Clustered Dynamic Graph CNN for Biometric 3D Hand Shape Recognition

Jan Svoboda [1 2] **Pietro Astolfi** [3 4] **Davide Boscaini** [3] **Jonathan Masci** [2] **Michael Bronstein** [5 6]

Abstract

The research in biometric recognition using hand shape has been somewhat stagnating in the last decade. Meanwhile, computer vision and machine learning have experienced a paradigm shift with the renaissance of deep learning, which has set the new state-of-the-art in many related fields. Inspired by successful applications of deep learning for other biometric modalities, we propose a novel approach to 3D hand shape recognition from RGB-D data based on geometric deep learning techniques. We show how to train our model on synthetic data and retain the performance on real samples during test time. To evaluate our method, we provide a new dataset NNHand RGB-D of short video sequences and show encouraging performance compared to diverse baselines on the new data, as well as current benchmark dataset HKPolyU. Moreover, the new dataset opens door to many new research directions in hand shape recognition.

1. Introduction

Biometric systems based on 3D hand geometry provide an interesting alternative in places where fingerprints and palmprints cannot be used (e.g. wearing latex gloves, very dirty hands) and face recognition is not an option either (e.g. wearing face masks, helmets, goggles or other protective equipment). Solutions have been proposed in the past (Kanhangad et al., 2009; 2011; Wang et al., 2014a; Svoboda et al., 2015), however, they do not offer satisfactory performance neither are easy to use as they often impose strong constraints on the acquisition environment. Recent advances in deep learning and 3D sensing suggest that one could try to drop many of the acquisition constraints. Attempts to propose a novel system that would take advantage of the latest trends, however, require new dataset, as evaluation data for such approaches are missing at the moment.

This paper presents a novel approach to biometric hand shape recognition by utilizing some recently developed principles in Geometric Deep Learning (GDL) (Bronstein et al., 2017a). In particular, our method is based on Dynamic Graph CNN (DGCNN) (Wang et al., 2019). Taking into consideration that a hand is a rather complex geometric object, we replace the Global Pooling Layer with so-called *Clustered Pooling Layer*, which allows having a piece-wise descriptor (per-cluster) of the hand, instead of creating just one global descriptor.

Successful training of GDL models however requires noticeable amount of annotated data, which one typically does not have in biometrics. To overcome this limitation, inspired by works of Schuch (Schuch et al., 2016) and Svoboda (Svoboda et al., 2017), we created a synthetic dataset of hand point clouds using the MANO (Romero et al., 2017) model and show how to train the proposed model fully on synthetic data while achieving good results on real data during the experiments.

In order to evaluate our method, this work presents a new dataset for less constrained 3D hand biometric recognition. The dataset was acquired using a low cost acquisition device (an off-the-shelf RGB-D camera) in variable environmental conditions (there were no constraints on where the system was placed during acquisition). Each sample is a short RGB-D video of a user performing a predefined gesture, which allows researchers to capture frames in different poses and opens door to possibly new research areas (e.g. non-rigid hand shape recognition, hand shape recognition from a video sequence, etc.). To set a baseline performance, we evaluate the novel dataset on two state-of-the-art GDL models, namely the PointNet++ (Qi et al., 2017) and DGCNN (Wang et al., 2019).

Our main contributions are three-fold

- Clustered DGCNN: A novel geometric deep learning architecture for 3D hand shape recognition based on the Dynamic Graph CNN.
- Transfer learning solution for training of 3D hand shape recognition models using a synthetically generated dataset of hands.

[1]Università della Svizzera italiana, Lugano, Switzerland [2]NNAISENSE, Lugano, Switzerland [3]Fondazione Bruno Kessler, Trento, Italy [4]University of Trento, Trento, Italy [5]Imperial College London, London, United Kingdom [6]Twitter, London, United Kingdom. Correspondence to: Jan Svoboda <jan.svoboda@usi.ch>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

- NNHand RGB-D: New biometric dataset of RGB-D video sequences for the purpose of 3D hand shape recognition.

2. Clustered Dynamic Graph CNN

The human hand is a complex and highly non-rigid surface. Moreover, RGB-D scans are often noisy. Matching noisy samples of hands using a global descriptor seems very challenging. An easier task would be to rather aim at describing the hand surface divided into semantically meaningful parts. These parts can be pre-defined based on human anatomy, for example by looking at the skeletal structure of the hand. Such clustered description (see Figure 1(b)) retains more information and should be robust against noise and, possibly non-rigid, transformations.

