2021 IEEE International Joint Conference on Biometrics (IJCB) | 978-1-6654-3780-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCB52358.2021.9484368

PointFace: Point Set Based Feature Learning for 3D Face Recognition

Changyuan Jiang[1,2], Shisong Lin[1,2], Wei Chen[3], Feng Liu[1,2], and Linlin Shen *[1,2]

[1]Computer Vision Institute, College of Computer Science and Software Engineering
[2]Guangdong Key Laboratory of Intelligent Information Processing
Shenzhen University, Shenzhen 518060, China
[3]University of Birmingham
{1900271054, linshisong2018}@email.szu.edu.cn, {feng.liu, llshen}@szu.edu.cn,
wxc795@cs.bham.ac.uk

Abstract

Though 2D face recognition (FR) has achieved great success due to powerful 2D CNNs and large-scale training data, it is still challenged by extreme poses and illumination conditions. On the other hand, 3D FR has the potential to deal with aforementioned challenges in the 2D domain. However, most of available 3D FR works transform 3D surfaces to 2D maps and utilize 2D CNNs to extract features. The works directly processing point clouds for 3D FR is very limited in literature. To bridge this gap, in this paper, we propose a light-weight framework, named PointFace, to directly process point set data for 3D FR. Inspired by contrastive learning, our PointFace use two weight-shared encoders to directly extract features from a pair of 3D faces. A feature similarity loss is designed to guide the encoders to obtain discriminative face representations. We also present a pair selection strategy to generate positive and negative pairs to boost training. Extensive experiments on Lock3DFace and Bosphorus show that the proposed PointFace outperforms state-of-the-art 2D CNN based methods.

1. Introduction

Face recognition is one of the most important biometric techniques for personal identification. Recently, convolution neural networks (CNNs) based 2D face recognition (FR) has achieved great success and has surpassed human performance [16, 24, 28]. Despite of powerful CNNs and tremendous accessible training data, 2D FR is still challenging due to various illumination conditions and self-occlusions caused by large head poses. However, in contrast to 2D images, 3D face data, *e.g.*, point cloud, can provide shape information intrinsically invariant to pose and illumination changes [4]. Previous works [4, 12] proved that 3D face recognition has the potential to address aforementioned challenges in the 2D domain.

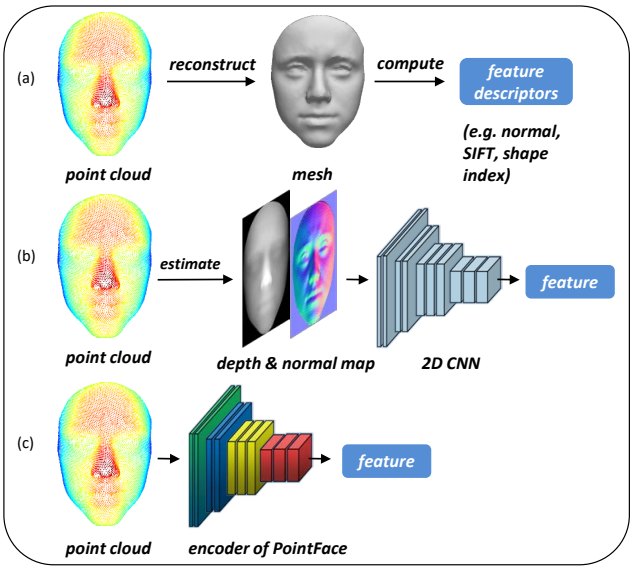

Figure 1. Difference among different pipelines. (a) Hand-crafted feature based methods. (b) 2D CNN based methods. (c) our method.

In 3D FR, both hand-crafted feature based methods and 2D CNNs based methods require data conversions before feature extraction. For instance, many hand-crafted feature based methods [4, 6, 14, 21] need to reconstruct 3D polygon surfaces (also called "mesh") from point clouds or depth images and then compute feature descriptors from the reconstructed surfaces (shown in Figure 1 (a)). 2D CNNs based

*Corresponding author
978-1-6654-3780-6/21/$31.00 ©2021 IEEE

methods [5, 15] firstly convert depth images to point clouds for preprocessing (*e.g*., cropping and dense alignment) and then project the point clouds back to depth images (shown in Figure 1 (b)) as the input of CNNs for feature extraction. While hand-crafted feature based methods cannot effectively handle occlusions caused by large poses, important geometric information may be lost in data conversions for 2D CNN based methods. We argue that a deep learning model that can extract features directly from point clouds of 3D faces is more concise and desirable (shown in Figure 1 (c)). Point clouds, which are independent on reconstruction algorithms and easily accessible in the real world, can also provide geometric information. To the best of our knowledge, our work is the first deep network that extracts features directly from point clouds for 3D FR.

