A Deep Learning Approach to Mid-air Gesture Interaction for Mobile Devices from Time-of-Flight Data

Thomas Kopinski
ENSTA ParisTech
858 Blvd des Maréchaux
Palaiseau, France
thomas.kopinski@ensta-paristech.fr

Fabian Sachara
Hochschule Ruhr West,
Institut Informatik
Lützowstrasse 5
46236, Bottrop, Germany
fabian.sachara@hs-rw.de

Uwe Handmann
Hochschule Ruhr West
Institut Informatik
Lützowstrasse 5
46236, Bottrop, Germany
uwe.handmann@hs-rw.de

ABSTRACT

This contribution presents a novel approach of utilizing Time-of-Flight (ToF) technology for mid-air hand gesture recognition on mobile devices. ToF sensors are capable of providing depth data at high frame rates independent of illumination making any kind of application possible for in- and outdoor situations. This comes at the cost of precision regarding depth measurements and comparatively low lateral resolution. We present a novel feature generation technique based on a rasterization of the point clouds which realizes fixed-sized input making Deep Learning approaches applicable using Convolutional Neural Networks. In order to increase precision we introduce several methods to reduce noise and normalize the input to overcome difficulties in scaling. Backed by a large-scale database of about half a million data samples taken from different individuals our contribution shows how hand gesture recognition is realizable on commodity tablets in real-time at frame rates of up to 17Hz. A leave-one out cross-validation experiment demonstrates the feasibility of our approach with classification errors as low as 1,5% achieved persons unknown to the model.

CCS Concepts

•**Human-centered computing** → *Ubiquitous and mobile computing systems and tools;*

Keywords

Deep Learning, mid-air gestures, Object Recognition

1. INTRODUCTION

The benefit of adding depth sensing to mobile devices is sometimes questioned as various difficulties have to be overcome in order to meet satisfying demands. First and foremost, any 3D sensing device would have to compete with already present hardware, able to capture data in real-time and make it processable for interesting applications. With Google pushing its 'Project Tango' and the simultaneous advancement of depth sensing technology for smartphones, the prospect of developing useful applications capable of harnessing all of the device's potential has come a step closer. One of the first devices about to be released for the consumer market is the PHAB 2 Pro with its specifically designed Time-of-Flight (ToF) module IRS1645C. According to the manufacturer, this is the only chipset integrating the necessary pixel matrix, activation, analog-to-digital converter (ADC) and interface on one chip.

This in turn means, there is little to no work dealing with ToF technology for mobile devices. While the main drawback is its imprecision, stemming mainly from material-dependent factors such as the reflection coefficient of objects, the noise occurring from ToF measurements can be reduced, if the parameters are chosen properly for the application in mind, allowing for illumination-invariant apps to be realized at high frame rates. The main benefit of ToF technology is indeed its capability of providing fast, reliable data in- and outdoors - a feature highly desirable for any HMI context on mobile devices. We present a novel contribution in this field of research, coupling ToF technology with mobile devices to demonstrate how a sophisticated data processing pipeline makes 3D depth perception processable, allowing for mid-air hand gesture recognition and interaction to be realized in real-time. The core of our setup is a novel data transformation technique in order to be able to train a deep convolutional neural network (CNN). Our setup constantly achieves frames rates between 14-17Hz, more than sufficient for any real-time applications to be realized on mobile devices. The rest of this contribution is laid out as follows: In Section 2 we start by giving an overview over the most important research conducted at the crossing of the disciplines of ToF-based sensing, Deep Learning, Mobile Computing and Hand Gesture Recognition. We go on to describe the setup of our system by describing the hardware being put to use along with an overview over the realized recognition pipeline with its individual modules (Section 3). The backbone of our system is a large-scale hand gesture database which we outline in Section 4 with respect to the most relevant aspects for this contribution. The main module is the feature generation which is described in Section 5. The results of our cross-validation tests are then described in Section 6. We evaluate the system's performance in terms of computation complexity in Section 7 and conclude this contribution with a summary of the most important results and an outlook

MOBIQUITOUS '16, November 28-December 01, 2016, Hiroshima, Japan
© 2016 ACM. ISBN 978-1-4503-4750-1/16/11...$15.00
DOI: http://dx.doi.org/10.1145/2994374.2994392