The core of our approach is a modified Dynamic Graph CNN architecture. In the following text, we will denote multilayer perceptron as MLP$(m, n, \ldots)$, where $m, n, \ldots$ are the nmber of parameters of each layer of the MLP. We further define the shape parameter space and pose parameter space as $\mathcal{S} \in \mathbb{R}^{10}$ and $\mathcal{P} \in \mathbb{R}^{12}$ respectively. Our model starts with two EdgeConv (Wang et al., 2019) modules, both with $k = 10$ nearest neighbors and *max* feature aggregation type. The first module has MLP$(6, 64, 64, 128)$ and the latter one MLP$(128 + 128, 256)$. Outputs of both EdgeConv modules are concatenated and passed forward. The model is then forked into two branches, one regressing the pose parameters $\mathbf{p} \in \mathcal{P}$ and the other one the shape parameters $\mathbf{s} \in \mathcal{S}$ of the input point cloud. The first branch composed of a *Global Pooling* (GP) module with MLP$(128 + 256, 1024)$ followed by another sub-network MLP$(1024, 512, 256, 12)$. The second branch, which outputs the shape parameters $\mathbf{s} \in \mathcal{S}$, replaces the GP module with novel *Clustered Pooling* module which internally has an MLP$(128 + 256, 1024)$ and is followed by another MLP sub-block defined as MLP$(1024, 512, 256, 10)$. Our clustered Dynamic Graph CNN architecture is schematically depicted in Figure 1(a).

Clustered Pooling Module inspired by the differentiable graph pooling (Ying et al., 2018; Cangea et al., 2018; Bianchi et al., 2019), the *Global Pooling Module* in the shape regression sub-network is replaced with a novel *Clustered Pooling Module* (see Figure 1(b)). It allows to dynamically learn a clustering function $l : \mathbb{R}^F \to \mathbb{R}^C$, which produces cluster assignment probability vector $\mathbf{c} \in \mathbb{R}^{N \times C}$ into $C \in \mathbb{N}$ clusters for a vector of $N \in \mathbb{N}$ feature points $\mathbf{x} \in \mathbb{R}^{N \times F}$ as

$$\mathbf{c} = \text{softmax}(l(\mathbf{x})). \tag{1}$$

To get the clustered representation, the input feature points $\mathbf{x} \in \mathbb{R}^{N \times F}$ further undergo a non-linear transformation defined as $f : \mathbb{R}^F \to \mathbb{R}^{F'}$ and are subsequently aggregated into the C clusters as:

$$\mathbf{x}_f = f(\mathbf{x}), \tag{2}$$

$$\hat{\mathbf{x}} = \frac{\mathbf{c}^T \mathbf{x}_f}{\mathbf{D}}, \tag{3}$$

where the division represents a Hadamard division, $\mathbf{D} \in \mathbb{R}^{C \times F'}$ is a matrix with identical columns, where each column is defined as $\left(\sum_{i=1}^{N} \mathbf{c}_i\right)^T \in \mathbb{R}^{C \times 1}$ and $\hat{\mathbf{x}} \in \mathbb{R}^{C \times F'}$ is the pooled representation of the transformed input $\mathbf{x}_f \in \mathbb{R}^{F'}$. The novel clustered DGCNN architecture which we call *Clustered DGCNN* is schematically depicted in Figure 1(a).

Training. The optimization of our model is posed as a regression over the shape and pose parameters $\mathbf{s} \in \mathcal{S}$ and $\mathbf{p} \in \mathcal{P}$, and simultaneous classification of the point clusters, while feeding a three-dimensional point cloud as an input. It is defined using the following objective function E for a batch of $M \in \mathbb{N}$ samples:

$$E = E_{\mathcal{S}} + \lambda_1 E_{\mathcal{P}} + \lambda_2 E_{clust}, \tag{4}$$

where $E_{\mathcal{S}}$ is the MSE loss for the regression of the shape parameters

$$E_{\mathcal{S}} = \frac{1}{M} \sum_{m=0}^{M-1} |\hat{\mathbf{s}}_m - \mathbf{s}_m|^2, \tag{5}$$

$E_{\mathcal{P}}$ is the MSE loss for the regression of the pose parameters

$$E_{\mathcal{P}} = \frac{1}{M} \sum_{m=0}^{M-1} |\hat{\mathbf{p}}_m - \mathbf{p}_m|^2, \tag{6}$$

and E_{clust} is a cross-entropy loss which enforces the classification of points into correct clusters. It is defined as:

