETIT). IEEE; 2016. p. 514–19.
[16] Kuznetsova A, Leal-Taixé L, Rosenhahn B. Real-time sign language recognition using a consumer depth camera. In: Proceedings of the IEEE international conference on computer vision workshops; 2013. p. 83–90.
[17] Van den Bergh M, Van Gool L. Combining rgb and tof cameras for real-time 3d hand gesture interaction. In: Proceedings of the 2011 IEEE workshop on applications of computer vision (WACV). IEEE; 2011. p. 66–72.
[18] Chen Y, Zhao L, Peng X, Yuan J, Metaxas D.N. Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. 2019. arXiv preprint arXiv:190708871.
[19] Manning CD, Raghavan P, Schütze H. Evaluation in information retrieval. Cambridge University Press; 2008. p. 139–61. doi:10.1017/CBO9780511809071.009. chap. 8
[20] Campagnola L, Klein A, Larson E, Rossant C, Rougier N. Vispy: Harnessing the gpu for fast, high-level visualization; 2015. p. 91–6. doi:10.25080/Majora-7b98e3ed-00e. http://vispy.org.
[21] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. CoRR 2015;abs/1512.03385. http://arxiv.org/abs/1512.03385.
[22] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. (IJCV) 2015;115(3):211–52. doi:10.1007/s11263-015-0816-y.
[23] HasanPour SH, Rouhani M, Fayyaz M, Sabokrou M. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. CoRR 2016;abs/1608.06037. http://arxiv.org/abs/1608.06037.
[24] Howard J, Gugger S. Fastai: A layered api for deep learning. Information 2020;11(2):108. doi:10.3390/info11020108.
[25] Fang L, Monroe F, Novak SW, Kirk L, Schiavon CR, Yu SB, et al. Deep learning-based point-scanning super-resolution imaging. bioRxiv 2019. doi:10.1101/740548. https://www.biorxiv.org/content/early/2019/10/24/740548.
[26] Pekalska E, Paclik P, Duin RP. A generalized kernel approach to dissimilarity-based classification. J. Mach. Learn. Res. 2001;2(Dec):175–211.