Dynamic Hand Gesture Recognition for Mobile Systems Using Deep LSTM

Ayanava Sarkar[1], Alexander Gepperth[2], Uwe Handmann[3], Thomas Kopinski[4]

[1]Computer Science Department, Birla Institute of Technology and Science, Pilani, Dubai Campus, UAE
[2]Computer Science Department, University of Applied Sciences Fulda, Fulda, Germany
[3]Computer Science Department, University of Applied Sciences Ruhr West, Germany
[4]Computer Science Department, University of Applied Sciences South Westphalia, Germany

Abstract. We present a pipeline for recognizing dynamic freehand gestures on mobile devices based on extracting depth information coming from a single Time-of-Flight sensor. Hand gestures are recorded with a mobile 3D sensor, transformed frame by frame into an appropriate 3D descriptor and fed into a deep LSTM network for recognition purposes. LSTM being a recurrent neural model, it is uniquely suited for classifying explicitly time-dependent data such as hand gestures. For training and testing purposes, we create a small database of four hand gesture classes, each comprising 40 × 150 3D frames. We conduct experiments concerning execution speed on a mobile device, generalization capability as a function of network topology, and classification ability 'ahead of time', i.e., when the gesture is not yet completed. Recognition rates are high (>95%) and maintainable in real-time as a single classification step requires less than 1ms computation time, introducing freehand gestures for mobile systems.

Keywords: Mobile Computing, Gestural Interaction, Deep Learning

1 Introduction

Gestures are a well-known means of interaction on mobile devices such as smart phones or tablets up to the point that their usability is so well-integrated into the interface between man and machine that their absence would be unthinkable. However, this can only be stated for touch gestures as three-dimensional or freehand gestures have to yet find their way as a means of interaction into our everyday lives. While freehand gestures are steadily being included as an additional means of control in different various fields (entertainment industry, infotainment systems in cars), within the domain of mobile devices a number of limitations present obstacles to be overcome in order to make this an unequivocally seamless interaction technique.

First and foremost, data has to be collected be in an unobtrusive manner, hence no sensors attached to the user's body can be utilized. As mobile devices have to remain operable independent of the user's location the number of employable technologies is drastically reduced. Eligible sensor technology is mainly limited to Time-of-Flight (TOF) technology as it is not only capable to provide surrounding information independent of the background illumination but moreover can do so at high frame rates. This is the presupposition to realize an interface incorporating freehand gesture control as it allows for the system's reaction times to remain at a minimum. TOF technology has to yet be established as a standard component in mobile devices (as e.g. in the Lenovo PHAB2 Pro) and it moreover suffers from a comparatively small resolution, potentially high noise and heat development. Despite these drawbacks it is a viable choice since the benefits outweigh the disadvantages as will be presented in this contribution. Realizing freehand gestures as an additional means of control not only overcomes problems such as usage of gloves or the occlusion of the screen interface during touch gesture interaction. It moreover also allows for increased expressiveness (with additional degrees of freedom) which in turns allows for a completely new domain of novel applications to be developed (especially in the mobile domain). This can be corroborated by the fact that car manufacturers, which have always been boosting innovations by integrating new technologies into the vehicle, have recently begun incorporating freehand gestures into the vehicle interior (e.g. BMW, VW etc.). The automotive environment faces the same problems such as stark illumination variances, but on the other hand can compensate difficulties such as high power consumption.

In this contribution we present a light-weight approach to demonstrate how dynamical hand gesture recognition can be achieved on mobile devices. We collect data from a small TOF sensor attached to a tablet. Machine Learning models are created by training from a dynamic hand gesture data base. These models are in turn used to realize a dynamic hand gesture recognition interface capable of detecting gestures in real-time.