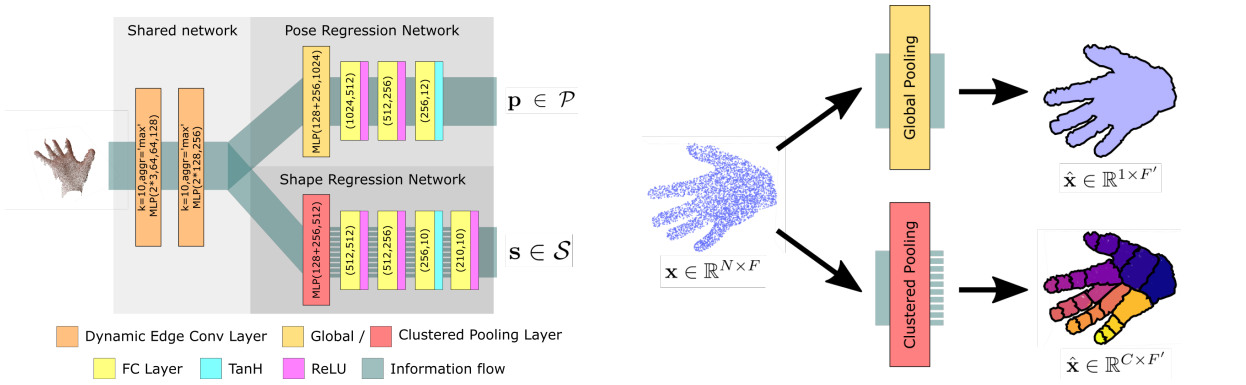
$$E_{clust} = \frac{1}{M} \sum_{m=0}^{M-1} -\log\left(\frac{\exp(\mathbf{c}_m^{y_m})}{\sum_j \exp(\mathbf{c}_m^j)}\right), \tag{7}$$

where $\mathbf{c}_m$ is the vector of cluster probabilities for points in a point cloud and y are the cluster labels of these points. Hyperparameter λ_1 is weighting the importance of regressing the pose parameters $\mathbf{p} \in \mathcal{P}$ with respect to the shape parameters $\mathbf{s} \in \mathcal{S}$ and λ_2 is a hyperparameter weighting the importance of the cluster classification loss.

3. Experiments

The following sections describe the datasets that have been used, the two simple baseline methods we compare to and results in different scenarios[1].

[1]Experimental setup details and additional epxeriments can be found in the supplemnetary material.



(a) Clustered DGCNN architecture (b) Global vs. Clustered Pooling

Figure 1. Our novel Clustered DGCNN: (a) The network architecture; (b) The difference between global and clustered pooling of the N input feature points. Global pooling creates a single new descriptor for the whole hand shape, while clustered pooling creates a new descriptor for each of the C semantically meaningful clusters.

3.1. Datasets

Synthetic training dataset Recent developments in hand pose estimation (Romero et al., 2017; Kulon et al., 2019; 2020) have provided us, besides others, with a very convenient deformable model of three-dimensional hands called MANO (Romero et al., 2017), which is publicly available. It allows generating hands of arbitrary shapes in arbitrary poses. We use the pre-trained MANO hand model to generate 200 subjects with 50 poses each, resulting in a total of 10000 three-dimensional hands, whose shape and pose is controlled via $\mathbf{s} \in \mathcal{S}$ and $\mathbf{p} \in \mathcal{P}$. Such three-dimensional models can be easily reprojected into range data.

Testing datasets Extensive evaluation of our approach on both the new dataset and a standard benchmark HKPolyU (Kanhangad et al., 2009; 2011) has been carried out.

3.2. Input point cloud pre-processing

Before feeding a hand point cloud to a model, it undergoes the following pre-processing steps. First, each point cloud is subsampled using Furthest Point Sampling (FPS) to 4096 points. Consequently, each sample is aligned to a reference hand point cloud using the Iterative Closest Point (ICP) algorithm.

3.3. Baseline methods

Two state-of-the-art algorithms in deep learning on point clouds have been used as baselines. In particular, the PointNet++ (Qi et al., 2017) architecture (PointNet++ and Big PointNet++ baselines), successor of the famous PointNet by (Qi et al., 2016), and the Dynamic Graph CNN (Wang et al., 2019) (DGCNN and Big DGCNN baselines), which are both implemented as a part of PyTorch Geometric library (Fey & Lenssen, 2019).

3.4. Feature matching

For matching, we consider the per-cluster shape parameters as the output feature vector in case of our novel *Clustered DGCNN*. There are 21 different clusters which results in a vector of 210 dimensions. For a fair comparison, in case of *PointNet++* and *DGCNN* baselines, which both perform a global pooling, we take the output of the layer before the last in the shape regression network as the feature vector, which has 256-dimensions. Different metrics have been tried for computing the distance, where the L1 metric has shown to be the most suitable one.

3.5. Results

We have evaluated our method in both All-To-All and Reference-Probe matching scenarios. Employed dataset splitting strategies for different datasets are described in the supplementary material. In both scenarios, our novel Clustered DGCNN outperforms both baselines by a margin and sets new state-of-the-art on the NNHand RGB-D dataset as well as HKPolyU v1 and v2 standard benchmarks.