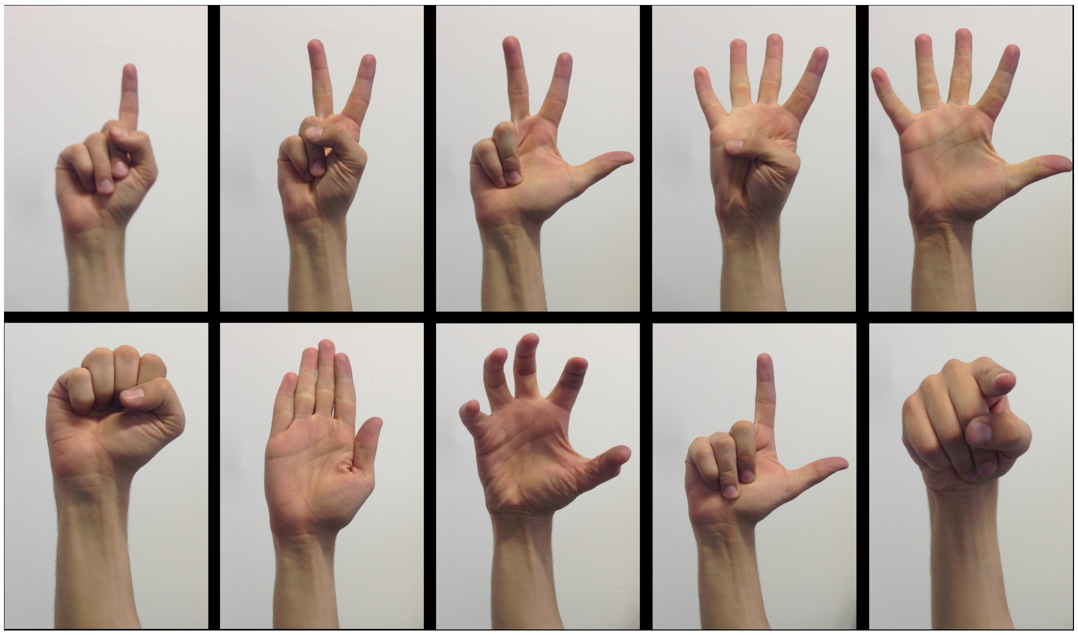

Figure 1: The hand posture database. From left to right, top to bottom: *ONE*, *TWO*, *THREE*, *FOUR*, *FIVE*, *FIST*, *FLAT*, *GRAB*, *PINCH*, *POINT*

(Section 8).

2. RELATED WORK

There is a large body of work on hand gesture recognition utilizing solely ToF sensors [3], [7], [8], [11], or coupling them with RGB information [23], [18], [2], [22]. Suarez et al. [21] provide a good overview over the common approaches within this complex research field. Adding RGB information to depth data adds the benefit, along with increased data robustness, of simple object segmentation via e.g. skin detection. However, one evidently loses the main advantage of having an illumination-invariant scenario which only persists if make use of ToF data in this case. Much of the research conducted in the recent years was pushed with the influx in availability of low-cost commodity hardware, mainly driven with the release of the Kinect to the consumer market. However, its depth sensing technology itself is not suited to be utilized in outdoor scenarios. There is a plethora of possibilities to categorize the various approaches from either the hardware perspective (sensors, invasive vs. non-invasive tech etc.), or the software perspective with emphasis put on Machine Learning (ML) algorithms, algorithm design or even HMI-related aspects such as applications or interface-related questions. When confining oneself to ML aspects, numerous approaches have been examined with either support vector machines (SVMs, [5], [15]), Neural networks (NNs, [20], [9]) or more recently with Convolutional Neural Networks embedded in the field of Deep Learning (DL, [16] [6]). NNs and CNNs, as compared to SVMs, provide the advantage of only having to create a single model for a multi-class classification problem. Moreover, training time of SVMs increases significantly with the number of training samples. CNNs have the added benefit of alleviating the (possibly) very cumbersome process of feature engineering by providing the optimal filters for the task at hand. Once trained, execution time for a single classification remains low albeit there exist possibly millions of parameters in a single model. Lane et al. ask the question whether Deep Learning can revolutionize mobile sensing [12] and to their findings the usage of SoCs and DSPs on mobile devices is likely to boost the performance of Deep Learning solutions in terms of e.g. robustness for mobile sensing tasks such as activity, emotion and speaker recignition. However, in this contribution highly satisfactory results were achieved utilizing mainly CPU power therefore still leaving room for improvement. Rallapalli et al. analyze the feasibility of utilizing very Deep Neural Networks on mobile devices [17]. To their findings, the requirements with regards to memory are rather high which is why several optimization techniques and various sizes of NNs were implemented in order to alleviate for these difficulties. However, in this contribution the size of the CNN as well as execution of the classification was realized in a more than satisfactory manner as to allow for applications to run in real-time. CNNs [14] have proven to be an efficient and robust model for object recognition and have been successfully applied to many fields of application, especially in object recognition tasks from images [10], [13], [19], [4]. They however require the input to be of fixed size, which poses a hurdle having to be overcome when dealing with objects of varying size as is the case in this contribution. Our approach differs in many aspects from work related to the topic at hand. We present a light-weight method of adding the feature of mid-air gesture

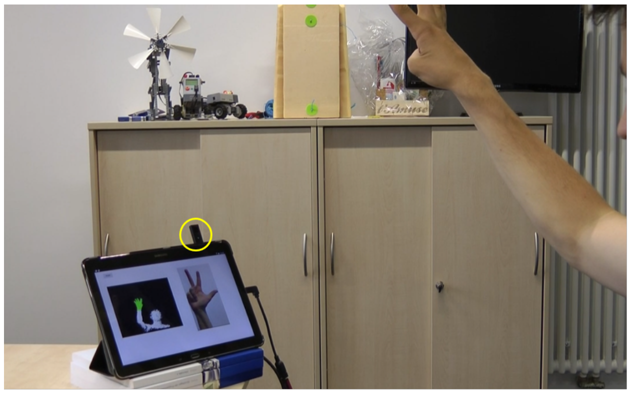

Figure 2: The Setup - tablet with a picoflex cam (indicated with yellow circle)