The approach presented in this contribution can be set apart from other work in the field of Human Activity Recognition (HAR) by the following aspects: We utilize a single TOF camera in order to retrieve raw depth information from the surrounding environment. This allows for high frame rate recordings of nearby interaction while simultaneously making the retrieved data more robust vs. nearby illumination changes. Moreover, our approach is viable using only this single sensor, in contrast to other methodology where data coming from various kinds of sources is fused. Furthermore, data acquired in a non-intrusive manner allows for full expressiveness in contrast to data coming from sensors attached to the user's body. The process as a whole is feasible and realizable in real-time insofar as that once the model is generated after training, it can be simply transferred onto a mobile device and utilized with no negative impact on the device's performance. The remaining sections are organized as follows: Work presented in this contribution is contrasted to state of the art methodology within the domain of dynamic freehand gesture recognition (Sec. 1.1). The Machine Learning models are trained on a database described in Section 2.1. Data sample/s are transformed and presented to the LSTM models in the manner outlined in Section 2.2. The LSTM models along with the relevant parameters are subsequently explained in Section 2.3. The experiments implemented in this contribution are laid out in Section 3 along with the description of the parameter search (Sec. 3.1 ff.) and model accuracy (Sec. 3.3). The resulting hand gesture demonstrator is explained in Section 5 along with an explanation of its applicability. Section 6 sums up this contribution as a whole and provides a critical reflection on open questions along with an outlook on upcoming future work.

1.1 Dynamic Hand Gesture Detection - An overview

Recurrent Neural Networks (RNNs) are employed for gesture detection by fusing inputs coming from raw depth data, skeleton information and audio information [11]. Recall (0.87) and Precision rates (0.89) peak, as expected, when information is fused from all three channels. The authors of [12] present DeepConvLSTM, a deep architecture fusing convolutional layers and recurrent layers from an LSTM for Human Activity Recognition (HAR). Data is provided by attaching several sensors to the human body and therewith extracting accelerometric, gyroscopic and magnetic information. Again, recognition accuracy improves strongly as more data is fused. Their approach demonstrates how HAR can be improved with the utilization of LSTM as CNNs seem not to be able to model temporal information on their own. The authors of [13] utilize BLSTM-RNNs to recognize dynamic hand gestures and compare this approach to standard techniques. However, again body-attached sensors are employed to extract movement information and results are comparatively low regarding the fact that little noise is present during information extraction. No information is given with regard to execution time raising the question of real-time applicability.

2 Methods

2.1 The Hand Gesture Database

Data is collected from a TOF sensor at a resolution of 320x160 pixels. Depth thresholding removes most of the irrelevant background information, leaving only hand and arm voxels. Principal-Component Analysis (PCA) is utilized to crop most of the negligible arm parts. The remaining part of the point cloud carries the relevant information, i.e., the shape of the hand. Fig. 1 shows the color-coded snapshot of a hand posture.

We recorded four different hand gestures from a single person at one location for our database: close hand, open hand, pinch-in and pinch-out. The latter gestures are performed by closing/opening two fingers. For a single dynamic gesture recording, 40 consecutive snapshots (no segmentation or sub-sampling) are taken from the sensor and cropped by the aforementioned procedure. In this manner, 150 gesture samples at 40 frames per gesture are present per class in the database, summing up to a total of 24.000 data samples.

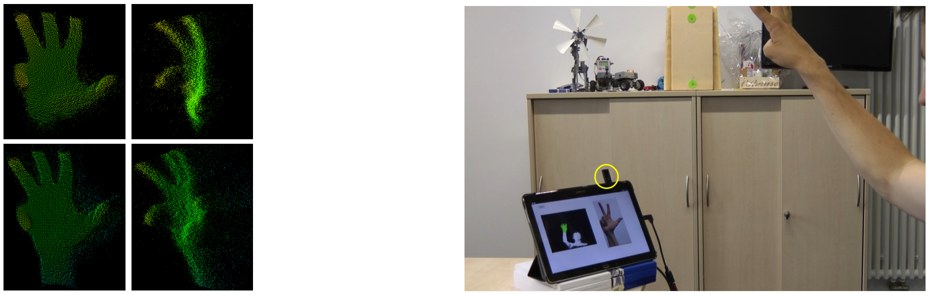


Fig. 1. Data and data generation. Left: Sample snapshot of a resulting point cloud after cropping from the front (left) and side view (right) during a grabbing motion. The lower snapshot describes the hand's movement for each viewpoint (left and right respectively). Right: The Setup - tablet with a picoflexx (indicated with yellow circle).

