This is a pre print version of the following article:

SHREC 2019 Track: Online Gesture Recognition / Caputo, F.M., Burato, S., Pavan, G., Voillemin, T., Wannous, H., Vandeborre, J.P., Maghoumi, M., Taranta, E.M., Razmjoo, A., J. J. Laviola Jr., ., Manganaro, F., Pini, S., Borghi, G., Vezzani, R., Cucchiara, R., Nguyen, H., Tran, M.T., Giachetti, A.. - 160897:(2019), pp. 93-102. (12th Eurographics Workshop on 3D Object Retrieval, 3DOR 2019, in conjunction with the 40th Annual Conference of the European Association for Computer Graphics, EG 2019 Genova 5-6 May 2019) [10.2312/3dor.20191067].

The Eurographics Association
Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

09/11/2026 07:21

(Article begins on next page)

Eurographics Workshop on 3D Object Retrieval (2019)
SHREC - 3D Shape Retrieval Contest (2019)
S. Biasotti, G. Lavoué, B. Falcidieno, and I. Pratikakis (Editors)
R. C. Veltkamp (SHREC organizer)

SHREC 2019 Track: Online Gesture Recognition

F.M. Caputo[*,1], S. Burato[*,1], G. Pavan[*,1], T. Voillemin[2], H.Wannous[*,2] , J.P. Vandeborre[2], M. Maghoumi[3,4], E. M. Taranta II[4] , A. Razmjoo[4,5] , J.J. LaViola Jr.[4], F. Manganaro[6], S. Pini[6], G. Borghi[6], R. Vezzani[6], R. Cucchiara[6], H. Nguyen[7], M.T. Tran[7], A. Giachetti[*,1],

[*]Track Organizers

[1] Department of Computer Science, University of Verona, Italy
[2] IMT Lille Douai, CRIStAL UMR CNRS 9189, Univ. Lille, France [3] NVIDIA
[4] University of Central Florida
[5] University of California, San Francisco
[6] University of Modena and Reggio Emilia Centro di Ricerca Interdipartimentale Softech-ICT Department of Engineering "Enzo Ferrari", Italy
[7] Faculty of Information Technology, University of Science, VNU-HCM

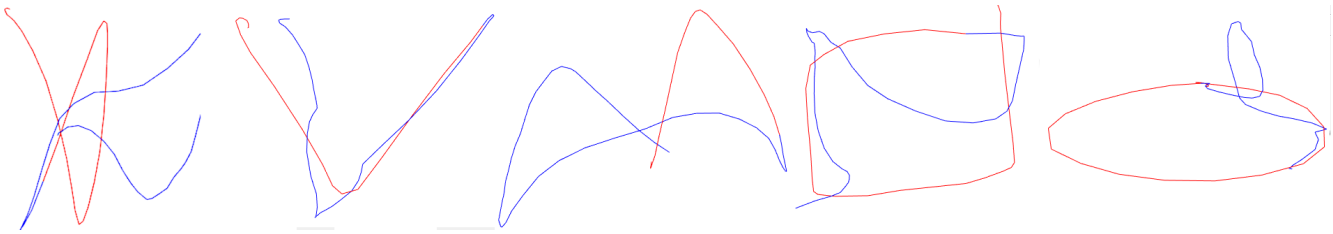

Figure 1: *Gestures to be detected (red) hidden in long 3D trajectories (blue). From left: cross, V-mark, caret, square, circle.*

Abstract

This paper presents the results of the Eurographics 2019 SHape Retrieval Contest track on online gesture recognition. The goal of this contest was to test state-of-the-art methods that can be used to online detect command gestures from hands' movements tracking on a basic benchmark where simple gestures are performed interleaving them with other actions. Unlike previous contests and benchmarks on trajectory-based gesture recognition, we proposed an online gesture recognition task, not providing pre-segmented gestures, but asking the participants to find gestures within recorded trajectories. The results submitted by the participants show that an online detection and recognition of sets of very simple gestures from 3D trajectories captured with a cheap sensor can be effectively performed. The best methods proposed could be, therefore, directly exploited to design effective gesture-based interfaces to be used in different contexts, from Virtual and Mixed reality applications to the remote control of home devices. (see http://www.acm.org/about/class/class/2012)

CCS Concepts

• ***Human-centered computing*** → ***Gestural input;***

1. Introduction

Interaction in Virtual and Mixed reality as well as in smart environments can be based on different input channels. When audio channels are not usable, and when handheld devices cannot be easily applied (that is the case of several potential applications of Mixed and Virtual Reality), gestural interfaces can be a good solution. Gestural interaction is quite popular in touch-based interfaces, where simple geometry processing solutions proved to be quite effective to design and implement user interfaces [VAW18] even in the case of low computational resources. Similar approaches can be applied also for mid-air gesture recognition, as there are consolidated technologies able to track hand gestures without the necessity of using handheld controllers or wearables [GCC*16], but using simple depth sensors like the Leap Motion controller to capture the finger trajectories. Several methods have been proposed in recent years to recognize gestures from 3D trajectories, both using simple heuristics [CPC*18] and advanced pattern recognition tools like recurrent networks [MLJ18, DMXY18, CWG*19]. In order to evaluate the feasibility of a simple design of gestural interfaces with this kind of algorithm, it is, however, necessary to test them in simple, but realistic use case, starting from very simple ones. This fact motivated us to create a very simple dataset with classic command gestures similar to those that can be employed in touch interfaces, propos-

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

DOI: 10.2312/3dor.20191067

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

Figure 2: *Gesture templates suggested to the subjects participating in the data acquisition. From left: cross, circle, V-mark, caret, square.*

ing a quite simple online gesture recognition task from 3D hand trajectories and to set up this gesture recognition contest.