Figure 3: Picoflex Camera

interaction on mobile devices independent of its surrounding environment. By coupling an off-the shelf tablet with a small ToF camera we are able to take advantage of its high frame-rates at which it delivers reliable depth images of the nearby area. We overcome the problem of possibly highly-noisy data by a sophisticated feature generation process, the utilization of CNNs, classification thresholding and frame averaging which overall allows for a robust demonstrator setup. By normalizing the point clouds we moreover are able to deal with the complex problem of scaling in constant time. All in all, this setup allows for near-range mid-air hand gesture interaction to be realized at high frame rates on mobile devices, which, to the best of our knowledge, has not been realized in this fashion so far.

3. SYSTEM SETUP

3.1 Hardware

The system setup consists of a Galaxy Notepro 12.2 Tablet running Android 5.02. A Picoflex ToF sensor from PMD technologies is attached to the tablet via USB. It has an IRS1145C Infineonő 3D Image Sensor IC chip based on pmd intelligence which is capable of capturing depth images with up to 45 fps. VCSEL 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 (PC). The depth resolution of the Picoflex 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 samplings to be taken in order to reduce noise and increase precision. As the camera allows several pre-set modes with a different number of samplings we opt for 8 samplings 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.

3.2 Recognition Pipeline

With the setup described in the preceding section, our hand gesture recognition pipeline consists of several modules, of which this section should give a brief overview. In order to train a CNN, a large number of hand gesture samples is required. To this end, we build upon a large-scale hand gesture data base of 480.000 data samples which have been acquired from 16 different individuals. The details of this data base are outlined in the following section. In an initial step the background is cropped via depth thresholding leaving only palm, fingers and parts of the forearm. In order to get rid of the irrelevant arm parts, we analyze each point cloud with the help of a principal component analysis (PCA). Analyzing a cloud with respect to its three dimensions results in the Eigenvectors of each dimension as its principal components. Cropping along the principal component of the vertical axis allows for the removal of the irrelevant parts of the forearm. Figure 5 shows the effect of forearm cropping as the arm in the image in the top left corner (with PCA cropping) compared to the image in the bottom left corner (without PCA cropping) is significantly smaller in terms of PC-dimension. This is relevant for two reasons: Firstly, the arm part would negatively influence the feature generation process as meaningless feature would be generated stemming from the forearm. Secondly, CNNs require fixed-sized input to be trained and evaluated, therefore some form of data reduction to a fixed frame is required, which is more difficult to achieve if the forearm part is unpredictable in terms of its contribution to the feature generation process. The feature generation process for 3D point clouds is the core of the gesture recognition pipeline and it the specific 3D convolution of PCs to a format processable by CNNs what makes this approach feasible. Once the data base has been completely transformed it serves as the basis for training and evaluating our approach. We demonstrate how our approach achieves more than satisfactory results by running a number of experiments in a leave-one-out cross-validation scheme and moreover presenting a live demonstration of our system, showing its real-time capability, robustness versus changing illumination and invariance towards rotation, translation and scaling (one of the most difficult problems to solve). We are dealing with a complex multi-class classification problem of 10 different hand gestures. One single input vector presented to the CNN model generates, once propagated through the whole network, 10 activation values in the output layer which can be compared to class probabilities. We induce a thresholding measure to suppress uncertain cases, i.e. cases in which the net would produce uncertain results. The confidence threshold is chosen empirically and fixed at 0.4. This, in combination with an averaging window (5 frames averaging), stabilizes the live performance of our system, yielding a stable hand gesture recognition system running on a standard Samsung tablet. We proceed to describe the content of our database, the backbone of our system.

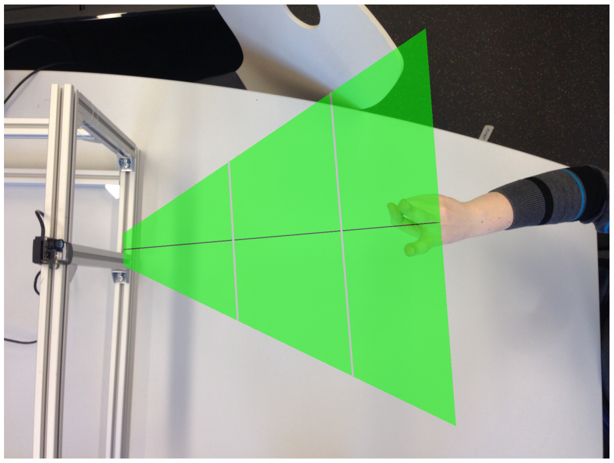
Figure 4: Setup for the recording of the database. The three zones of the recording: Near, intermediate and far.

4. THE HAND GESTURE DATABASE

Our database contains point cloud samples from 16 different individuals posing ten different gestures. Each gesture is recorded 3000 times over various distances yielding 480.000 data samples.

