2016 International Conference on Computing, Networking and Communications (ICNC), Workshop on Computing, Networking and Communications (CNC)

Touchless Interaction for Future Mobile Applications

Thomas Kopinski
Hochschule Ruhr West
Computer Science Institute
Lützowstraße 5
46236 Bottrop, Germany
Email: thomas.kopinski@hs-rw.de

Uwe Handmann
Hochschule Ruhr West
Computer Science Institute
Lützowstraße 5
46236 Bottrop, Germany
Email: uwe.handmann@hs-rw.de

***Abstract*—We present a light-weight real-time applicable 3D-gesture recognition system on mobile devices for improved Human-Machine Interaction. We utilize time-of-flight data coming from a single sensor and implement the whole gesture recognition pipeline on two different devices outlining the potential of integrating these sensors onto mobile devices. The main components are responsible for cropping the data to the essentials, calculation of meaningful features, training and classifying via neural networks and realizing a GUI on the device. With our system we achieve recognition rates of up to 98% on a 10-gesture set with frame rates reaching 20Hz, more than sufficient for any real-time applications.**

I. Introduction

As mobile devices are increasingly becoming the means of accessing the internet and the number of mobile phones nearly exceeds the world population, while the number of devices already exceeds the population size in most developed countries, the demand for new applications and new ways of interacting with them grows rapidly. Most of the smartphones available on the market are equipped with standard color cameras. However, devices with a 'real' 3D sensor are just beginning to enter the market with primary focus put on tablet PCs with smart phones following, as questions like power consumption and heat development being the main issues of concern.

Mobile devices have established multi-touch interfaces as de-facto standard for interaction as gestures provide a natural and intuitive means of control. Nevertheless, two-dimensional interaction faces certain restrictions, e.g. when trying to control objects in a virtual three-dimensional space. In such situations, space is limited on a device hence the interface has to be adapted accordingly. Three-dimensional gestures provide a solution by bringing along the third dimension for interaction while being more 'natural' as a means of control because manipulation of objects becomes more direct in the sense that cumbersome 2D-mapping can be omitted.

We present a real-time 3D-gesture recognition system for mobile devices and demonstrate how it can be incorporated into various Human-Machine Interaction (HMI) scenarios. Due to lack of availability we outsource the only step of data recognition via time-of-flight (ToF) technology onto a Laptop and implement the whole gesture recognition pipeline on a mobile device. We achieve recognition results of up to 90% on a 10 gesture set with a frequency of up to 20Hz. As the sensor technology is robust vs. any kind of lighting interferences, our system, being small in dimension, is applicable to possibly any kind of scenario.

The rest of the paper is set up as follows: We provide an overview of the relevant contributions to this topic in Section II. We go on to describe our hand gesture recognition pipeline along with the established database in Section III. The algorithms at the core of our system are responsible for creating descriptive features which is outlined in Section IV. These features serve as inout for our Neural Network module which is responsible for the hand gesture recognition part described in Section V. The process of detecting static and dynamic gestures is presented in Section VI. We corroborate the functionality of our system with a series of experiments by demonstrating that stable, satisfactory results are achievable in real-time in Section VII. Finally, in Section VIII, we conclude with a brief discussion and an outlook on future work.

II. Related work

As mentioned before, for temperature and energy consumption reasons, there has yet to be developed a mobile phone employing ToF-sensor technology. Therefore, research in this direction is scarce. When looking at work conducted towards 3D interaction with gestures on mobile devices most of it is addressing accelerometer or stereo camera usage. To the best of our knowledge, there exists no work, trying to leverage ToF-technology to implement a real-time hand gesture recognition system, while work utilizing ToF-sensors for 3D interaction is abundant.