2.2 From point clouds to network input

Description of a point cloud usually is implemented by so-called descriptors which, in our case, need to describe the phenomenology of hand, palm and fingers in a precise manner at a certain point in time. The possibilities of describing point cloud data are confined to either utilizing some form of convexity measure or calculating the normals for all points in a cloud. Either way, it has to remain computationally feasible in order to maintain real-time capability. In this contribution, the latter methodology is implemented: for a single point cloud, the normals for all points are calculated. Then, for two randomly selected points in a cloud, the PFH metric is calculated [14,15]. This procedure is repeated for up to 5000 randomly selected point pairs extracted from the cloud. Each computation results in a descriptive value which in turn is binned into a 625-dimensional histogram. Therefore, one such histogram provides a description of a single point cloud snapshot at a single point in time. These histograms form the input for training and testing the LSTM models.

2.3 LSTM model for gesture recognition

In our model for dealing with the video frames sequentially, we use a deep RNN with LSTM model neurons, where the LSTM term for neurons is ”memory cell” and the term for hidden layer is ”memory cell”. At the core of each memory cell is a linear unit supported by a single self-recurrent connection whose weight is initialized to 1.0. Thus, in the absence of any other input, this self-connection serves to preserve the cell's current state from one moment to the next. In addition to the self-recurrent connection, cells also receive input from input units and other cell and gates. The key component of a LSTM cell inside the memory block is its cell state, referred to as C_t or the cell state at time step t. This cell state remains unique for a cell and any change to the cell state is done with the help of gates - input gate, output gate and the forget gate. The output of the gates is a value between 0 and 1, with 0 signifying not ”let anything through the gate” and 1 signifying ”let everything through the gate”. The input gate determines how much of the input to be forwarded to the cell, then the forget gate calculates how much of the cell’s previous state to keep depending on how much to let the input affect the cell state, thus, the extent to which a value remains in the cell state and finally, the output gate computes the output activation, thereby, determining how much of the activation of the cell to be output.

At a time step t, the input to the network is x_t and $h_{t\text{-}1}$, where the former is the input and the latter is the output at time step t-1. For the first time step, the $h_{t\text{-}1}$ is taken to be 1.0. In the hidden layers or the memory blocks, the output of one memory block forms the input to the next block. The following are the equations revolving around the inner complexities of an LSTM model, where W refers to the weights, b refers to the biases and the σ refers to the sigmoidal function, outputting a value between 0 and 1:

$$i_{\mathrm{t}} = \sigma(W_{\mathrm{ix}} x_{\mathrm{t}} + W_{\mathrm{ih}} h_{\mathrm{t\text{-}1}} + b_{\mathrm{i}}) \tag{1}$$

Equation 1 refers to the calculation of the input gate. Final output of the input gate is a value between 0 and 1.

$$f_{\mathrm{t}} = \sigma(W_{\mathrm{fx}} x_{\mathrm{t}} + W_{\mathrm{fh}} h_{\mathrm{t\text{-}1}} + b_{\mathrm{f}}) \tag{2}$$

Equation 2 refers to the calculation of the forget gate. Final output of the forget gate is a value between 0 and 1.

$$o_{\mathrm{t}} = \sigma(W_{\mathrm{ox}} x_{\mathrm{t}} + W_{\mathrm{oh}} h_{\mathrm{t\text{-}1}} + b_{\mathrm{o}}) \tag{3}$$

Equation 3 refers to the calculation of the output gate. Final output of the output gate is a value between 0 and 1.

$$g_{\mathrm{t}} = \tanh(W_{\mathrm{gx}} x_{\mathrm{t}} + W_{\mathrm{gh}} h_{\mathrm{t\text{-}1}} + b_{\mathrm{g}}) \tag{4}$$

Equation 4 refers to the calculation of $\mathrm{g_t}$ that gives a value between -1 and 1, specifying the amount of importance of the input that is relevant to the cell state, where the tanh function outputs a value between -1 and 1. Here, $\mathrm{g_t}$ refers to the new candidate values that must be added to the existing or the previous cell state.

$$c_{\mathrm{t}} = f_{\mathrm{t}} c_{\mathrm{t\text{-}1}} + i_{\mathrm{t}} g_{\mathrm{t}} \tag{5}$$

Equation 5 refers to the calculation of the new cell state, replacing the old one.

$$h_{\mathrm{t}} = \tanh(c_{\mathrm{t}}) o_{\mathrm{t}} \tag{6}$$

Equation 6 refers to the calculation of the hidden state or the output of that particular memory block, which then serves as the input to the next memory block. The tanh function allows it to output a value between -1 and 1. Further information about these equations can be found in [3].