We show the importance of the novel clustering loss by additionaly comparing to a Clustered DGCNN model trained without it (*w/o* E_{clust} in the tables below).

Comparing to the results of (**?**Kanhangad et al., 2011), we use heavily downsampled inputs, yet obtain on-par or superior perfromance compared to the original works. Remarkably, in case of reference - probe matching on the HKPolyU v2 dataset, we can compare to the results presented by (Kanhangad et al., 2011), where we outperform their method in terms of EER by a huge margin of 7% (which is an improvement by 60% compared to their EER of 17.2%). This further supports the high potential of our novel method.

Table 1. Matching performance of presented methods on different datasets in terms of Top-1 accuracy and EER.

Matching Type	Method	NNHand RGB-D		HKPolyU v1		HKPolyU v2	
		Top-1 [%]	EER [%]	Top-1 [%]	EER [%]	Top-1 [%]	EER [%]
All-To-All	PointNet++	53.42	47.19	30.40	34.28	9.12	37.55
	Big PointNet++	48.35	34.79	40.62	38.51	17.72	44.24
	DGCNN	76.20	21.70	84.63	19.03	39.12	27.40
	Big DGCNN	73.54	22.05	73.79	19.66	29.30	27.62
	Ours (w/o E_{clust})	94.94	16.67	97.01	12.70	49.30	24.34
	Ours	**98.23**	**14.45**	**99.27**	**7.92**	**59.65**	**25.08**
Reference-Probe	PointNet++	24.05	39.08	17.29	33.14	12.10	35.39
	Big PointNet++	18.86	40.54	22.37	34.72	13.68	37.70
	DGCNN	24.56	24.31	54.24	16.79	42.98	16.58
	Big DGCNN	28.48	23.45	45.54	15.83	31.92	19.53
	Ours (w/o E_{clust})	51.14	17.09	69.83	11.82	56.84	14.36
	Ours	**62.28**	**13.75**	**85.65**	**7.26**	**69.82**	**10.48**