Henrysson et al. [1] research the idea of utilizing a phone's camera to move and rotate tangible objects for an augmented reality (AR) application, however it is focused rather in the direction of usability questions. The work of Gao [2] is directed towards making an AR App for handheld devices using a PrimeSense camera. His approach is heavily relying on skin color detection making a robust outdoor application difficult to realize. Nevertheless, the device is connected via USB to the tablet, making the data recording possible on the device directly. The work of Matilainen et al. [3] detects hands

978-1-4673-8579-4/16/$31.00 ©2016 IEEE

gestures via template matching on data coming from an RGB camera for mobile phone interaction.

III. System Setup and Hand Gesture Database

The setup of our system can be seen in Figure 1. The ToF-sensor is mounted to the front console recording the environment close to the driver. It is connected to a standard Laptop responsible for collecting the data coming from the sensor and sending the data packages to the iPad via WiFi. The rest of the recognition process is realized on the tablet and the complete pipeline is described in the next section.

A. System Setup

Fig. 1. Setup with ToF-sensor, Laptop and iPad in a car interior.

Our system consists of various modules handling the gesture recognition task as displayed in Figure 2. The gesture recognition pipeline is divided into two main tasks: 'Data Recording and Transmitting' on the server side and 'Data Processing and Gesture Recognition' on the mobile client. The main computation task is outsourced onto the mobile client, however the data recording and transmitting remains on the server side. The ToF sensor records the data in the near-driver environment and transmits it ASCII-encoded to the mobile client. The data package receiver module is responsible for making sure that the whole data package for a single frame is received completely and transmits the data package to the Point Cloud Generator. Once the point cloud is set up, it is transferred to the principal component analysis (PCA) module, which crops the unnecessary parts of the forearm. This technical side of this process is explained in more detail in Section IV-A. Once cropped, the feature vector for the remaining point cloud is calculated in the next step (cf. Section IV-B). This feature vector serves as input for the training and classification stages in the neural network (NN) classifier module. The output of the Neural Network module determines the class for the recorded frame at a time point *t*. The static gesture recognizer module (SGR) is responsible for this part. Based on this information, dynamic hand gestures are recognized in the dynamic hand gesture recognizer (DGR) from the output of the SGR. Since the whole system reacts to whether static or dynamic gestures have been recognized, the visualizer module utilizes input from both the SGR and DGR.

We implemented an infotainment system for testing, which the driver is able to interact with. The static hand gestures are mapped to static functions within the infotainment system, i.e. responding to channel selections, accepting/declining/ending an incoming phone call or, in the dynamic case, zooming in/out or increasing/decreasing the volume. At the very core of the hand gesture recognition pipeline lies our database which is described in the next section.

B. Hand Gesture Database

We recorded data from 20 persons, each displaying 10 different hand poses (cf. Figure 3). For each gesture, 3000 samples are recorded, summing up to 30000 samples per person and a total database of 600,000 samples. In order to induce some variance into the data, during the recording phase each participant is asked to rotate and translate their hand in all possible directions. Moreover, to tackle the task of scaling, for each gesture we define 3 different distance ranges, in which the participant is asked to perform the hand gesture in order to ensure sufficient sample coverage for various distances. Each frame is recorded at a resolution of 160x120px at 90fps with the Camboard nano, making it robust to daylight interferences and thus applicable in any outdoor scenario. This results in an alphabet of ten hand poses: Counting from 1-5 and *fist*, *stop*, *grip*, *L*, *point* denoted by *a-j* (cf. Figure 3). For the chosen participants, both male and female, the size of the hand ranges from 8,5cm - 9,5cm in width and from 17,0cm - 19,5cm in length.

IV. PCA Cropping and Feature Calculation

The module responsible for pruning the point clouds from its unnecessary parts - in this case the forearm for which it is difficult to predict to what extent it is contained in each cloud - is the PCA module, as mentioned in Section III-A. Once the important parts remain, the feature calculation for hand palm and fingers can be performed. These steps are described in the following two sections.