The final output of the LSTM network is produced by applying a linear regression readout layer that transforms the states C_t of the last hidden layer into class membership estimates, using the standard softmax non-linearity leading to positive, normalized class membership estimates.

3 Experiments and Observations

The implementation has been done in TensorFlow using Python. There is a total 150 video files for each of the 4 classes of hand gestures. The model is trained on $N_{tr} = 480$ total samples, with 120 samples belonging to each of the 4 classes of hand gestures. The model is then evaluated using a total of $N_{te} = 120$ samples, with 30 samples belonging to each of the 4 classifying classes. The three parts of the experiment adhere to this partitioning of the data. In our implementation, each gesture is represented by a tensor of 40x625 numbers, while the input of the deep LSTM network corresponds to the dimension of a single frame, that is 625 numbers.

3.1 Model Parameters

Network training was conducted using the standard tools provided by the TensorFlow package, namely the Adam optimization algorithm [9,10]. Since the performance of our deep LSTM network depends strongly on network topology and the precise manner of conducting the training, we performed a search procedure by varying the principal parameters involved here. These are given in Tab.1, as well as the range in which they were varied.

symbol	meaning	variability
B	Batch Size	2,5,10
M	# of Memory Blocks (MB)	1-4
C	# of LSTM Cells per MB	128,256,512
I	SGD training iterations	A=100,B=500,C=1000
η	Learning Rate	0.1, 0.001, 0.0001, 0.00001

Table 1. Principal parameter for network topology and training. The last column indicated the range of values that were exhaustively tested for these parameters.

3.2 Deep LSTM parameter search

Initially with $B = 2$, 5, 10, M is varied from 1 to 4 for each value of B, C is varied with 128, 256 and 512 for each value of B and M, and I has been varied between 100, 500 and 1000 for each value of the other three parameters. The learning rate is kept constant at 0.0001. Thus, for all combinations of the B, M, C and I, a total of 108 experiments has been carried out.

Now let the predictions for each sample data entered into the model be denoted by $\boldsymbol{P_i}$, where $\boldsymbol{i}$ refers to the index of the sample data in the test data. $\boldsymbol{P_i}$ is calculated for all frames of a test sample i, where the prediction obtained at the last frame defines $\boldsymbol{P_i}$. It is also possible to consider $\boldsymbol{P_i}$ for frames ¡ 40, achieving ahead-of-time guesses at the price of potentially reduced accuracy. $\boldsymbol{P_i}$ is a vector of length 4, since there are 4 classes for classification in the experiment. We take the argmax of these 4 elements to indicate the predicted class as shown in Equation 7. Now to test if the prediction is correct or not it is compared with the label of the data sample, l_i.

$$\tilde{p}_i = \mathrm{argmax}(\boldsymbol{P_i}) \tag{7}$$

$$\xi = 100\frac{\#(\tilde{p}_i = l_i)}{N_{te}} \tag{8}$$

Equation 8 refers to the simple formula used for calculating the accuracy.

3.3 Measuring accuracy as a function of observation time

In the second part of our experimentation, we train the model similar to Sec. 3.2, however in the testing phase, we calculate the predictions at different in-gesture time steps (frames) t. Let $\boldsymbol{P_{i,t}}$ denote the prediction for sample i at $t < 40$. In order to obtain an understanding of how the prediction varies more frames are processed, we calculate the predictions $\boldsymbol{P_{i,t}}$ at time steps $t = \{10, 20, 25, 30, 39, 40\}$. Here, we perform class-wise analysis to determine which classes lend themselves best to ahead-of-time ”guessing” which can be very important in practice.

3.4 Speedup and Optimization of the Model

The implementation shown so far is focused on accuracy alone. Since mobile devices in particular lack faster and more capable processing units, the aim of this part of the article is to speed-up gesture recognition as much as possible by simplifying the LSTM model, if possible without compromising its accuracy. To this end, B has been kept constant at 2, while M is taken to be 1 in all the experiments. The number of memory cells in the single memory block is taken as either 8 or 10. Now, with such a small network, we are able to greatly speed up the system as well as minimize the computation complexities involved regarding the entire model.

4 Experimental results

4.1 Deep LSTM parameter search

With the 108 experiments conducted by varying B, M, C and I, 20 accuracies have been reported in Tab.2, with the idea of covering the diversity of the experimental setup of 108 experiments.