To only capture the relevant data points which are part of the user's right hand, distance thresholding is introduced during the recording. Points recorded by the sensor are simply cropped if above a certain threshold value Θ. Furthermore, the recording takes place in a predefined Volume of Interest (VOI) to ignore irrelevant data points to the sides of the user's hand.

Our database comprises 10 different static hand postures. The individual postures are denoted *ONE*, *TWO*, *THREE*, *FOUR*, *FIVE*, *FIST*, *FLAT*, *GRAB*, *PINCH*, *POINT* (as shown in Figure 1). These hand postures were chosen with respect to a trade-off between meaningfulness and complexity (in terms of disambiguation). Regarding the meaning of the postures, all of them can be facilitated to represent typical functions useful in various HMI scenarios. Pointing, e.g. can be compared to the selection gesture while the flat hand typically denotes the halt of a system and grabbing is commonly employed in VR environments to pick up and move an object. Counting from one to five, as indicated by the number of fingers, are very generic gestures applicable to various scenarios like selecting channels or levels. The difficulty in disambiguation results from the fact that the difference between some postures is defined by one finger only (e.g., *ONE* vs *TWO*) which, depending on the distance to the sensors, is equivalent to as few as 20-40 voxels.

The sensor is mounted in front of the user recording the nearby environment from an orthogonal angle. Each posture is performed and recorded 3000 times. In order to induce some variance into the data, each participant is asked to translate and rotate her/his hand during the recording. Furthermore, the recording area is divided into three zones: near (15-30cm), intermediate (30-45cm) and far (45-60cm) with respect to the distance between sensor and hand (cf. Figure 4).

The result of such a recording can be seen in Figure 5. The resulting PC is depicted for two different snapshots in subsequent movements (top vs. bottom) of the same participant from two different angles (left vs. right). Points closer to the sensor are depicted in yellow color, points further distant in a dark green color. Depending on the angle the user postures her/his hand toward the sensor, more or less light is reflected back and hence the precision of the measurement suffers, which is another potential drawback when utilizing ToF technology. Another possible source for noise is the fact that depth measurement relies on the amount of light reflected from the object, however too much light reflected over-saturates the measurement. This is visible by the amount of noise (or outliers) existent in the image. In the upper row, the user poses in a rather orthogonal angle towards the sensor, therefore there are less outliers visible towards the edges of the object. As compared to the bottom row, more outliers are recognizable as can be seen in the front view (left) and the side view (right) of the same posture. Dealing with noise is an important factor for the task of hand posture recognition in particular as depth sensors typically have a lower resolution than RGB cameras and therefore data samples suffering from much noise tend to strongly impede the employed algorithms. Consequently, no filtering or noise reduction techniques have been utilized to remove said outliers. However, due to the movement performed by each individual during the recording, the amount of data points belonging to the forearm differs strongly as can be seen in Figure 5 (top left vs. bottom left). Data points belonging to the forearm carry no information necessary to distinguish any of the posture in the database therefore we employed a cropping algorithm relying on a Principal Components Analysis (PCA) of the hand-arm object. Automatically removing most of the forearm results in a smaller first principal component, and more relevant information included in each sample, leaving only the palm and the fingers. The database has been recorded with a Creative Gesture Camera which has a higher lateral resolution for depth measurements (320×160) than the Picoflex cam. However, this does not matter for the gesture recognition pipeline presented in this paper due to the way we set up our features for the CNN and the subsequently following normalization, which is described in the next chapter.

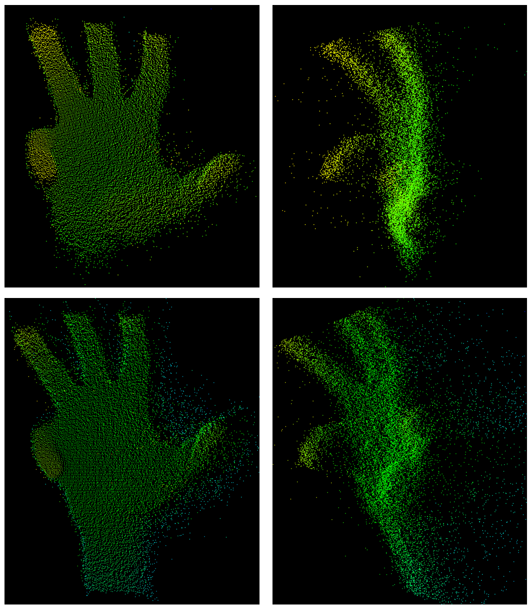

Figure 5: Sample recording of a hand posture. Top row: The same posture from the front view (left) and the side view (right). Bottom row: The same hand posture in a subsequent state taken after the snapshot in the top row (same angles). Noise and outliers resulting from errors in measurement are clearly visible when seen from the side view (right column).

5. NETWORK ARCHITECTURE

In order to be able to deal with three-dimensional input, this contribution presents an approach which transforms the raw 3D data into a format readable by CNNs. The need for a fixed-size input requires a specific partitioning of the 3D input. Given a set of 3D data points (voxels) of arbitrary extension (also referred to as point cloud), we propose the subdivision of the entire cloud into a fixed number of cubes, all having the same size. To this end, the maximal extension of the data points has to be calculated for the entire problem. This approach is explained in the following sections.