It is worth mentioning that 2D CNNs cannot process point cloud directly, as point clouds are unordered and unstructured, i.e., a specific point cloud can be presented by different orders of a point set. As a pioneering work in deep learning, Qi *et al*. [18] proposed an efficient network, PointNet, to directly process point clouds while achieving permutation invariance, which shows good performance on regular 3D shape classification. Based on PointNet [18], many works [13, 20, 30] proposed to improve local geometric feature learning by tailoring convolutional kernels for point cloud processing.

Since the geometric shape of different categories of 3D objects have significant differences, the PointNet based methods trained on single encoder architecture achieve impressive performance in regular 3D object recognition. However, they are not suitable for 3D FR, as they pay less attention to discriminative features, which is important to distinguish the point clouds of different 3D faces.

To address this issue, inspired by contrastive learning [1, 2], we propose a light-weight framework, called PointFace (shown in Figure 2), to directly learn fine-grained representations from 3D point clouds for 3D FR. PointFace is composed of two weight-shared encoders that extract embeddings (feature vectors) from a pair of point cloud faces. As shown in Figure 3, we construct the encoder with *set abstraction module* [20] and fully connected layer. Since PointFace takes point cloud pair as input to evaluate their similarity in the training stage, we present a pair selection strategy to generate positive and negative pairs to boost training. In order to obtain more discriminative features of 3D faces, we design feature similarity loss to supervise the training of our encoder. Feature similarity loss aims to separate the positive samples from negative samples by a distance margin. Jointly learning the differences among samples and identity information of each sample itself improves the performance of the encoder.

In summary, the main contributions are highlighted as follows:

- We propose PointFace to directly extract features from 3D point cloud for 3D FR. Benefited from contrastive learning, PointFace applies two weight-shared encoders to extract discriminative features of 3D faces.
- To efficiently measure feature similarity among samples and acquire fine-grained face representations, we present a pair selection strategy to boost training and design a feature similarity loss to supervise the training of the encoders.
- Our model achieves state-of-the-art performance on Lock3DFace [31] in terms of rank-one accuracy. In addition, we explore cross-quality scenario on Bosphorus [23] and our model surpasses the prior state-of-the-art Led3D [15] by 1.6%. Finally, comprehensive ablation studies demonstrate the effectiveness of different modules of our PointFace.

2. Related Work

We briefly review related works in the field of 3D face recognition and deep learning based point cloud processing.

3D face recognition. Approaches for 3D FR can be categorized into hand-crafted feature based methods and deep learning based methods. Hand-crafted feature based methods [4, 6, 14, 21] extract features from the key points, curvatures, shape indexes and normals of a 3D face mesh, then identify the face by comparing these features. They achieve impressive performance on high-quality public face datasets, *e.g*. FRGC v2 [17]. Nevertheless, these kind of approaches are sensitive to noises (data collected by 3D scanner in the real world are always noisy) and their generalization ability is limited.

Huge progress has been made in deep learning based 2D face recognition. CNNs [24, 26, 28] have been dominating this area, due to carefully designed network architectures, sophisticated loss functions and massive training data. To utilize 2D CNNs, many researchers transform 3D surfaces to 2D maps and extract features using 2D CNNs for 3D FR. For example, Kim *et al*. [10] used an augmented dataset with 123,325 depth images to fine-tune VGG-Face network, and Gilani [5] enriched training data by generating large scale synthetic 3D faces and fed three-channel geometry maps to VGG-16 network [25]. Recently Mu *et al*. [15] concentrated on low-quality 3D face and proposed a light-weight network to extract features from 2D depth images and normal maps. While [5] and [15] achieved state-of-the-art performance on high-quality and low-quality 3D data respectively, they need to transform 3D data to 2D maps as well.

Deep learning based point clouds processing. In contrast to images, the structure of point cloud is irregular. Deep learning models not only need to find a high density representation from sparse point clouds but also need to