B	2	5	10	10	5	2	10	5	5	10	10	5	2	5	2	5	5	2
M	1	1	4	3	2	1	3	2	4	1	2	1	4	4	2	4	2	1
C	512	256	128	512	128	256	256	512	128	128	128	512	128	512	512	256	128	128
I	C	C	B	B	B	C	B	C	C	B	C	C	C	B	C	C	C	C
ξ	100	96.7	100	98.3	100	95	96.7	96.7	100	97.5	99.2	100	100	99.2	100	100	95.8	96.7

Table 2. Results for exhaustive parameter search in topology space. In total, we conducted 108 experiments by varying the network topology and training parameters. The best 18 results are shown here. The column headings correspond to the symbols defined in Tab. 1.

From the observations, it can be concluded that for a given M, C and I, the accuracy improves with the increase in the value of B. Thus, $B = 10$ will have a greater accuracy on the test data as compared to $B = 2$ or $B = 5$. This can be explained by the fact that for a given I, the model undergoes a total of (I X B) times of training in this experimental setup. Thus, as the number of B increases, so does the value (I X B) and consequently the accuracy of prediction. Now, for a given B, C and I, if M is varied between 1 to 4, it has been observed that with the increase in the number of hidden layers or M, the accuracy of prediction improves significantly. This is because, as the number of layers increases, the network becomes more complex with the ability to take into account more complex features from the data and hence, account for more accurate predictions. Similarly, when keeping B, M and I constant and varying C between 128, 256 and 512, we observe that accuracy increases with the increase in the number of memory cells in each memory block, thereby bearing a directly proportional relationship. Similar results were observed when I is varied, keeping B, M and C as constant parameters, which can be explained by the fact that the model has more time or iterations to adjust its weight in order to bring about the correct prediction. Further, Tab.2 shows the accuracies for the different combinations of the network parameters.

4.2 Measuring accuracy as a function of observation time

In this part we calculate different quality measures as a function of the frame t they are obtained. The graph in Fig. 2 shows that as the number of time steps increases, the accuracy increases until the maximum accuracy is reached in the last 5 time steps. Furthermore, we can also evaluate the confidence of each classification: as classification of test sample i is performed by taking the argmax of the network output $\boldsymbol{P_i}$, the confidence of this classification is related to max $\boldsymbol{P_i}$. We might expect that the confidence of classification increases with $t < 40$ as well as more frames have been processed for higher t. Now, Fig. 3 and Fig. 3 depicts the average maxima plus standard deviations (measured on test data) as a function of their class. We observe that, in total coherence to the increase in accuracy over in-gesture time t, the certainty of predictions increases as well, although we observe that this is strongly depending on the individual classes, reflecting that some classes are less ambiguous than others.

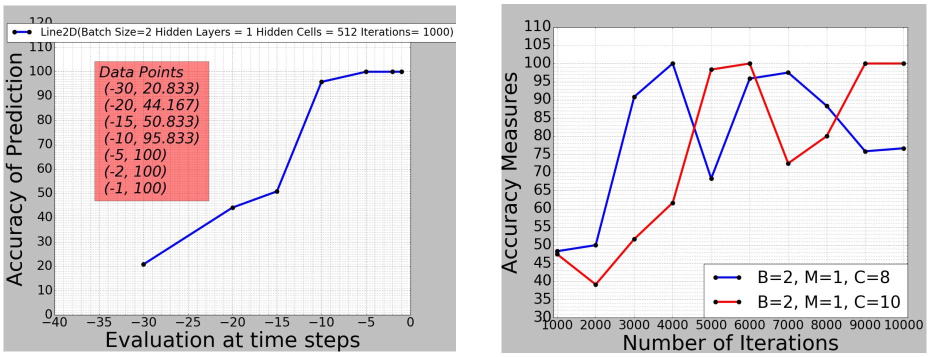
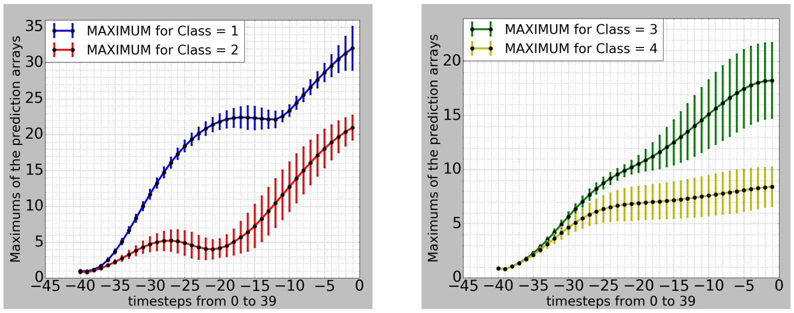


Fig. 2. Left: accuracy of prediction of a single test data sample, with B = 2, M = 1, C = 512 and I = 1000, at different in-gesture time steps t. Right: accuracy of prediction (taken at the end of a gesture) depending on training iterations for a small LSTM network size.