5.1 3D subdivision of point cloud input

In order to be able to work on 3D input data we employ a modified LeNet 5 implementation of the Theano library [1] with two convolutional layers. The input space is subdivided into n^3 hypercubes of fixed size. Each hypercube then contains a subset of data points from the original object. Depending on the density of the cloud, a certain number of cubes remains empty. In order to avoid too many empty hypercubes, which form the input for the CNN, we stretch the data to fit into the raster. To this end, the input cloud is normalized to the range (0,1) on each axis. This guarantees the data to be evenly distributed over all hypercubes. The value contained within a hypercube is determined by the number of data points it contains.

Each slice of the input vector, which will be described here on basis of an $8 \times 8 \times 8$ sized example, has to be reshaped to fit a designated pattern: The vector is reshaped in a way that each row fed into the convolutional layer represents one (x-y) slice of depth data in the original, resulting in an input matrix of 8×64 (cf. Figure 6 showing this for the case of 4^3). This way, a convolutional kernel of size 8×1 can be used to initially convolve the depth-axis, resulting in an 1×64 output of the first kernel. No max-pooling is used in this layer. The second layer reshapes this 1×64 output to 8×8, so that a 3×3 kernel can subsequently be utilized. This layer also implements 2×2 max-pooling, resulting in an output of 3×3. This output is then fed into the multilayer perceptron (MLP) layer of the convolutional net, which determines the output class.

5.2 Training setup

Training is performed on a single GeForce GTX 780 Ti graphics card. The main limit here is the device's memory capacity as our training/testing data set exceeds it's memory capability.

We evaluate our approach on a data set consisting of 480.000 data samples obtained from 16 different persons, each posing for 10 different hand gestures (cf. Figure 1). Each of the gestures is represented by 3.000 snapshots summing up to 30.000 data samples per person. This would result in the transformation and storage of 450.000 data samples by the Theano library during training for a 15-on-1 cross-validation (based on persons not on individual samples), including weights as well as the subsequent image transformation steps, which is more than the device can store during the training phase. The amount of data samples during training is therefore reduced to about 2.000 samples per gesture, each randomly taken from the whole sample set. This still yields a training set of 300.000 hand poses - more than enough to validate our approach.
Two different experimental runs are performed: an initial parameter search is started in order to determine the optimal setup for the CNN architecture. To this end, the whole data set is subsampled by randomly retrieving 100 data samples per person and pose, yielding 1000 samples per person for 10 randomly selected individuals. The number of test runs therefore amounts to:

$$\sum_{i=0}^{n} k_i^1 \sum_{j=0}^{m} k_j^2 \sum_{s=0}^{o} k_s^1 \sum_{s=0}^{o} k_s^2 \sum_{l=0}^{p} k_{mp}^2$$

Here, k_i^1 and k_j^2 denote the number of kernels within their respective layers. k_s^1 denotes a specific combination for the first layer, since we first transform the input as described in Section 5.1 (cf. Figure 6). If $k_s^1 = 0$ this conforms to an 8×1 kernel with no max-pooling. If $k_s^1 = 1$ this corresponds to a 7×1 kernel with 2×1 max-pooling etc. k_s^2 defines the size of the second kernel while k_2^{mp} consequently corresponds to the kernel size in the max-pooling layer. The resulting kernels from the first convolution layer are depicted in Figure 7.

6. EXPERIMENTS AND RESULTS

The results of our experiment run are presented in Table 1. We have conducted a series of experiments to test the validity of our approach on unseen data. To this end, each column represents the Classification Error (CE) achieved on Person p from the database when the network is trained on all the other data samples except on data coming from person p. Thus Table 1 shows the results of an n-fold cross-