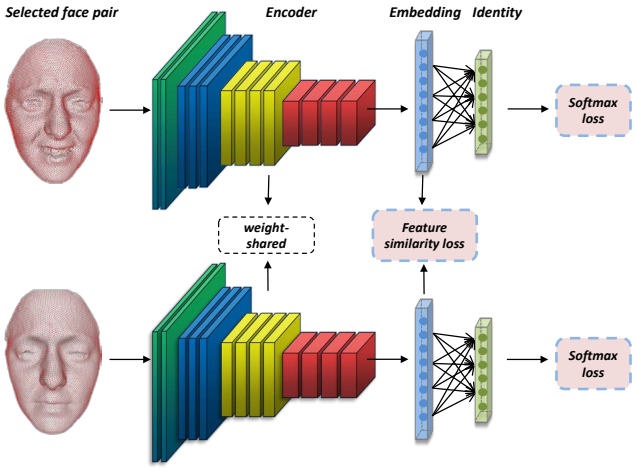

Figure 2. Overview of our PointFace. PointFace directly encodes a pair of facial point clouds into a pair of embeddings for feature similarity measuring and identity prediction. The parameters of upper and bottom branch of encoder are shared.The fully connected layers in the end are weight-shared as well.

achieve scale invariance and permutation invariance. *Volumetric CNNs:* [19] were the pioneers applying 3D convolutional neural networks to voxelized shapes. However, volumetric representation is limited by the resolution of sparse data and computational cost of 3D convolution.

Qi *et al*. [18] was the pioneer that proposed PointNet to directly process point clouds and achieved permutation invariance with a symmetric function. Specifically, it learns point-wise features independently via multi-layer perceptrons (MLPs) and summarizes them via max-pooling to obtain global feature. Since features are learned independently for each point in PointNet [18], the local structural information between points cannot be captured. Therefore, Qi *et al*. [20] proposed a hierarchical network, PointNet++, to capture local geometric information from the neighborhood of each point. Following works [13,30] tried to capture better local features by defining convolution kernels on point clouds.

3. Proposed Approach

In this section, we introduce the overall architecture of our PointFace in Section 3.1. Then we describe our encoder by briefly reviewing *set abstraction* [20] and relation-shape convolution [13] in Section 3.2. Finally, we detail the proposed pair selection strategy and feature similarity loss in Section 3.3.

3.1. Overall Architecture

As shown in Figure 2, given point cloud data for training, we firstly pair them as input of PointFace via our proposed strategy (detailed in Section 3.3). Then, two weight-shared encoders consisting of a stack of set abstraction modules

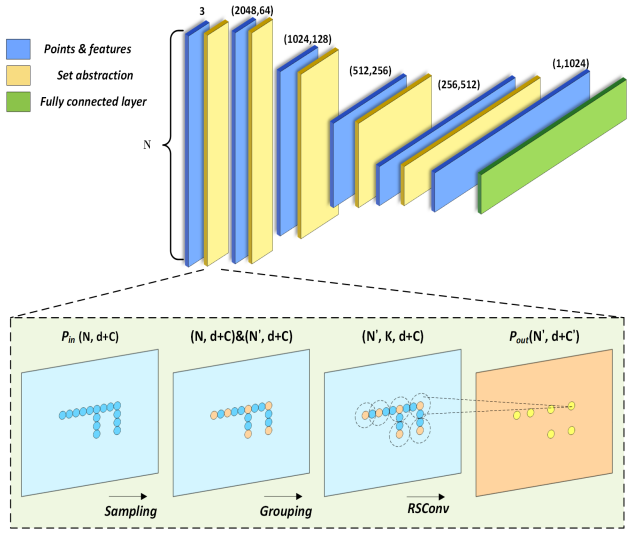

Figure 3. Details of the encoder of PointFace (top) and set abstraction module (bottom). The input of the encoder is point cloud with N points and 3-dimension coordinates. The number in bracket above each layer denotes the number of points and the dimension of corresponding features. The last fully connected layer outputs a 512-dimension embedding. "RSConv" means relation-shape convolution.

(detailed in Section 3.2) encode point cloud pair into two 512-dimension embeddings for identity prediction and feature similarity measurement.

In training phase, we design feature similarity loss to evaluate embedding similarity between two samples so that the encoder is able to distinguish 3D faces from different individuals and compactly cluster features of faces from the same individual. We employ softmax classification loss to supervise the training of encoder as well so that the encoder can learn identity information from each sample itself. Jointly supervised by feature similarity loss and softmax loss, our encoder can obtain more fine-grained representations. In testing phase, we only need embeddings produced by the encoder for identity recognition.

3.2. Encoder

As shown in Figure 3, the encoder of PointFace uses five set abstraction modules to produce features with dimension of 64, 128, 256, 512, 1024 and one fully connected layer to generate features with dimension of 512.

Set abstraction module [20], a module resembles convolution for 2D images, abstracts a set of points to produce a new set with fewer elements, as well as high-level features. Set abstraction module consists of three key layers: *Sampling layer*, *Grouping layer* and *Feature learning layer*.