A. Finding the principal axis of a point cloud

The main directions of the cloud are found using Principal Component Analysis (PCA) [4]. PCA aims to find uncorrelated basis vectors for an arbitrary set of data vectors. Eigenvectors (also termed "principal components") are ordered by the variance of data points projected onto them, allowing efficient data compression by omitting principal components of low variance. This algorithm is applied as shown below, using as input the set of n 3D coordinates of points in a point cloud denoted x_j, $j \in [0, n]$).

- The mean value $\bar{x} = \frac{1}{n} \cdot \sum_{j=1}^{n}(x_j)$ is computed.
- The scatter matrix is calculated :

$$S = \sum_{j=1}^{n}(x_j - \bar{x})(x_j - \bar{x})^\top$$

This matrix can be used as maximum-likelihood estimate of the covariance matrix.

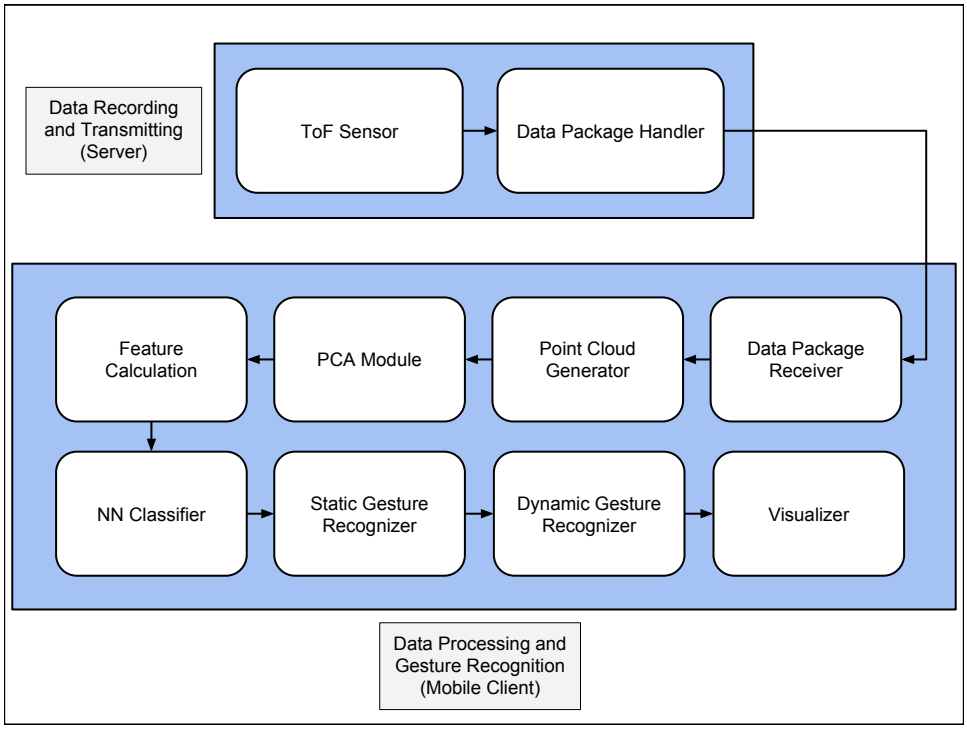

Fig. 2. The whole gesture recognition pipeline as described in Section III-A

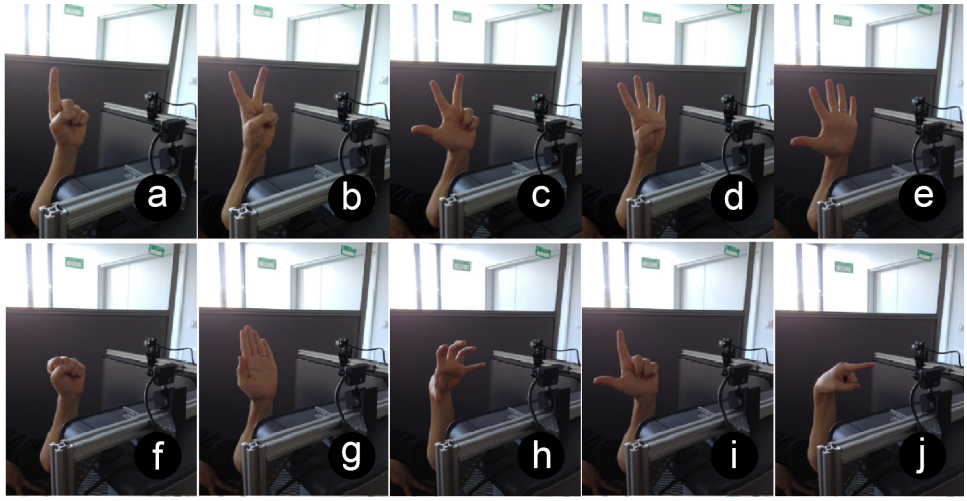

Fig. 3. The hand gesture database consisting of 10 different static hand poses.

- The Eigenvectors of this matrix yield the principal components.

We intend to cut off 'unnecessary' parts of the cloud, i.e. outliers and elongated parts of the forearm. In this case, the principal components correspond to orthogonal vectors that represent the most important directions in the point cloud. The vector with the most important y-component allows to recognize the axis hand-forearm.

The wrist, as the link between the hand and the forearm, is detected in order to determine a limit for the cropping. The employed method assumes that the distance between the endpoint of the fingers and the centroid is an upper bound of the distance between the centroid and the wrist.