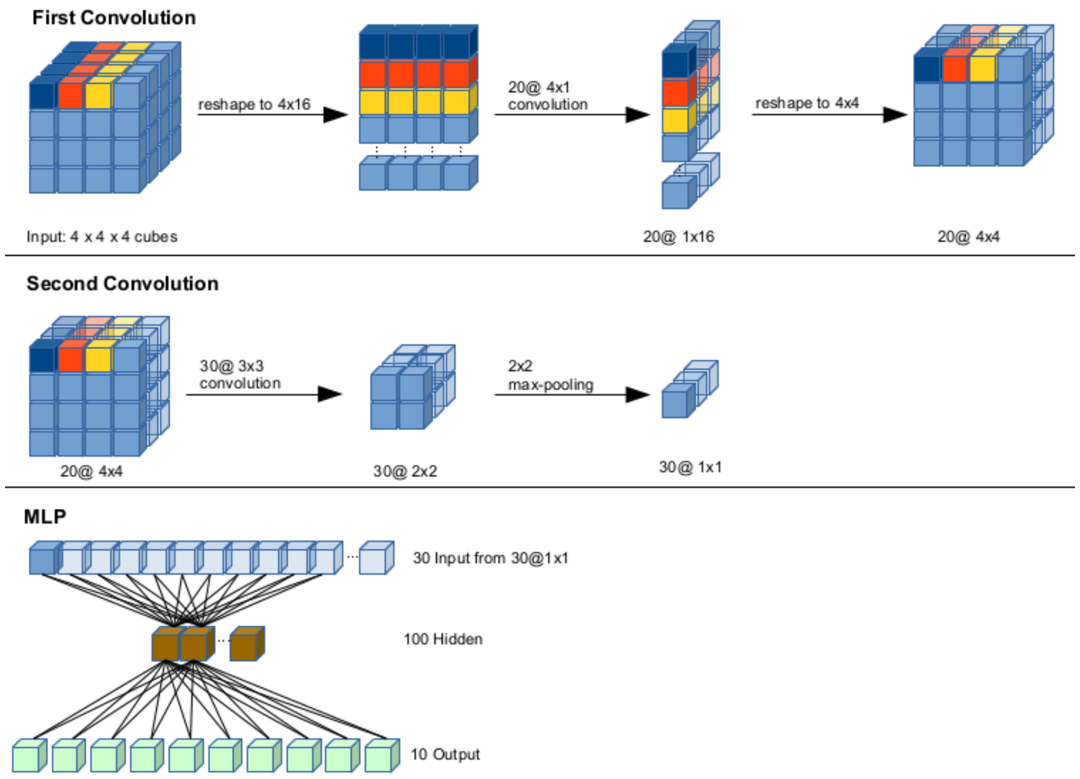

Figure 6: Setup of the CNN structure with two convolutional layers. Top row: First convolution step and reshaping. Center: Second convolution step and max-pooling. Bottom: MLP structure and input.

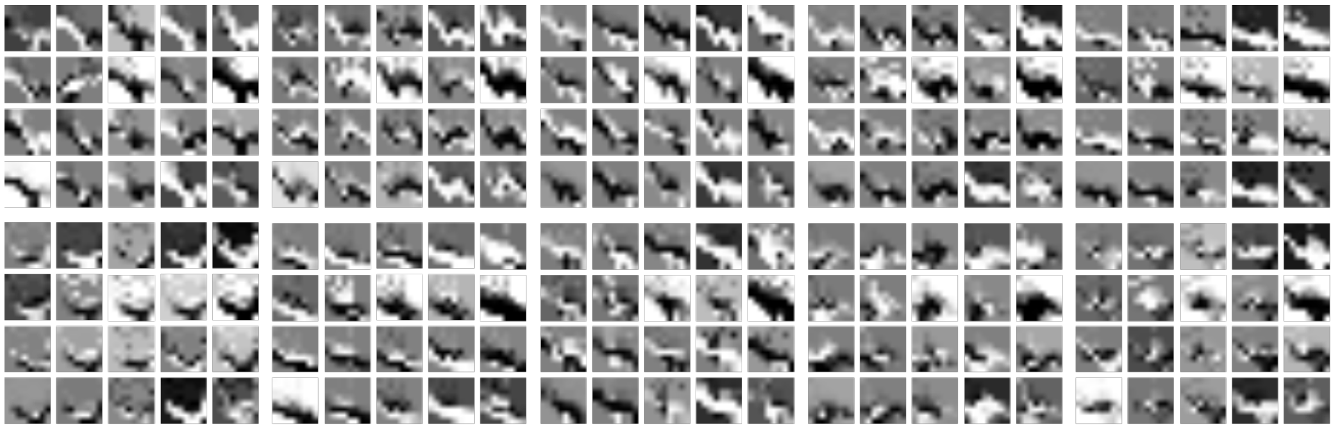

Figure 7: The resulting kernels from the first filter grouped together for each posture from the hand gesture data set (compare with Figure 1). The first layer of the CNN produces 20 different kernels. All 20 kernels produced per gesture are grouped and presented in analogous order from left to right, top to bottom.

validation run on the presented database. Experiments are conducted on an NVIDIA GTX 780 Ti using the Theano implementation of a CNN. As memory is limited for preparing and storing the data samples only 2000/3000 samples per gesture and person are randomly selected and taken for training and validation. Consequently, each run trains a CNN on 15 persons with 20.000 gestures samples each yielding a training set of 300.000 samples and a validation set of 20.000 samples.

The results show slightly varying, however very satisfactory performance. With the approach presented in this contribution we are able to demonstrate the robustness of the underlying methodology. Performance peaks with the model obtained for person 13 (1.5% error) and for 10 out of 16 persons error rates around or far below the 15% mark are achieved which is significant for such a large and diverse validation set of unseen data. This underlines the fact that CNNs paired with our data transformation technique are able to generalize well on this complex problem. Considering the fact that we are able to produce up to 40 classification steps per second this is more than enough to realize a real-time application which produces satisfactory behaviour. The main drawback is the somewhat poor performance on persons 2, 4, and 7 with CEs ranging from 30,5% to 48,3% and has to be attributed to various factors such as e.g. user behaviour or noise in the measurements.