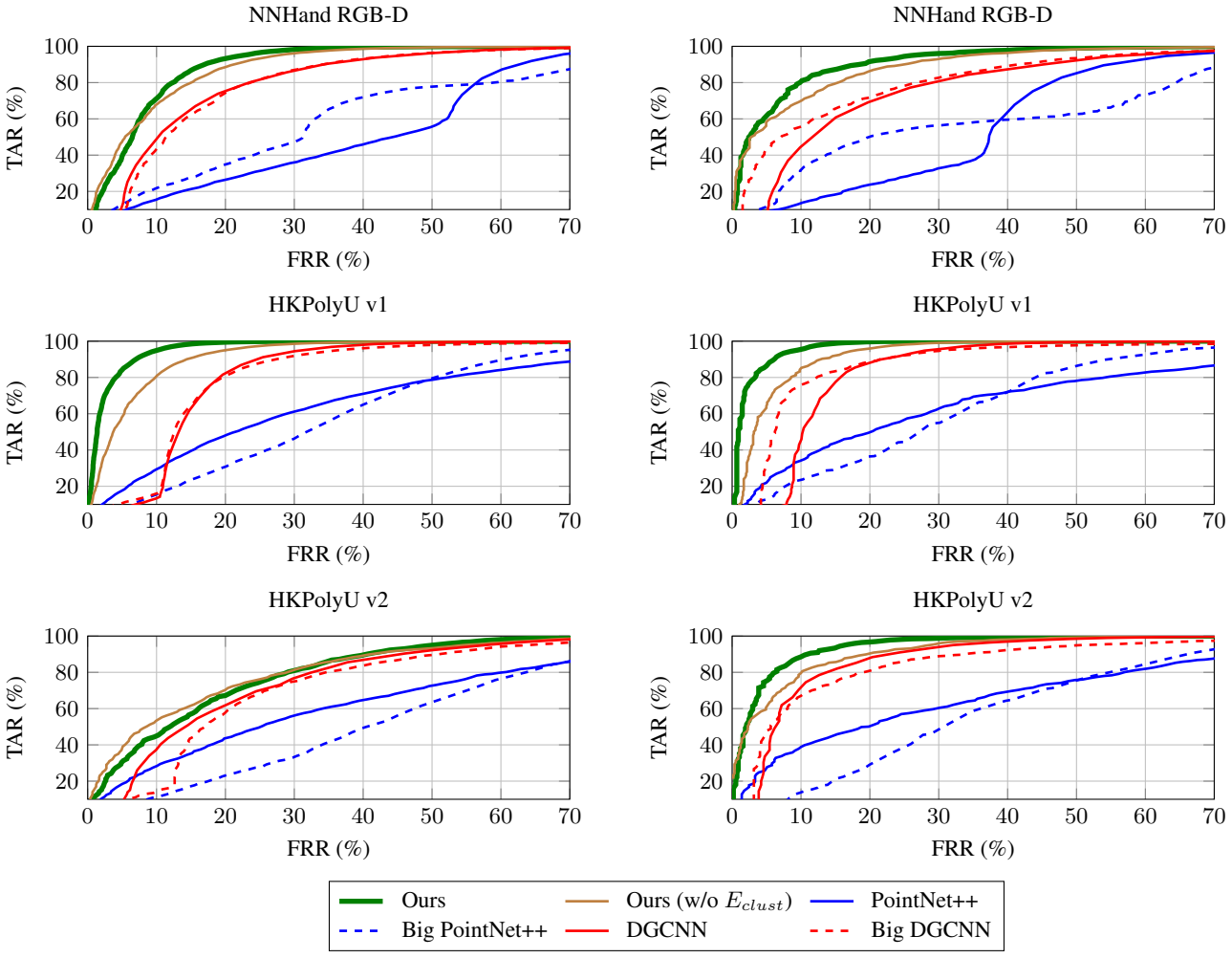


Figure 2. All-To-All matching ROC curves (tradeoff between acceptance and rejection rates) of the presented methods on different datasets.

Figure 3. Reference-probe matching ROC curves (tradeoff between acceptance and rejection rates) of the presented methods on different datasets.

Acknowledgements

This work was supported in part by ERC Consolidator grant No. 724228 (LEMAN).

References

Bianchi, F. M., Grattarola, D., and Alippi, C. Mincut pooling in graph neural networks. *arXiv:1907.00481*, 2019.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Process. Mag.*, 34(4):18 – 42, 2017a.

Cangea, C., Veličković, P., Jovanović, N., Kipf, T., and Liò, P. Towards sparse hierarchical graph classifiers. *arXiv:1811.01287*, 2018.

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., and Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. *Proc. CVPR*, 2017.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *Proc. ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Kanhangad, V., Kumar, A., and Zhang, D. Combining 2d and 3d hand geometry features for biometric verification. *Proc. CVPR*, 2009.

Kanhangad, V., Kumar, A., and Zhang, D. Contactless and pose invariant biometric identification using hand surface. *IEEE Transactions on Image Processing*, 20(5):1415 – 1424, 2011.

Kulon, D., Wang, H., Güler, R. A., Bronstein, M. M., and Zafeiriou, S. Single image 3d hand reconstruction with mesh convolutions. *Proc. BMVC*, 2019.

Kulon, D., Güler, R. A., Kokkinos, I., Bronstein, M. M., and Zafeiriou, S. Weakly-supervised mesh-convolutional hand reconstruction in the wild. *Proc. CVPR*, 2020.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. CVPR*, 2016.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Proc. NIPS*, 2017.

Romero, J., Tzionas, D., and Black, M. J. Embodied hands: Modeling and capturing hands and bodies together. *Proc. SIGGRAPH Asia*, 2017.

Schuch, P., Schulz, S., and Busch, C. De-convolutional auto-encoder for enhancement of fingerprint samples. In *Proc. IPTA*, 2016.

Svoboda, J., Bronstein, M. M., and Drahansky, M. Contactless biometric hand geometry recognition using a low-cost 3d camera. In *Proc. ICB*, 2015.

Svoboda, J., Monti, F., and Bronstein, M. M. Generative convolutional networks for latent fingerprint reconstruction. In *Proc. IJCB*, 2017.

Wang, C., Liu, H., and Liu, X. Contact-free and pose-invariant hand-biometric-based personal identification system using rgb and depth data. *Journal of Zhejiang University SCIENCE C*, 15:525–536, 2014a.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38 (5), 2019.

Ying, R., You, J., Morris, C., Ren, X., Hamilton, W. L., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. *Proc. NIPS*, 2018.

Zabatani, A., Surazhsky, V., Sperling, E., Ben Moshe, S., Menashe, O., Silver, D. H., Karni, T., Bronstein, A. M., Bronstein, M. M., and Kimmel, R. Intel realsense sr300 coded light depth camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2019.

A. NNHand RGB-D Dataset

This section introduces a new dataset of human hands collected for the purpose of evaluating hand biometric systems. At the moment, the first version of the dataset, with suffix *v1*, has been released and comprises of 79 individuals in total. It is planned to continue collecting extended version *v2* with the aim of about 200 different identities.