To find the endpoint of the hand towards the direction of the fingers, tests are made along the axis, starting at the centroid and moving progressively upward. At each step, we determine whether there are points within a designated small neighborhood around the axis. The upper end of the hand is marked if this number of neighboring points equals 0. Then the bottom limit for the wrist is fixed at the same distance from the centroid, but in the inversed direction along the y-axis. All points below this wrist limit are cut out which is exemplarily

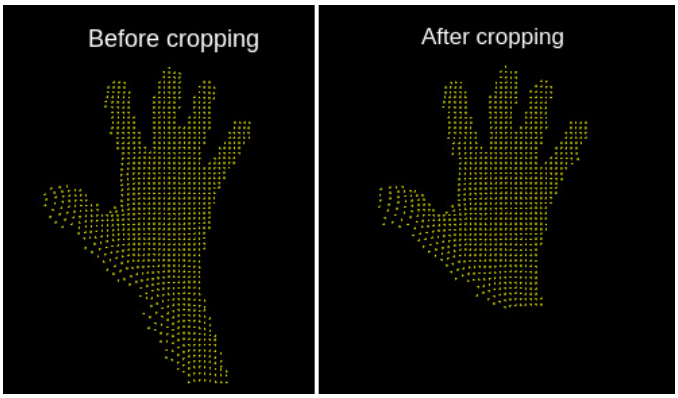

Fig. 4. Point cloud before PCA-cropping (left) and after (right).

shown in Fig.4.

B. Forming descriptive feature vectors from Point Clouds

The PFH-Descriptor (PFH-Histogram) [5] is a local descriptor which relies on the calculation of normals. It is able to capture the geometry of a requested point for a defined k-neighbourhood. Thus, for a query point and another point within its neighbourhood, four values (the point features or PFs) are being calculated, three of which are angle values and the fourth being the euclidean distance between these two points. The angle components are influenced by each point's normal, so in order to be able to calculate them, all the normals have to be calculated for all points in the cloud. Therefore we are able to capture geometric properties of a point cloud in a sufficient manner, depending on the chosen parameters. These parameters have been thoroughly examined in our previous work which led for example to an optimal choice for the parameter *n*, the radius for calculation of the sphere which encloses all points used to calculate the normal of a query point. One major drawback is the fact that the PFH-descriptor cannot be easily embedded into a real-time applicable system as the computation cost becomes too high, when extended to a global descriptor. To overcome this issue, we present a modification of the PFH-Descriptor.

