PAY BY SHOWING YOUR PALM: A STUDY OF PALMPRINT VERIFICATION ON MOBILE PLATFORMS

Yingyi Zhang[1], *Lin Zhang*[1,*], *Xiao Liu*[1], *Shengjie Zhao*[1], *Ying Shen*[1,*], *Yukai Yang*[2]

[1]School of Software Engineering, Tongji University, Shanghai, China
[2]Department of Statistics, Uppsala University, Uppsala, Sweden

ABSTRACT

With the fast development of smart mobile devices, mobile phones have gradually become an indispensable part of people's lives. Many biometric technologies based on mobile platforms have also developed rapidly, such as face verification and fingerprint recognition. However, the great potential of palmprint has been neglected. In this paper, we conducted a thorough study of palmprint verification on mobile devices for the first time. Firstly, we established an annotated, palmprint dataset named ***MPD***, which was collected by multi-brands phones in two different sessions. As the largest dataset in this field, ***MPD*** contains 16,000 palm images from 200 subjects. Secondly, we built a DCNN-based palmprint verification system named ***DeepMPV*** for mobile platforms. The efficiency and performance of our system have been corroborated on our collected dataset. The labelled dataset and the source code are publicly available at https://cslinzhang.github.io/deepmpv/.

Index Terms— Palmprint recognition, palmprint verification, mobile device, deep convolutional neural networks

1. INTRODUCTION

Nowadays, mobile phones have become a necessity in people's lives and an important tool for people to deal with their daily work. With the increasing concern about networking and mobility security, the demand for reliable user authentication technology has grown greatly [1]. Therefore, high-precision identity authentication based on mobile phones has long been one of the main focuses in the research field, such as fingerprint unlocking [2] and face payment [3].

However, ignored by many, palmprint is also a great choice for personal identity authentication. In fact, compared with face verification, palmprint verification is a non-invasive way of authentication, which is much easier for users to accept. Besides, fingerprint recognition often requires highly sensitive sensors, but palmprint images can be recognized using built-in cameras of mobile phones. Moreover, some people do not have clear fingerprint, while palmprint has more abundant feature information and can be recognized under low resolution conditions. As an important member of the biometrics family, palmprint has a variety of required characteristics, such as strong uniqueness, durability, user friendliness and so on.

In this paper, we perform a thorough study of palmprint verification on mobile devices.

*Corresponding author. Email: {cslinzhang, yingshen}@tongji.edu.cn

2. RELATED WORK AND OUR CONTRIBUTIONS

2.1. Related work

Palmprint [4, 5, 6, 7] refers to the skin pattern on the inner surface of the palm, which mainly includes two characteristics: palm frictional ridges (ridges and valleys) and palm flexion creases (discontinuities in epidermal ridges). To verify palmprint on mobile devices, there are two key components, region of interest (ROI) extraction and palmprint verification. In former literature, several different approaches have been proposed for palm ROI extraction. In [8], Han *et al.* used a predefined preview frame on screen to align the position between hand and camera before ROI extraction. Later in [9], Brown used three gaps between fingers to extract ROIs after separating palms from background. Then Aoyama *et al.* [10] used a radial distance function to detect specific points of finger gap and extract the ROIs. In later work of Aykut *et al.* [11], they used an advanced model based segmentation method named *AAM* to segment hand and extract palm ROIs. Franzgrote *et al.* [12] developed a hand orientation normalization method to extract ROI on mobile devices. Later, Zhao *et al.* in [13] proposed an approach to align palms using a projective transformation model that estimated from matched SIFT feature points.

In the field of palmprint verification, great efforts have been made on extracting feature from palmprint ROIs. In [14], Zhang *et al.* used multiple 2-D Gabor filters to extract orientation information from palm lines and stored them in a feature vector called the competitive code (CompCode). Then in [15], Jia *et al.* proposed a novel robust line orientation code for palmprint verification. In the same period, Iitsuka *et al.* in [16] used two-dimensional (2D) phase information to represent palmprint feature.

2.2. Our motivations and contributions

Through the investigation of relevant literature, we find that there is still a big gap in at least two aspects.

Firstly, up to now, there is no large-scale publicly available dataset of palmprints with complex backgrounds, which is collected by mobile phones and labelled carefully. It is essential for researchers to design and compare the performance of palmprint verification algorithms based on such a dataset.