2. Task and motivation

The task proposed is intended as a first step in evaluating the ability of different algorithms to detect and classify online command gestures usable in immersive or domotic interfaces. For this reason, we considered only very simple gestures without the need for articulating fingers, that should be recognized by simply analyzing single trajectories. The participants were asked to classify gestures from a set of 3D trajectories. Each trajectory is composed of a sequence of 3D points and quaternions for all the hand joints plus the palm and contains a segment that represents a gesture to recognize. The task proposed simulates a real-time detection and recognition problem, therefore the methods submitted had to process the sequence progressively and give a classification answer when the first gesture in the trajectory was detected without examining the entire trajectory first. With this additional rule, the task implicitly requires the solution of the gesture vs non-gesture discrimination problem since the trajectory is to be processed step-by-step, at any given time t_i there would be no guarantee of the presence of a gesture in the segment t_0 - t_i.

Furthermore, we also asked the contests' participants to provide an estimation of the time (i.e. entry in the sequence) at which the gesture had supposedly started and the decision time when the algorithm detected the gesture.

To better focus on testing the ability of the existing method of recognizing single-trajectories in an online setting, the gesture dictionary proposed for our task is simpler than the one proposed in the Shrec'17 track on 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset [SWV*17], that mixed gestures recognizable for the whole hand trajectory and gestures characterized by fingers' articulation. We used only five well defined gestures characterized by a particular whole hand trajectory shape: cross, V-mark, caret, square, circle see Figure 2). These gestures had to be detected and recognized within long sequences including different motion patterns related to other activities.

3. Data creation

In order to build the dataset, we created an interactive environment with Unity 3D, showing a 3D virtual interface including objects and widgets. We asked a set of subjects to perform randomized sequences of pre-defined actions on the interface in an immersive Virtual Reality setting, wearing a Head Mounted Display (Oculus Rift). The actions consisted of selecting objects, clicking on virtual buttons, moving a slider and spinning a globe with a swipe gesture.

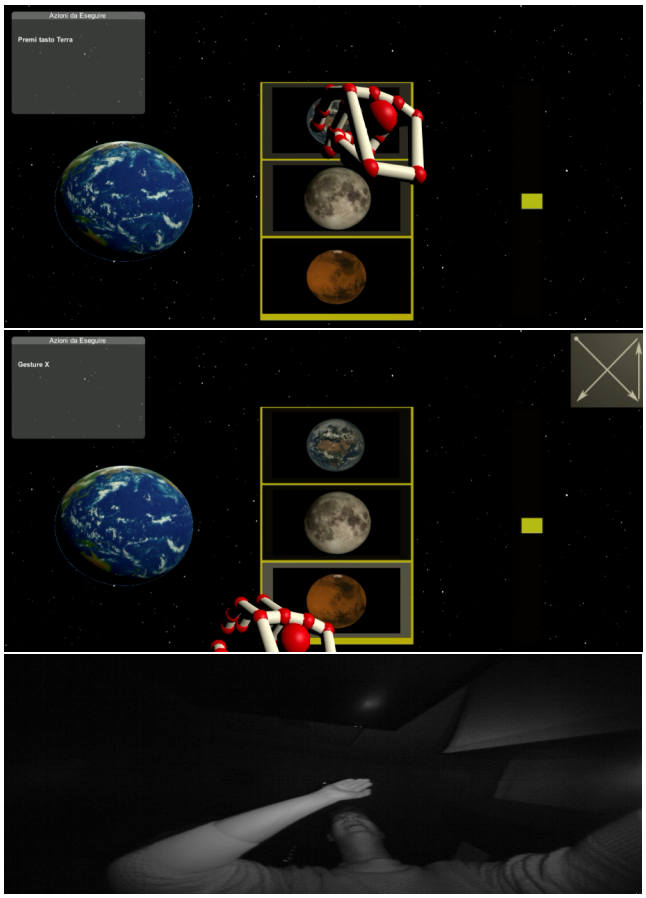

Figure 3: *Top, middle: two frames of the VR interaction application used for the dataset capture. Subjects had to perform a set of interactive tasks with their hands in an immersive VR visualization, and the command gestures to be detected are interleaved within the other actions. Bottom: the infrared camera view of the Leap Motion acquisition. We also recorded image sequences that could be used as well for recognition tests.*

Within each sequence of actions, subjects were also asked to perform one of the gestures to be recognized. These gestures are simple 2D patterns, that should be executed by users in a well-defined way (see Figure 2). Gesture orientation is clearly relevant to discrimination. Users had precise instructions to draw them before (and during) the task (see Figure 3). Hand trajectories have been captured with a Leap Motion sensor. Thanks to the sensor API we recorded the full position and orientation of the hands' joints, even if the information should be redundant for the contest task, as the gestures did not involve fingers' articulation. We provided, however, the full data to the participants.

The final dataset created with this procedure consists of 195 recordings performed by 13 different subjects, each one including a gesture in a different position, with known begin and end

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

frames. The dataset has been split in a training set of 60 recordings collected from 4 subjects, given to the participants with annotated start and end of the gestures and gesture label, and a test set, with the remaining 135 recordings, provided to the participants without annotations.

4. Evaluation

Each trajectory in the provided dataset contained one gesture in a precise trajectory interval. Participants were required to send a list of annotations marking estimated frames for the gesture beginning and detection decision time, and the labels for the guessed class.

In our evaluation, a gesture is considered correctly detected and labeled if the method provided the correct label locating the gesture within 2.5s. from the actual gesture time window.

For the given task we evaluated the following scores: percentage of correct detections and labeling of the gestures, percentages of errors divided in gestures with correct time location of the detection, but the wrong label, gestures detected before the correct time location, gestures detected after the correct time window or not detected. We also analyzed the results obtained separately for each gesture label looking at the ability of the different methods to detect and discriminate specific trajectories of the dictionary and the confusion matrices related to wrong labeling of gestures when detected in the approximately correct time frame.

Finally, we estimated the time difference between the decision time of the online algorithm and the actual start and end timesteps of the gestures.

5. Participants and methods

Four groups from different countries registered for the contest and sent results: one team from IMT Lille Douai / CRIStAL, one from NVIDIA, University of Central Florida and University of California, San Francisco, one from the University of Modena and Reggio Emilia, and one from Ho Chi Minh City University of Science. All the groups propose methods based on Neural Networks. As a baseline method, we tested a simple geometry based method (sliding window 3 Cent), developed at the University of Verona [CPC*18]. All the methods proposed are described in the following subsections.