Our version of the PFH-Descriptor makes use of its descriptive power while maintaining the real-time applicability. Using the PFH in a global sense would mean having to enlarge the radius so that every two point pairs in the cloud are used to create the descriptor. This quickly results in a quadratically scaling computation problem as a single PFH-calculus would have to be performed 10000 times for a point cloud of 100 points. Given the fact that our point clouds have a minimum size of 200 points up to 2000 points and more, this is not feasible for our purposes. Therefore we randomly choose 10000 point pairs and use the quantized PFs to build a global 625-dimensional histogram. We calculate one descriptor per point cloud which forms the input for the neural network. We have conducted numerous experiments with this descriptor in various application scenarios and found it to be well balanced in terms of descriptiveness and computation cost.

V. Neural Network Training and Feature Fusion

We trained two MLPs and divided the database accordingly to allow for the implementation of a sophisticated fusion technique. Both MLPs have three layers - input, hidden and output layer. Extensive parameter search in work conducted so far yielded this network structure with 50 hidden neurons in each MLP and standard parameters for training. Each output layer comprises 10 neurons corresponding to the 10 hand pose classes. The first MLP has an input layer of size 625, corresponding to the size of the feature vector while the second MLP has an input layer of size 635 - the size of the feature vector added to the number of output neurons of the first MLP. Each Point Cloud is transformed into a histogram of length 625 - as described in Section IV-B and fed into the first MLP. The MLP processes the feature vector, determines the neuron values in the output layer and concatenates these values again with the feature vector which is then presented as input into the second MLP. The neuron with the highest activation in the second MLP corresponds to the designated class. For more information please refer to our preceding work in e.g. [6] as the theory is beyond the scope of this paper. We have tested various techniques for this problem and this fusion approach resulted in the best generelization performance. The neural network architecture was implemented using the FANN library [7].

VI. Static and Dynamic Gestures

We define hand gestures as being dynamic, i.e. changing in state over time, and they can be contrasted against static hand poses which in turn do not change in state. Therefore, in an in-car infotainment system, a static hand pose as the *one* pose could be connected to selecting the first audio channel while a dynamic zooming in/out gesture could be applied in a typical maps application. Our approach makes use of the simple fact that a dynamic hand gesture must have a clearly distinguishable starting pose and a clearly distinguishable ending pose. Consequently a 'grabbing' movement can be defined by starting as hand pose 'h' and ending as hand pose 'f' in our hand pose database. This is a clearly defined feature and serves as a universal definition in that any kind of dynamic gesture can be captured in this sense. The number of theoretically definable gestures therefore sums up to $n\dot{(}n-1) = 90$ gestures definable from our static database, as any case is bidirectional i.e. a gesture from 'a' to 'b' can be performed vice versa.

We denote a static hand pose as a state s at any given point in time t: s_t. A sequence of n occurrences of a certain hand pose is defined as $< s_{t=0}, ..., s_{t=n} >$. During the interaction phase our gesture recognition module takes consecutive snapshots which are interpreted by the system via a voting scheme. For a series of 10 consecutive snapshots, a static hand pose is recognized by the most frequent occurrence within this series if the occurrence is above a certain threshold. In order to take into account that our frame rate can vary between 5-20Hz, a threshold of 7 yields satisfactory performance in terms of recognition rate and user acceptance as the feedback has to be provided to the user and in order to suppress too frequent

changes.
We use this as a basis of defining dynamic hand postures within this time series as follows. For a dynamic gesture any occurrence of the starting state at any given point in time s_t^{st} followed by any occurrence of the ending state s_{t+m}^{en} with $m \geq 1$ within the observed time series corresponds to the classification of the sequence as containing the dynamic gesture: $< s_{t=0}, ..., s_t^{st}, ..., .s_{t+m}^{en}, ..., s_{t=n} >$. This the most simplified notation which allows for the fact that misclassifications may occur in between the detection of the starting state and the detection of the ending state. The only condition being made here is that both classifications must occur within a certain time frame and that the starting state must be detected before the ending state.