The dataset is collected using an off-the-shelf range camera *Intel RealSense SR-300* (Zabatani et al., 2019) in different environments and lighting conditions. Each person contributing to the dataset is asked to repeatedly perform three different series of gestures with the hand in front of the camera, resulting in three RGB-D video sequences collected for each participant. Each subject in the dataset has the following annotations: *User ID*, *Gender* and *Age*. The dataset is mainly targetting three-dimensional hand shape recognition. However, the presence of RGB-D information also allows attempting two-dimensional shape or palmprint recognition. Attempting palmprint recognition on this dataset might however be extremely challenging due to the poor quality of the RGB data in many sequences.

Video sequences There are three types of gestures that each participant is asked to perform repeatedly four times. Between the gestures, the participants are asked to remove their hands from the scene and re-enter. This naturally forces them to re-introduce the hand in the scene each time and provides more diverse and realistic samples.

The recorded video sequences are depicted in Figure 4. A more detailed description of the dataset can be found on the project webpage[2].

Applications The main purpose of the dataset is to serve as a new evaluation benchmark for three-dimensional hand shape recognition based on a low-cost sensor. The dataset allows for experiments with non-rigid three-dimensional shape recognition from either dynamic video sequences or static frames as well as attempts to perform recognition viewing the hand from either its palm or dorsal side. Additionally, the *Gender* and *Age* information can be used for experiments aiming at recognizing the gender or age of a person based on the shape of their hand.

B. Experimental setup

The following sections describe the datasets that have been used, the two simple baseline methods we compare to and results in different scenarios.

[2]https://handgeometry.nnaisense.com/

B.1. Datasets

Synthetic training dataset Recent developments in hand pose estimation (Romero et al., 2017; Kulon et al., 2019; 2020) have provided us, besides others, with a very convenient deformable model of three-dimensional hands called MANO (Romero et al., 2017), which is publicly available. It allows generating hands of arbitrary shapes in arbitrary poses. The generation of hand sample is controlled by two sets of parameters. First are the so-called shape parameters in space $\mathcal{S} \subseteq \mathbb{R}^{10}$ that define the overall size of the hand and lengths and thickness of the fingers. The second group of parameters are the pose parameters in space $\mathcal{P} \subseteq \mathbb{R}^{12}$, where the first 9 parameters define the hand pose in terms of non-rigid deformations (e.g. bending fingers, etc.) and the last 3 parameters define the orientation of the whole hand in the three-dimensional space. We use the pre-trained MANO hand model to generate 200 subjects with 50 poses each, resulting in a total of 10000 three-dimensional hands, whose shape and pose is controlled via $\mathbf{s} \in \mathcal{S}$ and $\mathbf{p} \in \mathcal{P}$. Such three-dimensional models can be easily reprojected into range data.

NNHand RGB-D database A dataset of fixed RGB-D frames has been sampled from the video sequences. For each subject, the sequence number 1 has been taken and 10 samples have been acquired while the hand is held straight up with the fingers extended and palm facing the camera. The dataset currently contains 79 subjects, which gives a total of 790 samples. Similarly, the sequence number 2 has been used to obtain a second set of 790 samples. For reproducibility of this evaluation, the acquired subset of RGB-D frames is stored together with the original NNHand RGB-D dataset. Each frame captured from the video sequences undergoes several pre-processing steps.

First, the background is removed using the depth information. Subsequently, to avoid problems with objects or other parts of the body appearing in the frames, a mask keeping only the central area of each frame is applied (see Figure 5). Next, an OpenPose-based (Cao et al., 2017) single RGB image hand pose estimator[3] is used to estimate the hand keypoints. Thanks to the one-to-one mapping between RGB and depth information, this allows to filter out the undesired part of the hand below the wrist in the whole RGB-D frame. Step-by-step preprocessing of a random frame is depicted in Figure 5.

HKPolyU v1 database. A dataset of 177 subjects containing in total 1770 RGB-D samples that were acquired with high precision Minolta Vivid 910 range scanner. Each subject has been scanned in two sessions in different time periods, obtaining 5 samples per session. The precision of

[3]https://github.com/erezposner/HandKeyPointDetector

Figure 4. The three sequences (one in each row) recorded for each subject in the dataset. The first sequence is sliding hand vertically into the scene with an open palm and removing it again, repeatedly. In the second sequence, rotation of the hand is added when the hand is upright. In the last sequence, the user closes and reopens the fist while the hand is upright.

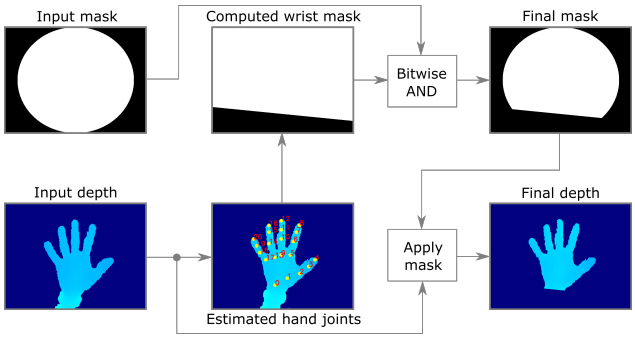


Figure 5. Each sample in the dataset undergoes the following pre-processing steps.

the data is enough to perform both 3D hand geometry and 3D palmprint recognition.