5.1. Baseline: sliding window 3 cent (SW 3-Cent) by Fabio Marco Caputo and Andrea Giachetti

The sliding window 3-cent is the online version of its previous offline version [CPC*18]. The idea in both versions is to guess the class of a new trajectory by computing distances from the labeled trajectories in the training set with a certain metric and then simply using a KNN to chose the candidate class. The core steps to compare and calculate the distance between two trajectories are:

- A resampling of both the trajectories in order to obtain the same number of points, using spline interpolation.
- A centering operation to move the reference frame origin on the centroid for each trajectory.
- A scaling operation to fit a fixed-size bounding-box.
- Finally, the distance is calculated as the sum of squared distances between pairs of points at the same index.

For the online version, there are some prior steps necessary to deal with for the localization of the correct segment to compare and classify. Hence the introduction of a sliding-window with the idea of monitoring a certain number of past frames and continuously compare the trajectory in the window with the steps previously described. The size of the window is pre-calculated as the average length of all the trajectories in the training set. Once the algorithm runs online, with each new frame of data available, the trajectory contained in the last segment of the pre-calculated size is used for a tentative classification. The decision on whether the segment in a window is to be considered belonging to any of the classes or none relies on a fixed distance threshold. This threshold is learned from training data as follows: a histogram of inter-class distances is created as well as a histogram of the inter-class distances, using also samples of a non-gesture class for this training procedure. The threshold is then set corresponding to the bin minimizing the sum of extra-class distances below it and the intra-class distances higher than it.

To add more flexibility to the size of the window, four other sizes extending and cropping the original window by 1/6 and 2/6 the original one, are also used simultaneously. If any of the comparisons return a distance below the threshold the classification of a trajectory inside the window takes place. If at the same time (in terms of frames of data), more distances are returned below the threshold, the one with the lowest distance is used for the classification step. This method has an average time-per-frame processing time of 0.0017s running on MATLAB and a Xeon E3-1231 v3 CPU and maximum that never exceeds the sampling rate of the dataset of 0.05s.

5.2. Palm-Index recurrent network (PI-RN), all joints recurrent network (AJ-RN) by Théo Voillemin, Hazem Wannous, Jean-Philippe Vandeborre.

Authors used a simple recurrent network model to real-time classify the gesture contained in a trajectory. Two methods were tested, one with two streams on the palm and the index joints data, inspired by [DS17], the other with only one stream but on all joints data available. A stream is an alternation of two LSTM layers with one fully connected layer, the end of the network is another fully connected layer with an output on the number of gesture plus a *no_gesture* label. In case of two streams, output of each stream are concatenate together then pass to the end of the network. The network is then trained with a RMSprop optimizer.

For the online detection, with the precise annotation of the start and the end of a gesture in the training data set, authors added a *no_gesture* label to train their recurrent network model to recognize each gesture but also when its start and end. With the small amount of training data available, they decided to use a 10-fold cross validation to find the best hyper-parameters to avoid over and under fitting. It was necessary to ensure that every different gesture is present in each fold. Authors found that a 256 and a 128 numbers of LSTM cells and 100 epochs to train on all the training data set is adequate.

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

The recurrent architecture then permits to process the test data set in an online way by providing the data frame by frame to the network. The states of the LSTM cells gates are saved and restored between each frame processed and re-initiate to zero at each beginning of a new sequence.

5.3. uDeepGRU: Unsegmented Deep Gesture Recognition Utility by Mehran Maghoumi, Eugene M. Taranta II, Alaleh Razmjoo, Joseph J. LaViola Jr.

uDeepGRU is an end-to-end deep learning-based unsegmented gesture recognizer designed for SHREC 2019 track on online detection of simple gestures from hand trajectories. It is based on DeepGRU [MLJ18], a deep learning-based recognizer for action and gesture recognition of samples collected form commodity input devices. Authors extended DeepGRU and applied it to the task of recognizing unsegmented hand gestures in an online application scenario.

5.3.1. Data Preparation

Each frame of the data is treated as one 48-dimensional vector f_t, containing the concatenated 3D position of all joints, exactly as appears in data files, for time step t. Authors z-score normalize every data using the mean and standard deviation of the training set. Afterwards they extract a direction vector $x_t = f_t - f_{t-1}$ for every time step and $x_0 = [0]$. This data is then fed to the network.

Deep learning techniques require large quantities of data in order to learn the underlying neural network weights. However, in many practical applications, such as with the SHREC'19 training dataset, quantities are limited. This limitation has been solved by employing synthetic data generation techniques to augment and expand the dataset. Namely, four complementary techniques have been used: gesture path stochastic resampling (GPSR) [TIMPLJ16], Fourier coefficient perturbations, time-series inversion, and rotations. First, GPSR uses a two-step approach to generate synthetic variations of a given trajectory, where one samples random points along the trajectory and then normalizes the distance between each pair of consecutive points. This process changes the velocity profile of a sample such that changes in curvature are temporally unique relative to the original input, thereby resulting in a unique shape. Second, since GPSR does not significantly warp low frequency information (straight edges will remain straight), the trajectory's Fourier coefficients have been further modified. Authors perform a discrete Fourier transform (DFT) on a given trajectory, randomly adjust the amplitude of each coefficient by $\pm 20\%$, and then perform an inverse DFT to synthesize a variant of the original trajectory. The effect of this transformation is that a given sample will have "wobbles" not previously embedded within the trajectory (e.g., previously straight edges may now have some bend). Note that results will also have additional high frequency noise that one can remove with a low pass filter; however in their testing, authors found this noise was unharmful. Third, in order to increase the amount of non-gesture training data available to the system, they reversed each training sample's trajectory. Finally, authors also added random rotations to each trajectory in order to increase orientation variance. Specifically, they randomly rotated trajectories ± 10 degrees around the x-axis. Using these techniques, 5 new synthetic samples per each real sample have been generated and authors trained uDeepGRU using all available data.