Given an input point cloud $\{x_i \in R^d\}$ and its corresponding features $\{f_{x_i} \in R^C\}$ for $i = 1, 2, ..., N$, we can use a tensor P_{in} of size $N \times (d + C)$ to represent the input point cloud. *Sampling layer* selects N' points to define the

centroids of local regions of input point cloud P_{in}. i.e., it takes point cloud P_{in} as input and outputs a centroid point tensor P_{cen} of size $N' \times (d + C)$. *Grouping layer* constructs local regions around the centroid points by finding K neighbors of each centroid within a spherical radius. In other words, it takes P_{in} as well as P_{cen} as input and outputs a tensor of size $N' \times K \times (d + C)$.

Feature learning layer encodes and aggregates each local region pattern into a feature tensor of size $N' \times (d+C')$, P_{out}. See Fig. 3 for better understanding. Here, we utilize relation-shape convolution [13] to act as *Feature learning layer* for better local feature learning.

Relation-shape convolution [13] aims to learn weights for each point in a local region from low-level relation prior and aggregate them to obtain condensed representation for the local region.

Specifically, let x_j be a centroid point sampled by *Sampling layer*, $\mathcal{N}(x_j)$ be its neighbor area within a spherical radius r and x_k be any point located in $\mathcal{N}(x_j)$, relation-shape convolution can be formulated as:

$$f_{x_j} = \mathcal{A}(\{\mathcal{M}(f_{x_k}), \forall x_k \in \mathcal{N}(x_j)\}), \tag{1}$$

where $\mathcal{A}$ is a symmetric aggregation function (we use max-pooling here) that helps the convolution achieve permutation invariance and aggregate local features, $\mathcal{M}$ is a function that learns features for all points in $\mathcal{N}(x_j)$. As defined in [13], $\mathcal{M}$ is formulated as:

$$\mathcal{M}(f_{x_k}) = \mathcal{H}(h_{jk}) \otimes f_{x_k}, \tag{2}$$

where $\otimes$ denotes Hadamard product, $\mathcal{H}$ is multi-layer perceptrons (MLPs) that learns convolution kernels (or weights) from low-level relation prior vector h_{jk} and h_{jk} is a prior vector that can be defined flexibly. For instance, we define h_{jk} as 10-dim vector $(x_j, x_k, x_j - x_k, ||x_j - x_k||_2)$ for our tasks. As a consequence, Eq. (1) becomes:

$$f_{x_j} = \mathcal{A}(\{\mathcal{H}(h_{jk}) \otimes f_{x_k}, \forall x_k \in \mathcal{N}(x_j)\}). \tag{3}$$

3.3. Pair selection and Feature Similarity Loss

Pair Selection. To fully explore the advantages of contrastive learning and boost training, we present a novel pair selection strategy to generate positive pairs and negative pairs. Specifically, for each sample called "anchor" in the training data, we randomly select one positive sample from the training data and pair them as input of the network. During the process, we ensure that each positive counterpart is selected only once and every pair is unique. The result is that all samples belonging to an individual are linked together with minimum number of pairs. After PointFace encodes a mini-batch of anchor-positive pairs into a mini-batch of paired embeddings, for each anchor, we pick the hardest negative sample (whose embedding have the closest distance from anchor) from counterpart of other anchors within the mini-batch to group anchor-negative pair. Triplets that contains anchors and their corresponding positive and negative samples within the mini-batch are then used to compute feature similarity loss. In practice, we use all anchor-positive pairs (if they exist), instead of only selecting the hardest positive, to boost convergence. Detailed implementation is shown in Algorithm 1.

Analysis of our Strategy. Generally speaking, in terms of generating pair data, the performance of model can be influenced by two factors [2, 24]: (1) ratio of positive and negative pairs, and (2) pair diversity (anchor shall compare with as many samples as possible).

It is important to balance the positive and negative pairs during training. Empirically, too much negative samples for each anchor will lead to poor performance of model. In our strategy, offline positive-anchor pair grouping guarantees at least one positive sample for each anchor and hardest negative mining ensures that only one hardest negative sample is compared with each anchor in the mini-batch. The number of positive and negative pairs can thus be balanced. In addition, we compare hardest negative mining with semi-hard negative mining [24], and find that hardest negative mining enable our network to perform better on 3D point clouds. i.e., we do not need to pick more negative samples for each anchor in our task. In practice, we find that using all positive samples within a mini-batch can accelerate convergence without degrading performance.

Our strategy maintains pair diversity as well. All positive samples are linked together with minimum number of pairs and hardest negative mining ensures that each anchor will be compared with every positive sample and hard negative sample directly or indirectly as training proceeds.