4.3 Speedup and Optimization of the Model

We observe that as the size of the network is greatly reduced comprising a single memory block and the number of memory cells being either 8 or 10, the accuracy is not as great as observed in Sec. 4.1. Hence, in order to accomplish the same level of accuracy as obtained in Sec. 4.1, the number of iterations for the training process was increased. The performances can be referred to in Fig. 2, showing that 100% accuracy can be achieved even with small networks, although training time (and thus the risk of overfitting) increases strongly.



(a) Average and standard deviations of prediction maxima plotted against in-gesture time for classes 1 and 2

(b) Average and standard deviations of prediction maxima plotted against in-gesture time for classes 3 and 4

Fig. 3. "Ahead of time" classification accuracy for classes 1 and 2 (left) as well as 3 and 4 (right).

5 System Demonstrator

5.1 Hardware

The system setup consists of a Galaxy Notepro 12.2 Tablet running Android 5.02. A picoflexx TOF sensor from PMD technologies is attached to the tablet via USB. It has an IRS1145C Infineon 3D Image Sensor IC chip based on pmd intelligence which is capable of capturing depth images with up to 45 fps. VC-SEL illumination at 850nm allows for depth measurements to be realized within a range of up to 4m, however the measurement errors increase with the distance of the objects to the camera therefore it is best suited for near-range interaction applications of up to 1m. The lateral resolution of the camera is 224×171 resulting in 38304 voxels per recorded point cloud. The depth resolution of the picoflexx depends on the distance and with reference to the manufacturer's specifications is listed as 1% of the distance within a range of 0.5 - 4m at 5fps and 2% of the distance within a range of 0.1 - 1m at 45fps. Depth measurements utilizing ToF technology require several sampling steps to be taken in order to reduce noise and increase precision. As the camera allows several pre-set modes with a different number of sampling steps we opt for 8 sampling steps taken per frame as this resulted in the best performance of the camera with the lowest signal-to-noise ratio. This was determined empirically in line with the positioning of the device. Several possible angles and locations for positioning the camera are thinkable due to its small dimensions of $68mm \times 17mm \times 7.25mm$. As we want to setup a demonstrator to validate our concept the exact position of the camera is not the most important factor however should reflect a realistic setup. In our situation we opted for placing it at the top right corner when the tablet is placed in a horizontal position on the table. However, it should be stated here that any other positioning of the camera would work just as well for the demonstration presented in this contribution.

5.2 System Performance

One classification step of our model takes about [1.6e-05, 3.8e-05] of computation time (in s). As Figure 4 indicates, the time required to crop the cloud to its relevant parts is linearly dependent on the number of points within the cloud.

This is the main bottleneck of our approach as all other steps within the pipeline are either constant factors or negligible w.r.t. computation time required. During real-time tests our systems achieved frame rates of up to 40 fps.

6 Conclusion

We presented a system for real-time hand gesture recognition capable of running in real time on a mobile device, using a 3D sensor optimized for mobile use. Based on a small database recorded using this setup, we prove that high speed and an excellent generalization capacity are achieved by our combined preprocessing+deep RNN-LSTM approach. As LSTM is a recurrent neural network