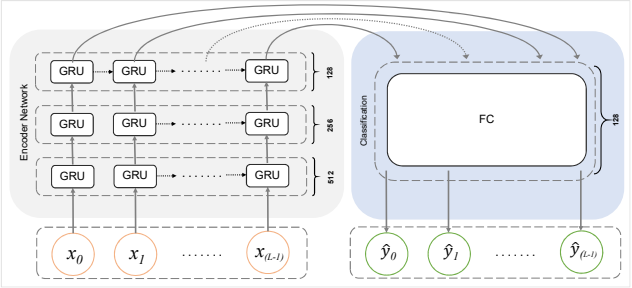

Figure 4: *uDeepGRU architecture which consists of an encoder network and a classification subnetwork.*

5.3.2. Architecture

Figure 4 depicts the network's architecture, which is very similar in spirit to DeepGRU [MLJ18]. At each time step, the network takes as input the feature vector x_t and produces the output label $\hat{y}_t \in \{\text{None, X, O, V, ^,[ ]}\}$. Note that no-gestures are treated as their own classes.

The model consists of an encoder and a classification subnetwork. Authors use unidirectional gated recurrent units (GRUs) [CvMG*14] as the recurrent layers of the encoder network with the following transition functions:

$$
\begin{aligned}
r_t &= \sigma\left(\left(W_x^r\, x_t + b_x^r\right) + \left(W_h^r\, h_{(t-1)} + b_h^r\right)\right) \\
u_t &= \sigma\left(\left(W_x^u\, x_t + b_x^u\right) + \left(W_h^u\, h_{(t-1)} + b_h^u\right)\right) \\
c_t &= \tanh\left(\left(W_x^c\, x_t + b_x^c\right) + r_t\left(W_h^c\, h_{(t-1)} + b_h^c\right)\right) \\
h_t &= u_t \circ h_{(t-1)} + \left(1 - u_t\right) \circ c_t
\end{aligned}
\tag{1}
$$

where σ is the sigmoid function, $\circ$ denotes the Hadamard product, r_t, u_t and c_t are reset, update and candidate gates respectively and W_p^q and b_p^q are the trainable weights and biases. In the encoder network, h_0 of all the GRUs are initialized to zero. The output of every GRU unit depends on all previously observed time steps only, making the network suitable for use in online recognition applications.

The classification subnetwork consists of a fully connected layer, preceded by batch normalization [IS15] and dropout [HSK*12]. Contrary to DeepGRU, uDeepGRU is a shallower network and does not include an attention [BCB15] subnetwork. These omissions were necessary to reduce overfitting due to the small training set of this track.

The network is trained end-to-end on the training set, with 10% of the data withheld for validation. Given a feature vector x_t at each time step, the network outputs a set of class-conditional probabilities $P(\hat{y}_t|x_t)$ where $\hat{y}_t$ is the predicted label of the input frame at time step t. To reduce the difference between predicted class labels $\hat{y}$ and the ground truth labels y during training, the algorithm

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

minimizes the unweighted sum of two losses, namely the cross-entropy loss and the negative Sorensen-Dice coefficients, which in turn maximize the F_1 score computed between per-frame predicted and ground truth labels.

The implementation of uDeepGRU is done with the PyTorch [PGC*17] framework. Authors use the Adam solver [KB14] ($\beta_1 = 0.9, \beta_2 = 0.999$) and the initial learning rate of 10^{-3} to train their model. The mini-batch size for the experiments is 128. Training is done on a machine equipped with one NVIDIA GeForce GTX 1080 GPUs, Intel Core-i7 6850K processor and 32 GB RAM. Authors save the model that produces the best F_1 score on the validation set.

5.3.3. Results

At test time, authors run each test sample through the network and obtain per-frame class labels $\hat{y}_t$. The outputs may optionally be post-processed by 1) thresholding the class-conditional probabilities based on a predefined threshold T, and 2) ensuring that at least C frames before the current frame are assigned the same output label.

Authors obtained 3 sets of results. The first set of result is the unprocessed output of the network obtained after applying the best-performing model on the supplied test data. The second set consists of the results of an ensemble of six networks, each trained on a different portion of the training set. The output of the models are averaged and post-processed with T=0.5, C=5 before the final labels are generated. The last set of experiments contain the results of the first set, plus post processing applied to the raw outputs with T=0.5, C=1.

5.4. DeA: *Divide et Agnosce* by Fabio Manganaro, Stefano Pini, Guido Borghi, Roberto Vezzani, Rita Cucchiara

This is a deep learning method to automatically detect and recognize simple gestures from hand trajectories acquired through the *Leap Motion* device.
The proposed approach consists of two different neural networks: the first one performs the on-line temporal segmentation (*divide*) task, *i.e.* the detection of the beginning and end of a gesture, while the second one is specifically designed for the classification (*agnosce*) task, *i.e.* the generation of the gesture label.
These modules cooperate with each other in a cascade-based manner: the segmentation block continuously processes the sequence of input data and identify gesture boundaries, estimating if the current frame contains a gesture or not. Then, the following data are stored until the detection of the end of the gesture. In the meantime, the buffer is classified by the on-line classification module.

5.4.1. Feature Extraction and Data Normalization

For each frame, 112 values are acquired through the sensor and the low level SDK representing the x, y, z coordinates of the palm center and of 15 joints of the fingers. In addition, joint orientations expressed as quaternions, are provided.
Authors expand this feature vector adding measures of the speed and the acceleration of each joint, obtaining a final feature vector of size 208. Given the sequence of the 3D position of the i-th joint $J_i^t = (x_i^t, y_i^t, z_i^t)$ at time t, speed $\mathbf{s}$ and acceleration $\mathbf{a}$ are computed following these formulas:

$$\mathbf{s}_i^t = \left[x_i^t - x_i^{(t-1)},\ y_i^t - y_i^{(t-1)},\ z_i^t - z_i^{(t-1)}\right] \tag{2}$$

$$\mathbf{a}_i^t = \left[x_i^t - 2x_i^{(t-1)} + x_i^{(t-2)}, y_i^t - 2y_i^{(t-1)} + y_i^{(t-2)}, z_i^t - 2z_i^{(t-1)} + z_i^{(t-2)}\right] \tag{3}$$

A normalization step is then applied: the mean and the standard deviation values for each feature across the training dataset are computed and used to obtain data with 0 mean and unit variance. The final values are provided as input to both the segmentation and the classification modules.