HKPolyU v2 database. It is a dataset of 114 subjects with a total of 570 RGB-D samples that were acquired using the Minolta Vivid 910 range scanner. Each subject has been scanned 5 times, each time presenting his hand on different global orientation. Besides, the precision of the data is enough to perform both 3D hand geometry and 3D palmprint recognition.

B.2. Input point cloud pre-processing

Before feeding a hand point cloud to a model, it undergoes the following pre-processing steps. First, each point cloud is subsampled using Furthest Point Sampling (FPS) to 4096 points. Consequently, each sample is aligned to a reference hand point cloud using the Iterative Closest Point (ICP) algorithm.

B.3. Baseline methods

Two state-of-the-art algorithms in deep learning on point clouds have been used as baselines. In particular, the PointNet++ (Qi et al., 2017) architecture, successor of the famous PointNet by (Qi et al., 2016), and the Dynamic Graph CNN (DGCNN) (Wang et al., 2019), which are both implemented as a part of PyTorch Geometric library (Fey & Lenssen, 2019).

PointNet++ The baseline PointNet++ architecture has two Set Abstraction (SA) (Qi et al., 2017) modules. The first SA module has subsampling ratio $r = 0.5$, neighborhood radius $\rho = 0.2$ and MLP$(3, 64, 64, 128)$. It is followed by second SA module with $r = 0.25, \rho = 0.4$ and MLP$(3 + 128, 128, 128, 256)$. The output of the second SA module is forked into two parallel branches. The first branch is supposed to output the shape parameters $\mathbf{s} \in \mathcal{S}$. It is composed by a Global Abstraction (GA) (Qi et al., 2017) module with MLP$(3+256, 256, 512, 1024)$ followed by another MLP subblock defined as MLP$(1024, 512, 256, 10)$. The second branch, instead, is outputting the pose parameters $\mathbf{p} \in \mathcal{P}$ and is composed of a GA module with MLP$(3 + 256, 256, 512, 1024)$ whose output is fed to an MLP module MLP$(1024, 512, 256, 12)$.

Big PointNet++ Second version with more parameters has been evaluated in parallel. This model has a bigger subnetwork for the shape regression. In particular, the GA module is equipped with MLP$(3 + 256, 256, 512, 1024 \times 21)$ whose output is fed to an MLP module MLP$(1024 \times 21, \frac{1024\times21}{12}, \frac{1024\times21}{24}, 10 \times 21, 10)$.

Dynamic Graph CNN (DGCNN) The model starts with two EdgeConv (Wang et al., 2019) modules, both with

$k = 10$ and *max* aggregation type. The first module has MLP$(6, 64, 64, 128)$ and the latter one MLP$(128 + 128, 256)$. Outputs of both EdgeConv modules are concatenated and passed forward. The model is then forked into two branches, one regressing the pose parameters $\mathbf{p} \in \mathcal{P}$ and the other one the shape parameters $\mathbf{s} \in \mathcal{S}$ of the input point cloud. The first branch composed of a GA module with MLP$(128 + 256, 1024)$ followed by another MLP subblock with defined as MLP$(1024, 512, 256, 12)$. The second branch is almost the same, with only one difference: The final MLP block's output is 10-dimensional as it outputs the shape parameters $\mathbf{s}$.

Big DGCNN Second version with more parameters has been evaluated in parallel. This model has a bigger subnetwork for the shape regression. In particular, the GA module is equipped with MLP$(128 + 256, 1024 \times 21)$ whose output is fed to an MLP module MLP$(1024 \times 21, \frac{1024 \times 21}{12}, \frac{1024 \times 21}{24}, 10 \times 21, 10)$.

B.4. Matching scenarios and splitting strategies

All-To-All matching In this experiment, each output feature vector is taken and its distance to feature vectors of all other samples in the dataset is computed. The sample with the shortest distance is taken as the matching class.

Reference-Probe matching A very popular way of evaluating biometric algorithms on diverse datasets is performing so-called reference - probe matching, where the dataset is split into two parts, one is the reference (i.e. the database) and the rest is the probe (i.e. the samples one wants to identify). Different splitting strategies have been applied depending on the dataset at hand.

For the HKPolyU v1 dataset, the splitting strategy proposed by (Kanhangad et al., 2009) is followed, choosing the 5 samples from the first session as the reference and the 5 samples from the second session as the probe for each user.