Secondly, there is still room for improvement in the performance of palmprint verification systems on mobile devices, mainly reflected in two aspects: palm ROIs extraction and palmprint verification. (1) In fact, vision-based palm ROIs extraction is a mission full of challenge. Because the illumination and background of palmprint photographs are very different and complex, it is difficult to locate palmprint ROI in photos. To solve this problem, the existing palmprint verification systems on mobile devices need to segment hand as preprocessing operation, or request the user to place the hand in the designated position in front of the camera. The former is time-consuming and error-prone, while the latter is not user-friendly. Moreover, the current proposed palm ROI extraction methods are difficult to have the characteristics of scale and rotation invariance. (2) Actually, in the phase of palmprint feature extraction, it is hard to distinguish whether two palmprint ROIs belong to the same hand due to the position offset or the difference of palm posture between two ROIs. Furthermore, if the hand is not parallel to the camera while taking photos, the palmprint photos will have some angular offset, and existing palmprint verification algorithms cannot indentify them as matching success. Therefore, how to effectively and accurately match palmprints on mobile phones remains a huge challenge.

In this work, we attempt to fill the research gaps to some extent. Our contributions in this paper are summarized as follows:

(1) To facilitate the study of palmprint verification, we have established a large-scale palmprint dataset named ***MPD*** and will make it publicly available. ***MPD*** comprises 16,000 palm images collected from multi-brands mobile phones and all the images are manually labelled with care. Such a dataset can be employed for training and testing new palmprint verification algorithms. Please refer to Sect. 4 for more details about this dataset.

(2) We proposed a data-driven learning-based approach named ***DeepMPV*** to verify palmprints on mobile devices. Given a palm image, we used a pre-trained detector based on PeleeNet [17] to detect the finger-gap-points at first. Finger-gap-points are defined as the midpoint at the junction of two fingers(yellow circles in Fig. 1). After that, we used the pre-trained MobileNetV2 [18] to classify the correct point pairs (A, B), which are labelled in Fig. 1. For each image, point A is the point located at the middle of gaps between index and middle fingers, and point B is the one between ring and little fingers. According to the correct point pair (A, B), we can construct the local coordinates for palms(blue arrows in Fig. 1). Then, we can extract palmprint ROIs (labelled as red square in Fig. 1) from palm images. Next, we used siamese network [19] to match palmprints. ***DeepMPV*** can work well on mobile phones, and it can match palmprints with slight position offset and angular deflection. Its efficiency and performance have been thoroughly evaluated in experiments.

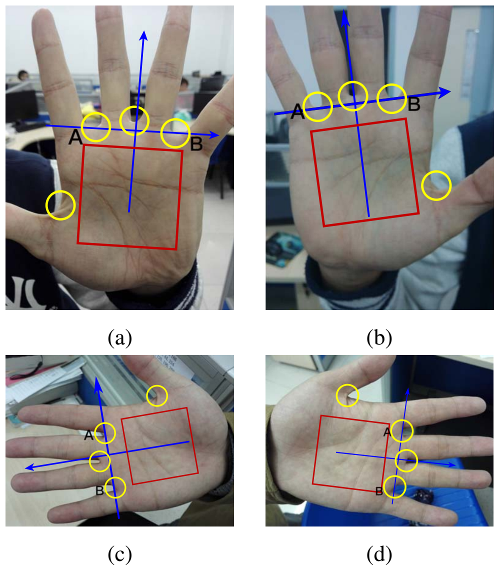

Fig. 1: (a), (b), (c) and (d) are examples of palm images in different backgrounds and different angles. (a) and (c) are taken from left hands while (b) and (d) are taken from right hands. Yellow circle marks indicate the positions of finger-gap-point; blue arrows represent the local coordinates; and red squares indicate the ROIs of palmprints.

The remainder of this paper is organized as follows: Sect. 3 states the details of our dataset ***MPD***. Sect. 4 introduces our novel approach ***DeepMPV*** for palmprint verification. Experimental results are presented in Sect. 5. Finally, Sect. 6 concludes the paper.

3. *DEEPMPV*: A LEARNING-BASED APPROACH FOR PALMPRINT VERIFICATION

In this section, *DeepMPV* will be presented in detail. It is designed to verify palmprints on mobile devices, and is consisted of three key components (as shown in Fig. 2), including finger-gap-point detection, point pair pattern classification and palmprint verification.