In order to stabilize recognition results, a simple extension of the definition above can be made. As soon as one occurrence of a starting state is made this starting point of a gesture is only taken as valid if it is immediately followed by one or multiple occurrences of the same state, i.e. $s_t^{st} = s_{t+1}^{st}$. The same rule can be applied for the the ending state of a dynamic gesture. The restriction that these consecutive occurrences of states must form uninterrupted subchains within the observed time frame suffices to define a robust dynamic gesture recognition pipeline, recognizing dynamic hand gestures well in real-time as we will see in Sec.VII. Of course a proper choice of parameters is immanent and strongly depends on factors such as frame rate, classification rate and user feedback. The choice of the length of the observed time frame restricts other parameters as the length of the subchains for starting and ending sequences. The benefit of this simple definition is the fact that we allow for uncertain states or even misclassification to occur in between starting and ending sequences of a dynamic gesture as well as within the time frame as a whole. Additionally, as we found out during the testing phase, this approach provides extra flexibility as every user has a different way of performing a gesture and thus an otherwise more restrictive definition of a dynamic gesture can be too obstructive.

VII. Experiments and Results

The system was realized to work on an iPad Air with an A7 chip with 64-bit architecture and M7 motion coprocessor and an iPhone 4S with an A5 chip based on dual-core ARM Cortex-A9 MPCore CPU. On both devices, gesture recognition was realized in real-time. With the implementation described above, we achieved frame rates of up to 20Hz. The main bottleneck is the feature generation taking from 0.03s up to 0.1s, depending on the cloud size. Preparing the data takes around 0.007s while classification remains at around 0.0003s with the FANN library, being the most negligible part as are the other steps in our frame work concerning computation time.
The ToF-sensor employed is the camboard nano capable of capturing 3D data with up to 90 fps under any lighting condition. Recognition rates are around 85% for 400 dynamic hand gestures performed by 10 persons with this setup. We achieve an averaged 74% classification rate for all static gestures with a leave-one-out cross-validation test. Individual results vary between 51% and 98% depending on the participant.

Fig. 5. Demo Application: The ToF-Sensor is mounted to the iPhone to be able to capture close-range interaction. A user zooms in on the teapot with a dynamically defined gesture.

Tests for dynamic gestures were conducted with 10 different persons whose data is not contained in the database, i.e., the results given in Table I demonstrate the generalisation

performance of the system to previously unseen persons. The system was explained aforehand to each participant and feedback provided its response when a hand gesture was recognized. For the experiments in this contribution, a time frame of length $n = 10$ was defined, meaning that from the moment user input is generated we observe the last 10 consecutive snapshots and the corresponding classifications in order to determine whether a dynamic gesture is contained or not. Moreover, we found that for a time frame of this size, two identical consecutive starting poses and two identical consecutive ending poses suffice to efficiently detect a dynamic gesture.

Four gestures were defined to this end: Grab, release, zoom in and zoom out. Each gesture can be defined via an unambiguous static state from our database. The grabbing motion (shown in Fig. 5) is defined as starting with hand pose 'h' and ending in hand pose 'f'. The corresponding release gesture is the exact inverse starting with hand pose 'f' and ending with hand pose 'h' (cf. Fig.3). The pinching/zooming gesture is defined analogously and can be seen as the same gesture known from pinching/zooming in 2D in e.g. a typical maps application. To this end, pinching is defined as starting with hand pose 'i' and ending with hand pose 'f'. Consequently the inverse movement from 'f' to 'i' defines the zooming gesture cf. Fig.3. All users found the concept easy to grasp and interacted with our system by this means naturally.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
grab	10	6	3	5	5	8	6	7	5	4
release	7	6	9	8	9	8	8	9	9	7
zoom in	10	10	10	10	10	10	10	10	10	10
zoom out	9	10	7	10	9	9	10	8	9	9

TABLE I
EACH GESTURE WAS PERFORMED 10 TIMES BY EVERY PERSON. A COLUMN ENTRY REPRESENTS THE NUMBER OF CORRECTLY RECOGNIZED SAMPLES PER PERSON AND GESTURE.