An in-depth evaluation shows the very specific way in which the persons in question pose one and the same gesture during the process of the recording with a rapid change in finger positioning. This is one of the main reasons for the system's fluctuation in performance as the CNN tries to capture all the possible variants of the given classes which leads to problems in those cases where gestures are similar and somehow posed 'wrongly' by some participants. As for the training time, most of the CNNs converged quickly with as few as a thousand iterations (corresponding to roughly 1h train time) required for finding their global optimum in most cases with only one case (more than 22K iterations for person 14) requiring extensive parameter search. This is, however, the case which simultaneously corresponds to the model generalizing best with a CE of 1,5%. The termination condition for training is the non-improvement of the CE on the validation set for 100 subsequent epochs which in this case demonstrates the potential of extended optimization search. Execution time for a single sample falls well below 10ms which is negligible in regarding system workload.

7. SYSTEM PERFORMANCE

This chapter gives an overview of the performance of our system. Figure 8 demonstrates how we are able to achieve a light-weight 3D data processing pipeline able to run on 17Hz on average. Each plot contains 800 data samples represented by a single data point. Each data sample was processed by the system as described in Section 3.2.

The plots show the distribution of the computation load onto the various steps. Each diagram shows the plot for number of points in the cropped PC versus computation time which is shown in ms. The plot in Figure 8, top left show the linear increase of computation time depending on the number of points for the cropping algorithm which makes sense, as the cropping does indeed need more time to process a cloud with increasing PC size. For small clouds of size < 500 voxels between 30-60 ms are required larger PC samples of 4000-5000 voxels require 60-80 ms processing time. The plot in the top right shows the computation time for rendering the resulting point cloud as displayed in Figure 9.

A linear dependency between computation time and PC size is also visible, however this is negligible due to the fact that this part is used for the visualization of the feedback only and moreover ranges in a computation interval irrelevant to the process overall (avg.: 3ms). The computation time to create the PPV features as described before and process them through the network remains constant (nearly constant in the case for feature generation) which is the most interesting finding. Independent of cloud size input, it takes mostly below 10 ms to process the data through the CNN and around 1ms to compute the features. This is significant and one of the most important findings of these statistics as constant computation time is a highly desirable feature when working on mobile devices with limited computation power.
It should be noted here that although there is a number of outliers in each plot, this is mainly due to the garbage collector stopping all threads in the running application, thus forcing the increased computation in each case. However, most of the data samples clearly reflect the normal system behavior. The bottleneck of the system is the cropping algorithm having to deal with a large number of data points in order to prepare the PC for feature generation. The average computation time for a single data sample is 59.33ms and the average frame rate is about 16.85 Hz.

8. SUMMARY AND OUTLOOK

This contribution presents a novel approach to hand gesture recognition with ToF sensors on mobile devices. Coupling a small ToF camera with a standard tablet allows for a simple, robust demonstrator to be realized showing the potential of the technology when used in line with modern object recognition algorithms. To handle the data imprecision we introduce a robust feature generation technique which realizes fixed-size input required by CNNs. Normalizing the point cloud data allows for hand gestures to be recognized independent of scale. Near range interaction becomes possible at various distances in mid-air in a range of 15-50 cm. Rolling frame averaging and decision thresholding further increase robustness of the system. We achieve recognition results of up to 98,5% which corroborates the generalization capability of our contribution. With this setup, our demonstrator works at frame rates of up to 17Hz - more than satisfactory for any real-time application. This contribution shows that mid-air hand gesture recognition is possible on mobile devices with a single ToF sensor, adding this novel interaction technique to the already present and well-known touch interaction. Future work will be directed to further increase precision, testing different classifiers as well as the creation of interesting apps, which is, without a doubt, an exciting and novel field of research.

9. ACKNOWLEDGMENTS

This work was funded by a scholarship grant of the ENSTA ParisTech, Université Paris-Saclay.

10. REFERENCES

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and

Table 1: CE per person (in percent, 2nd row) and number of iterations (in thousand, 3rd row) per run

Pers.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CE	15.3	48.3	21.3	33.3	12.6	16.4	40.3	20.8	4.6	9.5	7.2	5.4	1.5	2.7	3.5	6.2
Iter.	8.3	1.1	3.7	1.5	1.2	0.6	1.8	5.5	11.9	1.9	7.9	9.1	1.0	22.2	3.6	6.5