5.4.2. Model Architecture

Each module architecture is based on a *Long Short-term Memory* (LSTM) [HS97] block. LSTMs are a class of recurrent neural networks that have achieved good performance on many tasks based on the processing of temporal-coherent data sequences, like visual recognition [DAHG*15] and image captioning [KFF15].
The proposed LSTM models can be expressed with the following equations:

$$I_t = \sigma(W_i x_t + U_i H_{t-1} + b_i) \tag{4}$$
$$F_t = \sigma(W_f x_t + U_f H_{t-1} + b_f) \tag{5}$$
$$O_t = \sigma(W_o x_t + U_o H_{t-1} + b_o) \tag{6}$$
$$G_t = tanh(W_c x_t + U_c H_{t-1} + b_c) \tag{7}$$
$$C_t = F_t \odot C_{t-1} + I_t \odot G_t \tag{8}$$
$$H_t = O_t \odot tanh(C_t) \tag{9}$$

where F_t, I_t, Ot are the gates, C_t is the memory cell, G_t is the candidate memory, and H_t is the hidden state. W, U, and b are learned weights and biases, while x_t corresponds to the input at time t as defined in the previous section. Finally, the $\odot$ operator is the element-wise product.

5.4.3. *Divide*: the Segmentation Module

The segmentation module is trained to detect the beginning and the end of a single hand gesture. Authors exploit the LSTM model described in Sec. 5.4.2, with a hidden size of 128 units and 2 layers, adding a fully connected layer of 2 classes and a *softmax* activation function to obtain the final output. Authors apply dropout regularisation ($p = 0.2$) [SHK*14] on the model and its output.
During the training phase, the network is fed with a complete sequence in which each frame is labelled either as gesture, if it lies between the start and the end of the gesture, or as no-gesture, based on dataset annotations. The *binary categorical cross-entropy* is exploited as loss function. Authors adopt a learning rate of 10^{-3} and the *Adam* [KB14] optimizer.

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

5.4.4. *Agnosce*: the Classification Module

Given an input sequence containing a gesture, the classification module is trained to output the corresponding gesture label. The LSTM described in Sec. 5.4.2 is used with a hidden size of 256 units and 2 layers, followed by a fully connected layer with size 5. Authors apply the *softmax* activation and the *categorical cross-entropy* loss function.
This network is trained using fixed-length sequences as input, a batch size 2 and a learning rate of 10^{-2} dynamically decreased every 30 epochs by a factor of 10.
Each sequence is cropped on the beginning and the end of the gesture according to the dataset annotations. If the obtained sequence is shorter than the longest cropped sequence of the train split of the dataset, the first frame is repeated until reaching the fixed length.

5.4.5. Running Procedure

The proposed method processes the input data frame by frame, *i.e.* row by row. After the feature extraction and the normalization procedures, data is fed into the segmentation module. The beginning and the end of a gesture is identified in presence of n consecutive frames with the same segmentation label. In the experiments the value of $n = 5$ was used.
A temporary buffer is maintained, accumulating data from the detection of the gesture start until the n-th consecutive detection, but freed if a non-gesture is detected. As soon as n consecutive rows are detected as a gesture, the classification module is initialized with the last n rows (temporarily saved in the buffer) and it is continuously updated until the end of the gesture is detected.
The final gesture classification corresponds to the last prediction before the gesture end.

5.5. Segment LSTM by Hung Nguyen and Minh-Triet Tran (Faculty of Information Technology, University of Science, VNU-HCM)

5.5.1. Data preprocessing

Authors performed data augmentation by cropping each trajectory into sub-trajectories with the window size of 20. Since the sampling rate is 0.05s, 20 frames contain information of trajectory in 1 second period. Next, the sub-trajectories (from now on will be called segments) are divided into two groups: - The first group contains segments that start from (i +- 0.1n) and end at (j +- 0.1n) with i, j, n is the beginning, ending and length of the gesture in each trajectory, 10 random segments in each trajectory are put into this group, so it has 600 segments in total, each of them is labelled from 1 to 5 according to the gesture they have. - Other segments are put into the second group and labelled 0 as non-gesture segments. To keep the training data distribution even, authors random picked 600 from the second group, combined with 600 from the first group gave a training dataset with 1200 segments.

5.5.2. Training

Authors adopt an LSTM network with input size is 112 (all of the position and rotation values of hand palm and finger joints), hidden dimension is 32. The outputs of network are fed into a linear layer to get the scores for 6 classes (5 gestures and 1 class for non-gesture). The parameters are optimized using negative log likelihood loss and the training process took 5000 epochs to complete. Since splitting data for training and validating may make the data insufficient and could break its distribution, authors decided to train on all 1200 segments in the training dataset.

5.5.3. Prediction and Classification

Although this is an online method, authors considered the data in less than 20 frames (1 second) not sufficient for classifying and predicting the gesture. Therefore, the test data is cropped into 20-frame-length segments up to the latest frame. All of them are fed into the network and the output scores are collected, the segment that contains highest score and is predicted as a gesture (labelled from 1 to 5) will be chosen as the gesture label and decision time of the gesture is the first frame of the segment. The prediction time is the first frame of nearest segment that is labelled as non-gesture (0) before the decision time.

Authors performed 2 runs, Run 1 with the highest accuracy weight and Run 2 with the weight estimated at epoch 5000.

6. Results

Table 1 shows the outcomes of the first detection task, with the percentage of gestures correctly detected and classified and the percentages of false positives detected before the gesture, misclassified gestures within the correct time window and missed gestures. The error in the gesture start location is also reported.

The method providing the best results is the uDeepGRU technique, based unidirectional gated recurrent units, trained using smart data augmentation methods, showing that Neural Networks can be trained with a limited number of examples of gestures. The run using an ensemble of networks gave the best scores.