Feature Similarity Loss. For the purpose of separating positive pairs from negative samples by a margin, inspired by triplet loss [24] and contrastive loss [2], we design feature similarity loss to supervise the training of our encoder. Note that we aim to minimize the similarity among samples from same subjects and maximize the discrepancy among different subjects simultaneously in high-dimension feature space. Specifically, given M triplets with f_i^a, f_i^p and f_i^n $(i = 1, 2, ..., M)$ as L_2 normalized feature vectors of an anchor, positive sample, and negative sample respectively, our feature similarity loss can be described as:

$$L_{sim} = \sum_{i=1}^{M} [D(f_i^a, f_i^p) + m - D(f_i^a, f_i^n)], \tag{4}$$

where m denotes the margin and $D(*, *)$ denotes function that calculates distance between two vectors. Different from triplet loss in [24], we utilize cosine distance, instead of L_2-distance, to evaluate similarity between two vectors. The reasons are: (1) good embeddings of different samples belonging to the same subject should have the same angle in high-dimension feature space [29] and cosine distance as

Algorithm 1 PointFace pipeline

Input: batch size B, number of subjects C, encoder E sample lists $\{S_c\}$ from subject c ($c = 1, 2, ..., C$)

Initialize empty set P
for $c = 1$ to C **do**
  let $length_c$ be the length of S_c
  for $i = 1$ to $length_c$ **do**
    append $(S_c[i], S_c[(i)mod(length_c) + 1])$ to P
  end for
end for
while encoder E is not converged **do**
  generate mini-batch $\{(x_i, x_i^+)\}_{i=1}^B$, $(x_i, x_i^+) \in P$
  for each (x_i, x_i^+) in $\{(x_i, x_i^+)\}_{i=1}^B$ **do**
    perform augmentation for x_i and x_i^+
    $emb_i = E(x_i)$
    $emb_i^+ = E(x_i^+)$
  end for
  Initialize empty set T
  for each emb_i in $\{(emb_i, emb_i^+)\}_{i=1}^B$ **do**
    let emb_i^- be the hardest negative embedding found in $\{emb_j^+\}_{j \neq i}^B$
    append $(emb_i, emb_i^+, emb_i^-)$ to T ▷ in practice, we use all positive samples
  end for
  compute Eq. (5) using T and softmax loss
  update parameters of encoder E
end while
return encoder E

loss function can guide the encoder to achieve this goal, (2) L_2-distance attempts to project different samples from a specific subject to a point in high-dimension feature space, which can not guide network to reach optimal status for our tasks (as our experiments shows). Due to monotonic decrease of cosine function (when angle is in range [0,π/2]) , we reformulate L_{sim} in Eq. (4) as:

$$L_{sim} = \sum_{i=1}^{M} [1 - D_{cos}(f_i^a, f_i^p) + D_{cos}(f_i^a, f_i^n) - m], \quad (5)$$

where $D_{cos}(*, *)$ denotes cosine distance between two feature vectors.

Combining feature similarity loss and softmax classification loss $L_{softmax}$, the total loss for PointFace can be described as:

$$L_{total} = L_{softmax} + \lambda L_{sim}, \quad (6)$$

where λ is a constant to balance each item.

4. Experiments

In this section, we firstly detail the implementation of PointFace in Section 4.1 and databases we use in Section 4.2. Then, we conduct experiments on Lock3DFace [31] for 3D FR in Section 4.3. Moreover, following [15], we explore the cross-quality evaluation on Bosphorus [23] to further validate the generalization ability of our model in Section 4.4. Finally, extensive ablation studies are carried out in Section 4.5.

4.1. Implementation Details

We train our PointFace in an end-to-end fashion. Our network is implemented using Pytorch. We employ Adam [11] optimization for training, with a mini-batch size of 2×32. The learning rate begins with 0.001 and decays with a rate of 0.7 every 20 epochs. Dropout rate (for fully connected layer) is set to 0.4. For feature similarity loss in Eq. (5), we set m to 0.35 and 0.4 for Lock3DFace [31] and Bosphorus [23] respectively, and in Eq. (6), we empirically set λ to 1.

4.2. Datasets

Lock3DFace [31] is a comprehensive database consisting of low-quality 3D faces collected by Kinect V2, which includes 5,671 RGB-D video clips of 509 individuals. The dataset can be divided into five subsets covering variations in expression (FE), neutral face (NU) , occlusion (OC), pose (PS) and time lapse (TM).

Bosphorus [23] is a high-quality database with 4,666 3D faces of 105 individuals, presenting variations in expression, occlusion, and pose.

FRGC v2 [17] consists of 4,007 high-quality 3D face of 466 individuals with expression variations. Facial data are collected by a high-precision 3D Laser scanner.

4.3. 3D Face Recognition on Lock3DFace.