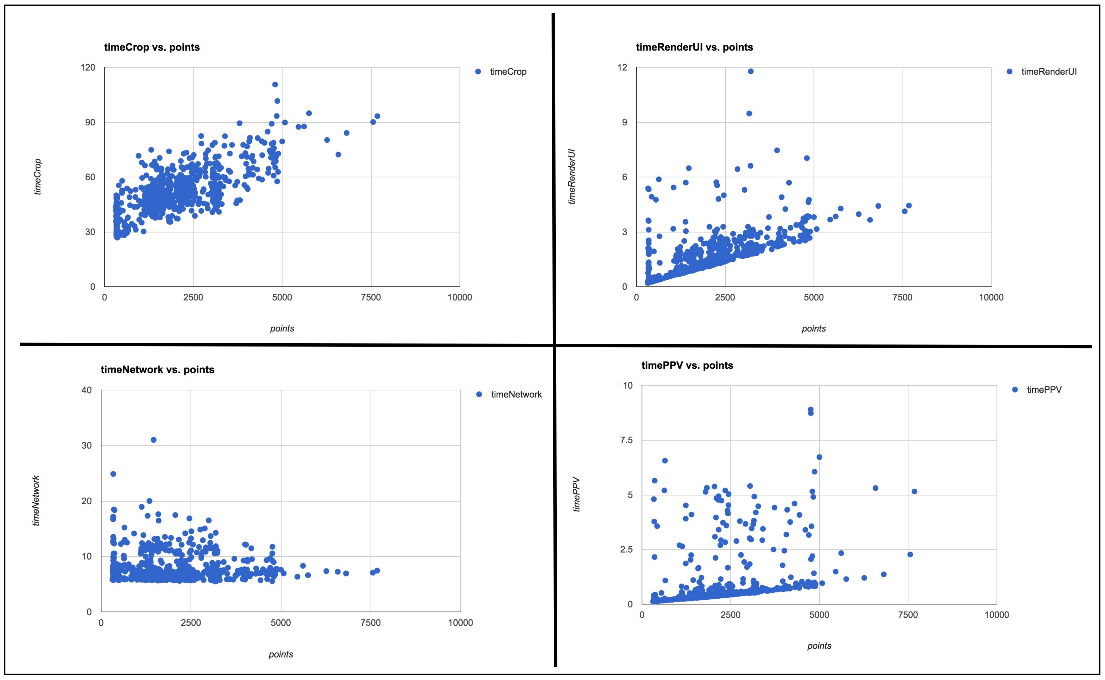

Figure 8: Computation load for a single data sample.

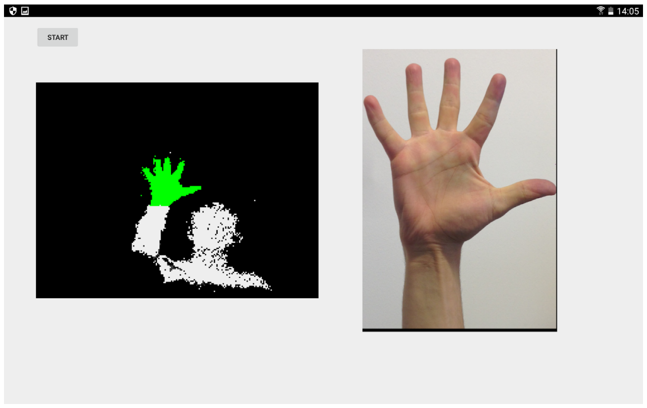

Figure 9: Demonstrator with the cropped point cloud (left) and the resulting interpretation of the hand gesture (right). The remaining voxels after PCA cropping are highlighted in green.

Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] K. K. Biswas and S. K. Basu. Gesture recognition using microsoft kinect®. In *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pages 100–103. IEEE, 2011.

[3] P. Breuer, C. Eckes, and S. Müller. Hand gesture recognition with a novel ir time-of-flight range camera–a pilot study. In *International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications*, pages 247–260. Springer, 2007.

[4] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[5] N. H. Dardas and N. D. Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and Measurement*, 60(11):3592–3607, 2011.

[6] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[7] E. Kollorz, J. Penne, J. Hornegger, and A. Barke. Gesture recognition with a time-of-flight camera. *International Journal of Intelligent Systems Technologies and Applications*, 5(3-4):334–343, 2008.

[8] T. Kopinski, S. Geisler, L.-C. Caron, A. Gepperth, and U. Handmann. A real-time applicable 3d gesture recognition system for automobile hmi. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2616–2622. IEEE, 2014.

[9] T. Kopinski, S. Magand, U. Handmann, and A. Gepperth. A pragmatic approach to multi-class classification. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] A. Kurakin, Z. Zhang, and Z. Liu. A real time system for dynamic hand gesture recognition with a depth sensor. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1975–1979. IEEE, 2012.

[12] N. D. Lane and P. Georgiev. Can deep learning revolutionize mobile sensing? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 117–122. ACM, 2015.

[13] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[14] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[15] Y. Liu, Z. Gan, and Y. Sun. Static hand gesture recognition and its application based on support vector machines. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD'08. Ninth ACIS International Conference on*, pages 517–521. IEEE, 2008.

[16] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pages 342–347. IEEE, 2011.

[17] S. Rallapalli, H. Qiu, A. Bency, S. Karthikeyan, R. Govindan, B. Manjunath, and R. Urgaonkar. Are very deep neural networks feasible on mobile devices?

[18] A. Ramey, V. González-Pacheco, and M. A. Salichs. Integration of a low-cost rgb-d sensor in a social robot for gesture recognition. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 229–230. ACM, 2011.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] E. Stergiopoulou and N. Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *Engineering Applications of Artificial Intelligence*, 22(8):1141–1158, 2009.

[21] J. Suarez and R. R. Murphy. Hand gesture recognition with depth images: A review. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 411–417. IEEE, 2012.

[22] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool, and M. Buss. Real-time 3d hand gesture interaction with a robot for understanding directions from humans. In *2011 Ro-Man*, pages 357–362. IEEE, 2011.

[23] M. Van den Bergh and L. Van Gool. Combining rgb and tof cameras for real-time 3d hand gesture interaction. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 66–72. IEEE, 2011.