3.1. Finger-gap-point detection

For a given palm image, we need to detect all the finger-gap-points on it at first. Having investigated the literature, we find that PeleeNet is a state-of-the-art general-purpose object

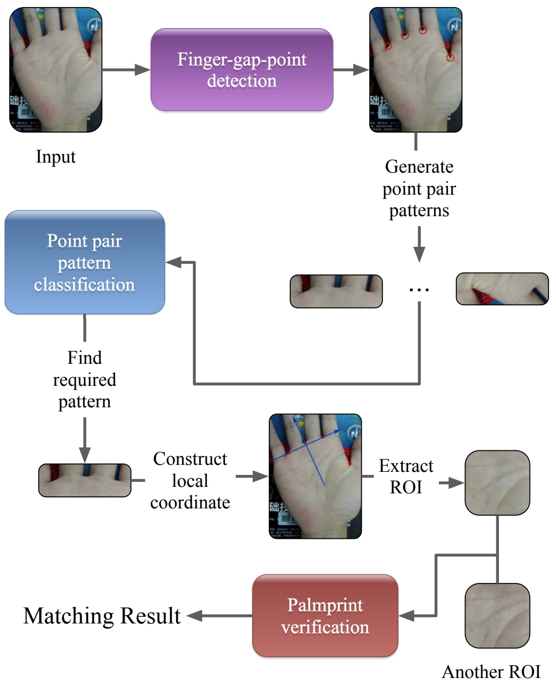
Finger-gap-point detection
Input
Generate point pair patterns
Point pair pattern classification
Find required pattern
Construct local coordinate
Extract ROI
Palmprint verification
Matching Result
Another ROI

Fig. 2: The system flowchart of *DeepMPV*.

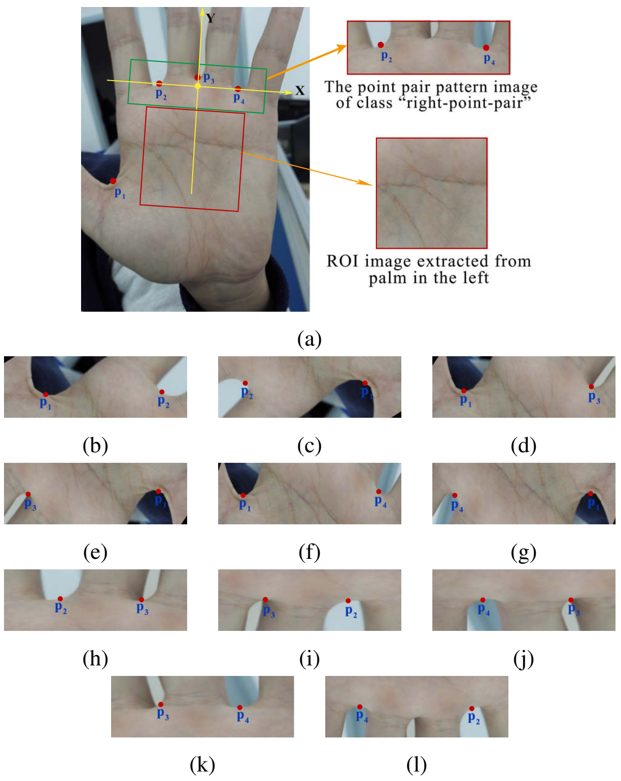
Y
X
The point pair pattern image of class "right-point-pair"
ROI image extracted from palm in the left
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)

Fig. 3: For each palm image, we can obtain 12 point pair pattern images. Only the point pair pattern image in (a) is the one belongs to class "right-point-pair"; and (b)-(l) belong to class "wrong-point-pair".

detector based on DCNN for mobile devices. Compared with MobileNet+SSD [20, 21] and Tiny Yolo V2 [22] , PeleeNet has higher speed and accuracy in real-time object detection on mobile devices. Hence, our finger-gap-point detector $\boldsymbol{D}$ is based on PeleeNet.

To train the detector $\boldsymbol{D}$, we need to prepare training samples. On a given palm image, the position of all its finger-gap-points were manually marked. For each finger-gap-point $\mathbf{p}_i$, a square box of size $s \times s$ centered on $\mathbf{p}_i$ is regarded as the ground-truth bounding box of $\mathbf{p}_i$. The box size s is decided by the length of point pair (A, B) in Fig. 1 as $s = \frac{1}{2} \times \|AB\|$, so that our detector $\boldsymbol{D}$ can have the characteristics of scale invariance. The details of $\boldsymbol{D}$'s training set will be shown in Sect. 4. And in Sect. 5, we will quantitatively evaluate the performance of $\boldsymbol{D}$.