On the other hand, a simple trajectory matching method (SW-3cent) provided results that are not too far from the best, showing that this flexible approach, often used for touchscreen interaction for mobile devices could be feasible also in the case of mid-air gestures.

Other network-based methods provided poorer results with respect to uDeepGRU and SW-3cent, probably due to the fact that these methods require a relevant effort for optimizing training strategies working with a limited number of examples and participants had a limited time to prepare the submission. Table 1 shows that the other methods (DeA, AJ-RN, PI-RN, Seg LSTM) are detecting a lot of false positives, and typically provide a false detection as the first result. Proper tuning of the method could avoid this effect.

A relevant issue in the design of a simple gesture-based interface based on hand trajectory only is the choice of an optimal gesture dictionary. Considering the five gestures of our dataset, it is possible to see that the recognition rate is not the same for the different classes.

If we look at the number of correctly detected and classified gestures per class, it is possible to see that the best run of uDeepGRU

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

First gesture detection results					
Method	**Corr. Class.**	**Mislab.**	**False P.**	**Missed**	**Avg.T1 Err.**
uDeepGRU$_2$	**85.2%**	7.4%	3%	4.4%	0.54s
uDeepGRU$_1$	79.3%	8,1%	3%	9.6%	0.55s
uDeepGRU$_3$	79.3%	8,1%	2.2%	10.4%	0.58s
SW 3-cent	75.6%	16.3%	2.2%	5.9%	0.48s
DeA	51.9%	18.5%	25.2%	4.4%	0.83s
AJ-RN	28.1%	43%	23%	5.9%	0.49s
PI-RN	11.1%	39.3%	48.9%	0.7%	0.56s
Seg. LSTM$_1$	11.1%	28.9%	60%	0%	2.1s
Seg. LSTM$_2$	6.7%	25.2%	68.1%	0%	2.61s

Table 1: *The table shows, for each method, the percentage of gestures detected in the correct time window and correctly classified, the percentage of gestures detected in the correct time window and mislabeled, the percentage of false positives encountered before the true gesture, the percentage of missed detections and the average error in estimating the initial mark T1. uDeepGRU$_{1,2}$ and $_3$ refer to the results provided by the relative authors and described in subsection 5.3.3. Seg. LSTM$_1$ and $_2$ refer to the results provided by the relative authors described in subsection 5.5.3.*

Method	**Avg. T2-End**	**Avg. T2-Start**
uDeepGRU$_2$	-1.66s	0.66s
uDeepGRU$_1$	-2.11s	0.21s
uDeepGRU$_3$	-1.87s	0.46s
SW 3-cent	-0.7s	1.61s
DeA	-1.31s	1.01s
AJ-RN	-1.65s	0.67s
PI-RN	-2.99s	-0.67s
Seg. LSTM$_1$	3.23s	5.56s
Seg. LSTM$_2$	-0.41s	1.91s

Table 2: *The table shows, for each method, the distance, in time, of the detection frame (i.e. T2) from the average end mark and the average start mark frame of a gesture.*

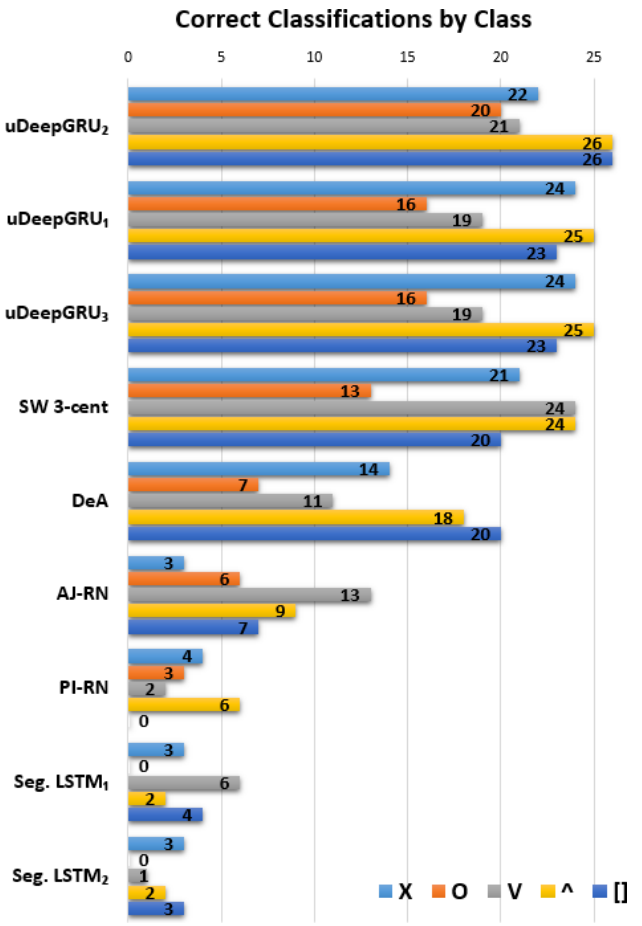

Figure 5: *Number of correct detection and labellings by class for each method. It is possible to see that the recognition of the different gestures is not constant. Circle gestures are more likely not correctly detected, while caret and square are the ones more easily recognized by uDeepGRU and DeA. V-gestures are quite accurately detected by SW-3cent.*

provided perfect results for caret and square gestures, so that an interface using these gestures could be quite effective. SW 3-cent also obtained sufficiently good results and it is the best one in the detection and classification of V-gestures. The other methods based on recurrent networks did not perform well. The reason may consist in the limited number of examples for training or in noise when the methods consider all the joints. It should be noted, however, that for AJ-RN and PI-RN the use of all the joints relevantly improves the classification accuracy.

In our case, the recognition task was hard due to the fact that gestures chosen were quite simple and that makes difficult to distinguish them from other hand actions classified as non-gesture.

Looking at the confusion matrices of the classification of the gestures detected in the correct time frames, it is possible to see that there are some ambiguities related to similar gestures (square and circle, cross and V). uDeepGRU is quite accurate but labeled as cross a V gesture a couple of time, sliding window 3-cent often labeled V gesture as a cross and a square gesture as a circle.