The overall classification rate is 82.25% averaged over all persons and gestures. There is a 100% recognition rate for zooming in, followed by 90% for zooming out, 80% for release and 59% for grabbing. This shows that with a robust detection mechanism for static hand poses our approach resembles a viable solution. However misclassifications still occur for all hand gestures, although this amounts to only a few cases as the statistics show. In the case of zooming in/out, the misclassifications sum up to 16 and 3 cases respectively, which in turn makes up for 1% or 0.1% of all the cases. For grab/release the numbers are higher, namely 151 and 78 misclassifications respectively which in turn makes up for 10% and less than 5% of all classifications. Comparing these number to the figures in Tab. I helps explaining why the individual gestures perform more poorly as it seems evident that more misclassifications of static hand poses impair the performance of the system. However it also shows that misclassifications are allowed to happen while a gesture is still recognized correctly, which shows the flexibility of our approach. A more in-depth analysis of our recordings reveals that misclassifications occur in 153 cases of all the correctly recognized gestures performed by the participants within the time frame and between starting and ending sequence. Nevertheless our approach helps to remain robust by dismissing these samples. This shows that such a simple definition of a dynamic gestures is able to provide a satisfactory and stable performance under challenging conditions in real-time. As these statistics also indicate, users tend to remain longer in the final positions of a gesture, nearly 3-4 times longer in some cases. Hence e.g. in the case of 'grabbing' the number of detected ending states (state 'f' - 1160 samples) is more than 3 times higher than the number of detected starting states (state 'h' - 338 samples). Why that is the case is subject to further analysis but it helps to provide further stability mechanisms for the problem at hand.

VIII. Discussion and Outlook

This paper demonstrates how a gesture recognition pipeline for outdoor use, i.e. use under challenging lighting conditions, is realizable with ToF-sensors on mobile devices. As the work in this area limits itself mainly to exploiting existing RGB cameras built into the device, we show the capabilities of making the transition to ToF-based interaction. We demonstrate how a real-time applicable 3D gesture recognition pipeline can be implemented with the main computational complexity outsourced onto the device itself. Two systems were realized - one on an iPad serving as an infotainment replacement in a car and a 3D demo of a tangible object which can be rotated and translated via 3D gestures in all directions. As mobile devices are used in nearly all possible scenarios, this approach shows how ToF-sensors can be utilized to make stable 3D-interaction on handheld devices happen in everyday lives. Future work will focus on boosting computation power in order to improve the overall recognition results as well as developing more complex scenarios and applications. Using time-of-flight technology allows for research applications to work under direct sunlight, therefore potential applications facilitating 3D-gesture control for outdoor use will also be the focus of further research.

References

[1] A. Henrysson, J. Marshall, and M. Billinghurst, "Experiments in 3d interaction for mobile phone ar," in *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*. ACM, 2007, pp. 187–194.

[2] L. Gao, "Natural gesture based interaction for handheld augmented reality," 2013.

[3] M. Matilainen, J. Hannuksela, and L. Fan, "Finger tracking for gestural interaction in mobile devices," in *Image Analysis*. Springer, 2013, pp. 329–338.

[4] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[5] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3384–3391.

[6] T. Kopinski, S. Magand, A. Gepperth, and U. Handmann, "A pragmatic approach to multi-class classification," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, p. to appear.

[7] S. Nissen, "Implementation of a fast artificial neural network library (fann)," *Report, Department of Computer Science University of Copenhagen (DIKU)*, vol. 31, 2003.