Protocol. We follow the same protocol proposed in [3, 15] for training and testing. Firstly, we randomly select 340 subjects for training and the rest (169) are used for testing. In training set, we sample six frames from each video with an equal interval. In testing set, we also extract six frames for each subject and the first frame from each neutral expression video is taken as gallery samples (resulting in 169 gallery samples) and the remaining frames are used as probes. Secondly, for both training set and testing set, we preprocess all samples by nose-tip based alignment for non-facial area removal [15] and face cropping. After preprocessing, all data are in point cloud format. Finally, we uniformly downsample each preprocessed point cloud to around half size and estimate normal for each point in the point cloud data via *Open3D* [32].

Settings for PointFace. During training, we sample 5,000 points using *farthest point sampling* from each training sample and normalize them to a unit sphere at each epoch. Moreover, during training, we randomly keep the

Table 1. Rank-one recognition rate (%) of 3D FR on Lock3DFace

Method	Input	Lock3DFace					
		FE	NU	OC	PS	TM	Total
VGG-16 [25]	Depth	94.76	99.57	44.68	49.21	34.50	70.58
ResNet-34 [7]	Depth	96.09	99.29	54.91	61.39	45.00	76.56
Inception-v3 [27]	Depth	93.56	98.97	56.98	54.14	42.17	74.44
MobileNet-v2 [22]	Depth	95.74	98.91	61.44	69.92	43.00	79.49
Led3D [15]	Depth	97.62	99.62	68.93	64.81	64.97	81.02
	Depth&Normal	98.17	99.62	78.10	70.38	65.28	84.22
PointFace (Ours)	XYZ	97.93	99.35	77.06	72.03	66.33	84.78
	XYZ&Normal	**98.52**	99.46	**80.67**	**73.69**	**67.00**	**87.18**

FE: expression. NU: neutral face. OC: occlusion. PS: pose. TM: time lapse.

original data or augment the data with one of following approaches: random anisotropic scaling in the range [-0.66, 1.5], random translation in the range [-0.2, 0.2] and random rotation in the range [-90°, 90°] on yaw-direction and [-30°, 30°] on pitch-direction. We train PointFace from scratch. Two types of input data: coordinates (XYZ) and coordinates with normal (XYZ&Normal) of point clouds are tested.

For evaluation, firstly, we uniformly sample 5,000 points from each testing sample and normalize them to a unit sphere. Secondly, we extract embeddings of probes and galleries from the encoder. Then, we match the feature of each probe with all identities in the gallery. At last, the identity of the probe is assigned as that of gallery who has the minimum cosine distance.

Settings for 2D CNN counterparts. For comparison, we train several representative 2D CNNs [7,22,25,27] using depth images acquired by the preprocessing pipeline [15] following the same protocol. Besides, we use FRGC v2 [17] and Bosphorus [23] to pre-train these 2D CNNs to obtain better performance. Furthermore, we use "random crop" technique for data augmentation in training phase and match features based on cosine distance in evaluation phase.

Results. Table 1 reports the performance of PointFace and other typical 2D CNNs on low-quality Lock3DFace dataset. Note that the performance of state-of-the-art Led3D is reported in [15]. According to Table 1, our PointFace outperforms the Led3D by 3.76% in terms of rank-one recognition accuracy when both only use depth information (PointFace is trained only with XYZ of point clouds and Led3D is trained with depth images). Besides, we can see that our model reach relatively higher rank-one accuracies on subset OC (77.06%) and PS (72.03%) than Led3D. PointFace trained with XYZ from scratch surpasses all typical 2D CNNs on subset FE, OC, PS and TM. It illustrates that when compared with 2D CNNs based methods, PointFace can acquire more geometric information robust to occlusions and pose changes. Surprisingly, we also see that PointFace trained with only XYZ has better performance (84.78%) than Led3D trained with both depth images and normal maps (84.22%).

We now estimate normal for each point in the point cloud data and concatenate it with XYZ as an extra input, i.e., now the input becomes six channels. As extra information is introduced, the performance of both Led3D and our PointFace improves. For example, the total accuracy of Led3D and PointFace increase from 81.02% to 84.22% and 84.78% to 87.18%, respectively. Again, the total accuracy of our approach (87.18%) is about 2.96% higher than that of Led3D, when normal is included as the input. In particular, PointFace achieves much higher accuracies on the subsets of OC (80.67%) and PS (73.69%).

4.4. Cross-quality Evaluation

Protocol and setting. Since low-quality and cheap sensors are more affordable than high-precision sensors, a common setting in the real world is that high-quality data and low-quality data are used as gallery samples and probe samples [15] respectively. As the quality of probes and galleries is significantly different, we call such testing as cross-quality evaluation.