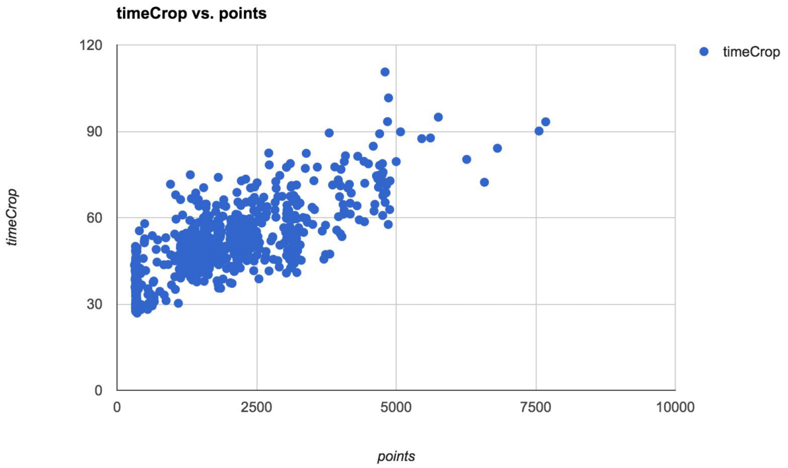


Fig. 4. Graph plotting the time required to crop the hand and reduce the number of relevant voxels with respect to the number of total points in the cloud.

model, it can be trained on gesture data in a straightforward fashion, requiring no segmentation of the gesture, just the assumption of a maximal duration corresponding to 40 frames. The preprocessed signals are fed into the network frame by frame, which has the additional advantage that correct classification is often achieved before the gesture is completed. This might make it possible to have an ”educated guess” about the gesture being performed very early on, leading to more natural interaction, in the same way that humans can anticipate the reactions or statements of conversation partners. In this classification problem, it is easy to see why ”ahead of time” recognition might be possible as the gestures differ sufficiently from each other from a certain point in time onwards.

A weak point of our investigation is the small size of the gesture database which is currently being constructed. While this makes the achieved accuracies a little less convincing, it is nevertheless clear that the proposed approach is basically feasible, since multiple cross-validation steps using different train/test subdivisions always gave similar results. Future work will include performance tests on several mobile devices and corresponding optimization of the used algorithms (i.e., tune deep LSTM for speed rather than for accuracy), so that 3D hand gesture recognition will become a mode of interaction accessible to the greatest possible number of mobile devices.

References

1. Hochreiter, Sepp. ”The vanishing gradient problem during learning recurrent neural nets and problem solutions.” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998): 107-116.

2. Venugopalan, Subhashini, et al. "Translating videos to natural language using deep recurrent neural networks." *arXiv preprint arXiv:1412.4729* (2014).
3. Hochreiter, Sepp and Schmidhuber, Jürgen. "Long short-term memory." *Neural computation* (1997): 1735-1780.
4. Bengio, Yoshua and Simard, Patrice and Frasconi, Paolo. "Learning long-term dependencies with gradient descent is difficult." *IEEE transactions on neural networks 5.2.* (1994): 157-166.
5. Mikolov, Tomas, et al. "Recurrent neural network based language model." *Interspeech.* Vol. 2. 2010.
6. Graves, Alex. "Generating sequences with recurrent neural networks." *arXiv preprint arXiv:1308.0850.* (2013).
7. Graves, Alex and Abdel-Rahman, Mohamed and Hinton, Geoffrey. "Speech recognition with deep recurrent neural networks." *Acoustics, speech and signal processing (ICASSP), 2013 ieee international conference on.* IEEE, 2013.
8. Karpathy, Andrej and Fei-Fei, Li. "Deep visual-semantic alignments for generating image descriptions." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition..* 2015.
9. Kingma, Diederik and Ba, Jimmy. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980.* (2014).
10. Bengio, Yoshua. "Practical recommendations for gradient-based training of deep architectures." *Neural networks: Tricks of the trade..* Springer Berlin Heidelberg, 2012. 437-478.
11. Neverova, Natalia, et al. "A multi-scale approach to gesture detection and recognition." Proceedings of the IEEE International Conference on Computer Vision Workshops. 2013.
12. Ordóñez, Francisco Javier and Roggen, Daniel. "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition." Sensors 16.1 (2016): 115.
13. Lefebvre, Grégoire, et al. "BLSTM-RNN based 3D gesture classification." International Conference on Artificial Neural Networks. Springer Berlin Heidelberg, 2013.
14. Rusu, Radu Bogdan, et al. "Aligning point cloud views using persistent feature histograms." I ntelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008.
15. Caron, Louis-Charles, David Filliat, and Alexander Gepperth. "Neural network fusion of color, depth and location for object instance recognition on a mobile robot." Second Workshop on Assistive Computer Vision and Robotics (ACVR), in conjunction with European Conference on Computer Vision. 2014.