In case of HKPolyU v2, we use the splitting strategy used in (Kanhangad et al., 2011), where 1 sample is chosen as probe and all the other 4 as reference. This process is repeated 5 times, always picking different sample as the probe to produce the genuine and impostor scores for the generation of the ROC curve and computation of the EER.

NNHand RGB-D database has 10 samples per user from sequence 1 and another 10 samples from sequence 2. For each user, the 10 samples from sequence 1 are selected as the reference and the other 10 samples from sequence 2 are left as the probe.

B.5. Semantic segmentation analysis

Our method (*Clustered DGCNN*), besides others, outputs the semantic segmentation of the point cloud into parts, which the network was enforced to learn during training by the cluster assignment loss (see Equation 4) using the cluster annotations provided with the synthetic training samples.

There is no ground truth segmentation for the testing data and thus we provide a qualitative evaluation in Figure 6, which supports that the *Clustered DGCNN* has learnt to segment the point cloud in a meaningful way. Aggregating information inside each cluster, therefore, provides a meaningful piece-wise representation of the point cloud.

One should notice that due to the presence of noise in the input point clouds, the segmentation is prone to produce some outliers in the finger regions (see Figure 6). Influence of such inconsistencies on the final descriptor is reduced by averaging feature vectors in each semantic region in order to produce the global segment descriptor (i.e. a cluster).

B.6. Ablation study

Two ablation studies are performed in order to justify our architecture design choices as well as the employed loss function.

Clustered Pooling Layer To clarify that the novel architecture does not perform better only because its increased capacity compared to the classical PointNet++ and DGCNN, we created their extended versions, which we call *Big PointNet++* and *Big DGCNN* respectively. The architectures are the same, but the number of parameters in the shape regression subnetwork is changed (see Section B.3 for details). The results in Table 1 and Figure 2 and 3 show that simply increasing the network capacity does not result in noticeable performance gain in most cases.

Semantic hand clustering We train another version of Clustered DGCNN without the cluster assignment loss E_{clust} to demonstrate its importance. An example of the learnt segmentation without E_{clust} is shown in the last row of Figure 6. Compared to our novel solution (the first two rows in the figure), there is no apparent meaning to the produced segmentation of the hand. Moreover, by far not all of the 21 available clusters are well exploited. The results in Table 1 and Figure 2 and 3 confirm that without semantically meaningful clustering, the solution is less robust and yields significantly lower performances especially in the case of reference-probe matching.

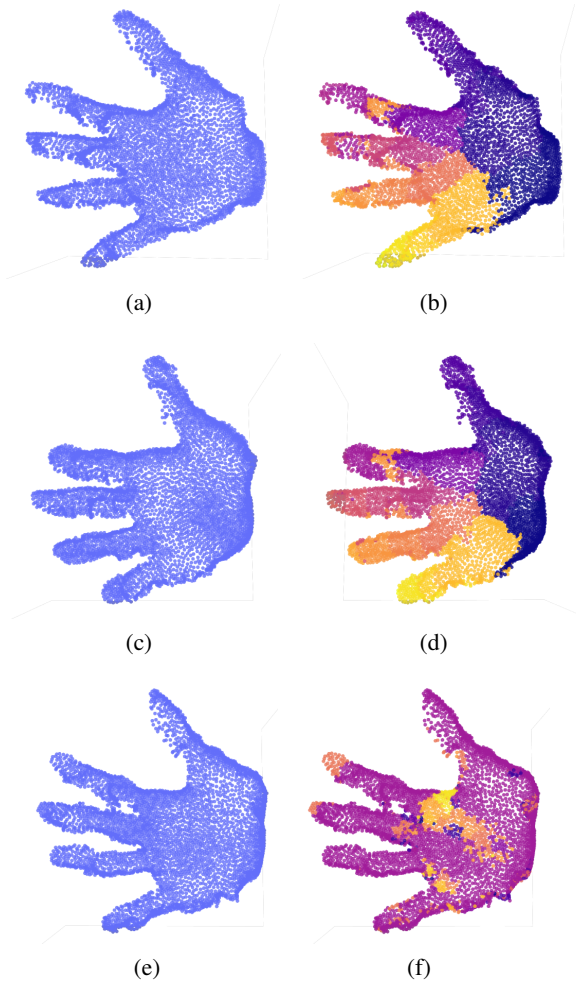


Figure 6. The first two rows show clustering of points computed by Clustered DGCNN for two real samples. One can notice some inconsistency around some of the fingers. Last row shows the effect of omitting clustering loss E_{clust} during training. (a,c,e) The original point cloud; (b,d,e) Result of clustering the point cloud.