To see whether our model has good performance on cross-quality evaluation, we utilize FRGC v2 as training set and evaluates PointFace on Bosphorus. Thanks to Mu *et al*. [15], we acquire the enriched FRGC v2 dataset whose size is increased by virtual ID generation method in [5] for model training. In the training stage, we generate low-quality data from high-quality dataset FRGC v2 by adding Gaussian noise $N(0, 16)$ to the depth values (values along axis Z) of point cloud data, following what presented in [15].The training set contains both high-quality and low-quality point clouds of 1,000 identities. For testing set, we use the face with neutral expression of each individual in Bosphorus as gallery (resulting in 105 galleries) and the remaining faces are converted into low-quality data as probes by the same operation. In addition to the setting of HL (high-quality galleries and low-quality probes), we have evaluated our model on the setting of LL where both galleries and probes are converted into low-quality as well. Note that we preprocess the point cloud data for both datasets by cropping the facial area, downsampling and estimating normal as well.

We observe that some faces in Bosphorus have occlu-

sions due to large pose changes, while faces in FRGC v2 are all frontal faces. So, we utilize *hidden point removal* proposed in [9] to generate the same kind of occlusions caused by pose changes for data augmentation in training phase. In testing phase, we extract embeddings of samples from the encoder and the identities of probes are decided based on the minimum cosine distance between embeddings of probes and galleries.

Results. Table 2 shows the results of cross-quality 3D FR. Note that the results of Inception-v2 [8] and Led3D [15] are reported in [15]. Our PointFace achieves 91.86% rank-one accuracy in LL setting and 92.96% in HL setting, surpassing the performance of Led3D in both scenarios. The results suggest that our model have relatively good generalization ability and is robust to data quality changes.

Table 2. Rank-one recognition rate (%) of cross-quality evaluation on Bosphorus (HL: high-quality in gallery and low-quality in probe; LL: low-quality in both gallery and probe).

Method	HL	LL
Inception-v2 [8]	78.56	77.23
Led3D [15]	91.27	90.70
PointFace (Ours)	**92.96**	**91.86**

4.5. Ablation Studies

In this section, all ablation studies are conducted on Lock3DFace using protocol and training setting described in Section 4.3. Note that we only use coordinates of point clouds as input in these experiments.

Ablation study on architecture. Table 3 summarizes the results of ablation study. The baseline (model A) is trained without relation-shape convolution, but with a two-layer MLP as function $\mathcal{M}$ in Eq. (1). Besides, the baseline, similar to PointNet++ [20], is just a single encoder trained with softmax classification loss. Model B is also a single encoder using relation-shape convolution as function $\mathcal{M}$ in Eq. (1). Model C replaces the relation-shape convolution in PointFace with a two-layer MLP. Model D is PointFace trained without softmax classification loss.

The baseline (model A) only gets rank-one recognition accuracy of 75.31%. Compared with baseline, relation-shape convolution improve the accuracy of model B to 80.80%. The improvement demonstrates that local feature learner based on convolution obtains better local region representations than simple MLPs. Employing two weight-shared encoder trained with feature similarity loss (model C) improves the rank-one score to 80.10% (4.79% higher than the baseline), i.e., our proposed architecture has stronger capability to distinguish point cloud faces from different subjects. Although PointFace trained without softmax loss (model D) doesn't perform well in challenging subsets (OC, PS and TM), it can still learn features from neutral faces. Moreover, our PointFace (model E) outperforms model B and model D by 3.97% and 11.12% respectively, proving that the two loss functions jointly help our encoder to learn discriminative feature.

Table 3. Ablation studies of PointFace in terms of rank-one recognition rate (%) on Lock3DFace (FSloss: feature similarity loss; Sloss: softmax loss; RSConv: relation-shape convolution).

Model	RSConv	Sloss	FSloss	Lock3DFace					
				FE	NU	OC	PS	TM	Total
A		✓		94.45	98.80	64.45	48.40	46.00	75.31
B	✓	✓		96.41	99.13	72.59	59.38	57.17	80.80
C		✓	✓	96.17	98.43	70.93	61.86	47.17	80.10
D	✓		✓	92.35	97.29	63.05	53.26	23.67	73.66
E	✓	✓	✓	**97.93**	**99.35**	**77.06**	**72.03**	**66.33**	**84.78**

Comparison with different pair selection strategies. We tested four different pair selection strategies in this evaluation. The rank-one accuracy and the approximate number of epochs for our PointFace to converge are shown in Table A in the supplementary. As shown in the table, our proposed strategy achieves the highest accuracy, i.e. 84.78%, among all strategies.