If we consider the difference between the decision time of the algorithms for the correct detections and the real start or end of the gestures, it is possible to see that uDeepGRU runs are also quite efficient in determining the correct gesture before the actual end of the gesture. Run 2 is the most accurate and also the one providing the earliest detection, predicting the correct gesture 2 seconds before the actual end on average. Almost all the methods perform the gesture detection early, and this fact is certainly quite positive for the practical use of the methods, allowing a smooth interaction and potentially allowing a time filtering of the decision output potentially increasing the detection/classification accuracy.

7. Conclusions

In this paper, we presented the results of the SHREC 2019 track on online gesture recognition. State of the art methods for trajectory-

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

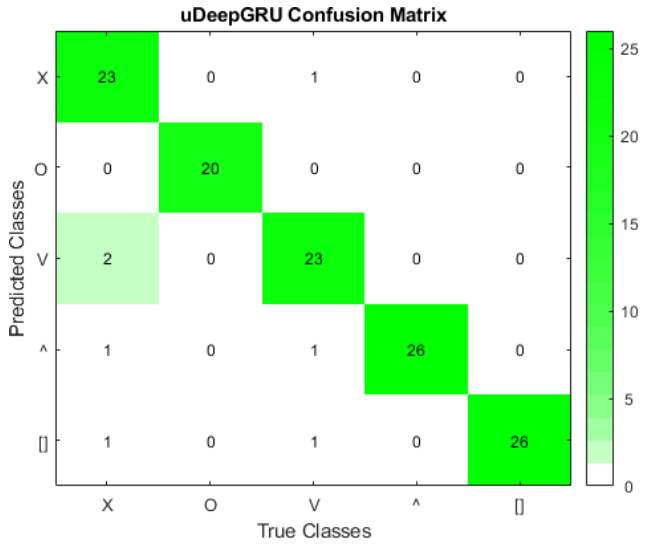

Figure 6: *Confusion matrix of the uDeepGRU$_2$ run, considering only the classification attempts in the correct time window.*

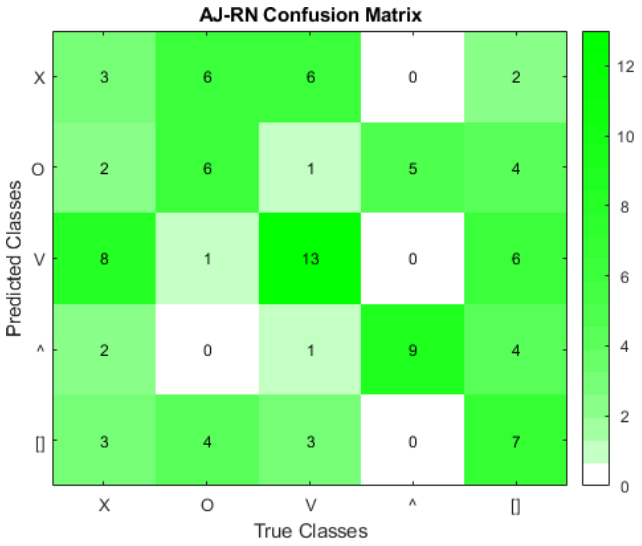

Figure 8: *Confusion matrix of the AJ-RN run, considering only the classification attempts in the correct time window.*

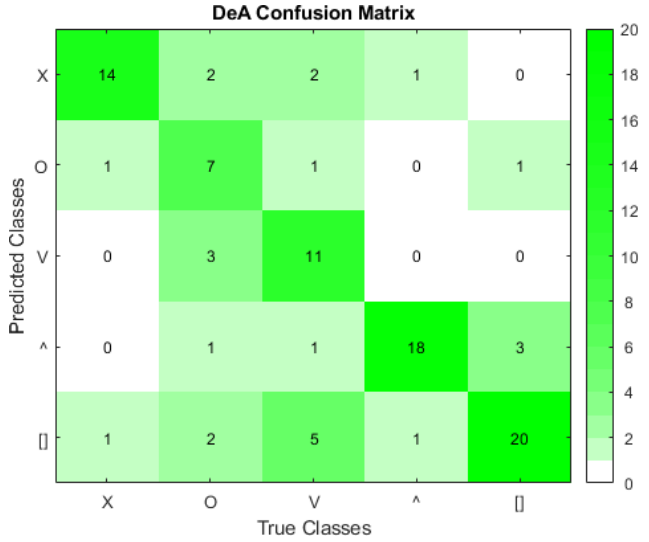

Figure 7: *Confusion matrix of the DeA run, considering only the classification attempts in the correct time window.*

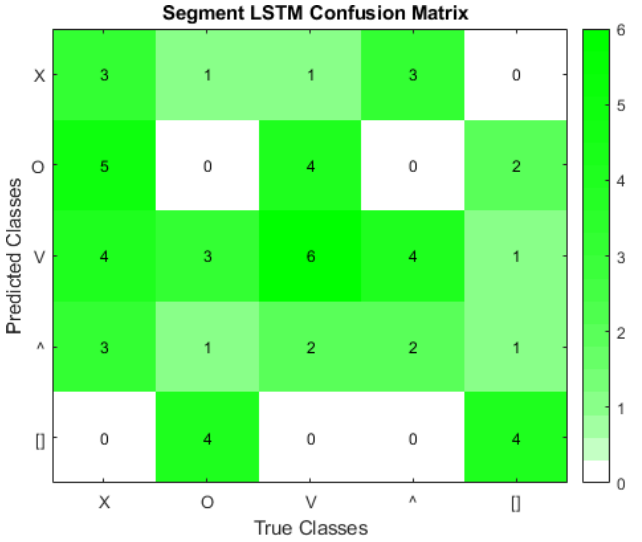

Figure 9: *Confusion matrix of the Segment LSTM$_1$ run, considering only the classification attempts in the correct time window.*