4.6. Model Complexity

Table C in supplementary summarizes the space (number of parameters in the network) and time (multiply-add operations/sample) costs of PointFace in comparison with other representative 2D CNN models. Among them, PointFace requires the least parameters (1.8M). In addition, PointFace is about 3× and 4.5× more efficient than ResNet-34 [7] and Inception-v3 [27], in terms of multiply-add operations per sample.

5. Conclusion

In this paper, we propose a framework, called PointFace, to directly process point cloud faces for 3D face recognition. Inspired by contrastive learning, we utilize two weight-shared encoders to extract features and design feature similarity loss to supervise the training of our encoder. Jointly learning the differences among samples and identity information from each sample itself enables our encoder to obtain discriminative face representations. Extensive experiments on Lock3DFace and Bosphorus show that the proposed PointFace can achieve state-of-the-art performance, which is also better than 2D CNNs based methods. Moreover, ablation studies also demonstrate effectiveness of our weight-shared twin-encoder architecture and feature similarity loss. We hope our work can shed the light on point cloud based 3D face recognition.

6. Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 62076163 and Grant 91959108; in part by the Shenzhen Fundamental

Research Fund under Grant JCYJ20190808163401646 and Grant JCYJ20180305125822769; and in part by the Tencent "Rhinoceros Birds"—Scientific Research Foundation for Young Teachers of Shenzhen University.

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.

[2] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 2, pages 539–546, Los Alamitos, CA, USA, jun 2005. IEEE Computer Society.

[3] J. Cui, H. Zhang, H. Han, S. Shan, and X. Chen. Improving 2d face recognition via discriminative face depth estimation. In *ICB*, pages 140–147, 2018.

[4] H. Drira, B. A. Boulbaba, S. Anuj, M. Daoudi, and R. Slama. 3d face recognition under expressions, occlusions, and pose variations. *TPAMI*, 35(9):2270–2283, 2013.

[5] S. Z. Gilani and A. Mian. Learning from millions of 3d scans for large-scale 3d face recognition. In *CVPR*, pages 1896–1905, 2018.

[6] S. Z. Gilani, A. Mian, F. Shafait, and I. Reid. Dense 3d face correspondence. *TPAMI*, 40:1584–1598, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[9] S. Katz, A. Tal, and R. Basri. Direct visibility of point sets. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 24–es, New York, NY, USA, 2007. Association for Computing Machinery.

[10] D. Kim, M. Hernandez, J. Choia, and G. Medioni. Deep 3d face identification. In *IJCB*, pages 133–142, 2017.

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[12] H. Li, Y. W. Di Huang, Jean-Marie Morvan, and L. Chen. Towards 3d face recognition in the real: A registration-free approach using fine-grained matching of 3d keypoint descriptors. *International Journal of Computer Vision*, 113:128–142, 2014.

[13] Y. Liu, B. Fan, S. Xiang, and C. Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, pages 8887–8896, 2019.

[14] A. S. Mian, M. Bennamoun, and R. Owens. Keypoint detection and local feature matching for textured 3d face recognition. *International Journal of Computer Vision*, 79(1):1–12, 2008.

[15] G. Mu, D. Huang, G. Hu, J. Sun, and Y. Wang. Led3d: A lightweight and efficient deep approach to recognizing low-quality 3d faces. In *CVPR*, pages 5766–5775, 2019.

[16] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 41.1–41.12. BMVA Press, September 2015.

[17] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge. In *CVPR*, volume 1, pages 947–954 vol. 1, 2005.

[18] C. R. Qi, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 77–85, 2017.

[19] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, pages 5648–5656, 2016.

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30, pages 5099–5108. Curran Associates, Inc., 2017.

[21] C. Samir, A. Srivastava, and M. Daoudi. Three-dimensional face recognition using shapes of facial curves. *TPAMI*, 28(11):1858–1863, 2006.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[23] A. Savran, N. Alyüz, H. Dibeklioğlu, O. Çeliktutan, B. Gökberk, B. Sankur, and L. Akarun. Bosphorus database for 3d face analysis. In *Biometrics and Identity Management*, pages 47–56. Springer Berlin Heidelberg, 2008.

[24] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[26] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 1988–1996. Curran Associates, Inc., 2014.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.

[28] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[29] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, pages 5265–5274, 2018.

[30] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM ToG*, 38(5), Oct. 2019.

[31] J. Zhang, D. Huang, Y. Wang, and J. Sun. Lock3dface: A large-scale database of low-cost kinect 3d faces. In *ICB*, pages 1–8, 2016.

[32] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.