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

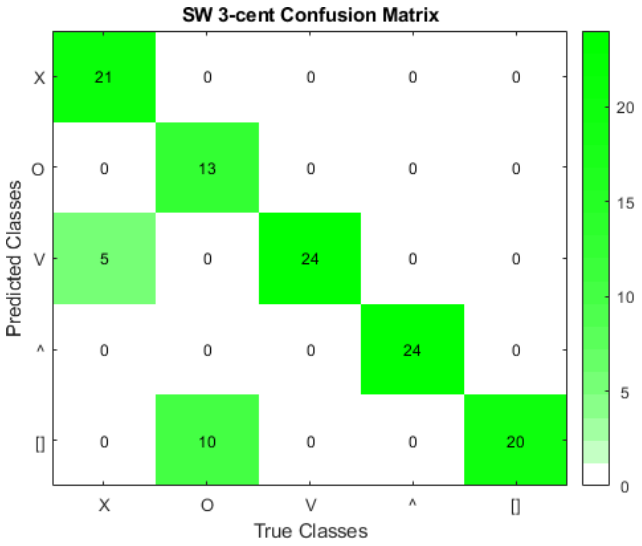

Figure 10: *Confusion matrix of the Sliding Window 3-cent run, considering only the classification attempts in the correct time window.*

based gesture recognition have been tested for online gesture recognition in a simple but realistic use case.

The potential use of simple hand gestures to control user interface can be quite useful to design interaction frameworks in VR/AR and remote control of smart devices.

We focused on the online recognition of very simple gestures, that can be characterized by single trajectories and should in principle distinguished by other movements that may occur during the interaction. Results show that recognition is feasible, even if the gesture dictionary proposed is quite simple, causing a relevant number of false detection. Both network-based methods and simple geometrical methods provided good detection and recognition performances and could be effective for building gestural interfaces.

While geometrical methods could be attractive for the simplicity and for the fact that can be easily used to design interfaces with novel sets of gestures, methods based on recurrent networks are clearly more scalable for the recognition of gestures based on hand articulation. However, the training procedures should be carefully designed to obtain optimal performances. This fact is also demonstrated by the method that provides the best results overall, uDeepGRU, where not only an effective network architecture has been employed, but also a smart data augmentation procedure, creating a large set of synthetic data from the small set of available examples.

The dataset created will be released with annotations in the contest webpage http://www.andreagiachetti.it/shrec19/ together with IR images acquired by the Leap Motion sensor.

We plan to test online gesture recognition methods on more complex datasets with different orientations and fingers' articulation in the near future. The goal is both to find optimal techniques and optimal gestures' dictionaries for user interfaces. Online gesture recognition will be also tested them within immersive VR and AR applications.

Acknowledgements

This work has been partially supported by project MIUR Excellence Departments 2018-2022.

References

[BCB15] BAHDANAU D., CHO K., BENGIO Y.: Neural machine translation by jointly learning to align and translate. In *Proc. ICLR* (2015). 4

[CPC*18] CAPUTO F. M., PREBIANCA P., CARCANGIU A., SPANO L. D., GIACHETTI A.: Comparing 3d trajectories for simple mid-air gesture recognition. *Computers & Graphics 73* (2018), 17–25. 1, 3

[CvMG*14] CHO K., VAN MERRIËNBOER B., GÜLÇEHRE Ç., BAHDANAU D., BOUGARES F., SCHWENK H., BENGIO Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 1724–1734. URL: http://www.aclweb.org/anthology/D14-1179. 4

[CWG*19] CHEN X., WANG G., GUO H., ZHANG C., WANG H., ZHANG L.: Mfa-net: Motion feature augmented network for dynamic hand gesture recognition from skeletal data. *Sensors 19*, 2 (2019), 239. 1

[DAHG*15] DONAHUE J., ANNE HENDRICKS L., GUADARRAMA S., ROHRBACH M., VENUGOPALAN S., SAENKO K., DARRELL T.: Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 2625–2634. 5

[DMXY18] DEVINEAU G., MOUTARDE F., XI W., YANG J.: Deep learning for hand gesture recognition on skeletal data. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)* (2018), IEEE, pp. 106–113. 1

[DS17] DE SMEDT Q.: *Dynamic hand gesture recognition-From traditional handcrafted to recent deep learning approaches*. Université de Lille 1, Sciences et Technologies; CRIStAL UMR 9189., 2017. 3

[GCC*16] GIACHETTI A., CAPUTO F. M., CARCANGIU A., SCATENI R., SPANO L. D.: Shape retrieval and 3d gestural interaction: position paper. In *Proceedings of the Eurographics 2016 Workshop on 3D Object Retrieval* (2016), Eurographics Association, pp. 1–4. 1

[HS97] HOCHREITER S., SCHMIDHUBER J.: Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780. 5

[HSK*12] HINTON G. E., SRIVASTAVA N., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R. R.: Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012). 4

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. pp. 448–456. URL: http://jmlr.org/proceedings/papers/v37/ioffe15.pdf. 4

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 5

[KFF15] KARPATHY A., FEI-FEI L.: Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3128–3137. 5

[MLJ18] MAGHOUMI M., LAVIOLA JR J. J.: DeepGRU: Deep Gesture Recognition Utility. *arXiv preprint arXiv:1810.12514* (2018). 1, 4

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

[PGC*17] PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E., DEVITO Z., LIN Z., DESMAISON A., ANTIGA L., LERER A.: Automatic differentiation in pytorch. In *NIPS-W* (2017). 5

[SHK*14] SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R.: Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958. 5

[SWV*17] SMEDT Q. D., WANNOUS H., VANDEBORRE J.-P., GUERRY J., SAUX B. L., FILLIAT D.: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. In *Eurographics Workshop on 3D Object Retrieval* (2017), Pratikakis I., Dupont F., Ovsjanikov M., (Eds.), The Eurographics Association. doi:10.2312/3dor.20171049. 2

[TIMPLJ16] TARANTA II E. M., MAGHOUMI M., PITTMAN C. R., LAVIOLA JR J. J.: A rapid prototyping approach to synthetic data generation for improved 2d gesture recognition. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (2016), ACM, pp. 873–885. 4

[VAW18] VATAVU R.-D., ANTHONY L., WOBBROCK J. O.: $q: a super-quick, articulation-invariant stroke-gesture recognizer for low-resource devices. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2018), ACM, p. 23. 1

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.