3.2. Point pair pattern classification

After applying the finger-gap-point detector $\boldsymbol{D}$ on the test image, points whose confidence scores greater than δ will be considered as finger-gap-points. Suppose that $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$ are finger-gap-points detected in a palm image, as Fig. 3a shown. After permutation and combination, we obtained 12 patterns of point pairs. For each pattern, we can construct a local coordinates. It's worth noting that $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$ and $\overrightarrow{\mathbf{p}_2\mathbf{p}_1}$ are different point pair patterns. Take $\overrightarrow{\mathbf{p}_2\mathbf{p}_4}$ as an example, the local coordinate of this point pair pattern is established as illustrated in Fig. 3a , which takes the midpoint of $\mathbf{p}_2$ and $\mathbf{p}_4$ as its origin, and $\overrightarrow{\mathbf{p}_2\mathbf{p}_4}$ as its X-axis. Its Y-axis can be consequently determined. In this coordinate system, we defined a rectangular region $\mathbf{R}$, which is symmetric both to the X-axis and the Y-axis. For $\mathbf{R}$, its side length along the X-axis is set as $\|\mathbf{p}_2\mathbf{p}_4\| + \Delta$, and its side length along the Y-axis is set as Δ, too. We set Δ as s, which is the size of the boxes we detected using detector $\boldsymbol{D}$ in Sect. 3.1. After the definition of region $\mathbf{R}$, we can generate 12 pattern images as illustrated in Fig. 3 by extracting the image region covered by $\mathbf{R}$, normalizing it to the size $h \times w$, and regard the extracted image patch as the local image pattern defined by $\overrightarrow{\mathbf{p}_2\mathbf{p}_4}$.

For a given image, there are 12 patterns now. We took the correct point pair pattern $\overrightarrow{\mathbf{p}_2\mathbf{p}_4}$ as class "right-point-pair" (Fig. 3a), and other point pair patterns as class "wrong-point-pair" (Fig. 3b-3l). Only one pattern (Fig. 3a) is the correct pattern in these 12 point pair patterns, which is $\overrightarrow{\mathbf{p}_2\mathbf{p}_4}$ in Fig. 3a. Considering that $\boldsymbol{C}$ will be transplanted to mobile phones, we decided to train $\boldsymbol{C}$ based on MobileNetV2. The details of $\boldsymbol{C}$'s training set will be shown in Sect. 4. In Sect. 5, we will quantitatively evaluate its performance.

When we find the correct point pair pattern for a test image, we can set the local coordinate system of the correct point pair pattern as the local coordinate system of the test image.

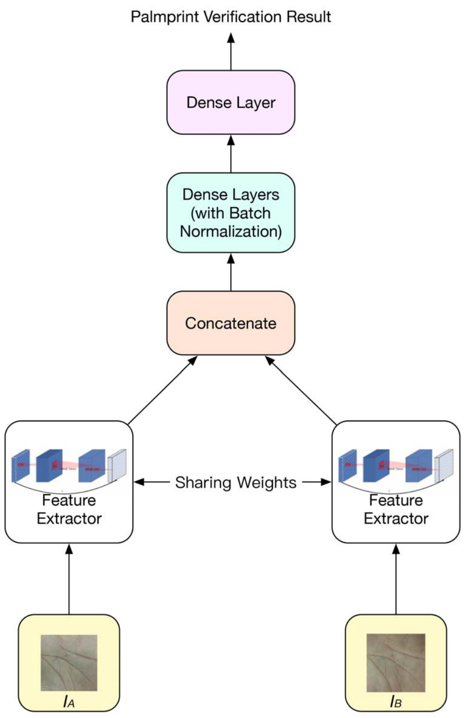

Fig. 4: The architecture of our customed SiameseMobileNetwork. I_A and I_B are two input palm ROI images.

Take the palm image in Fig. 3a as example, we defined the region of interest as *ROI* which is symmetric both to the X-axis and the Y-axis. For *ROI*, its side length along both axises are set as s_R. Finally, we can extract the palm ROI from the palm image by extracting the image region covered by **ROI**.

3.3. Palmprint verification

After palm ROI extraction, there is a pair of palmprint ROIs waiting for verification. In traditional methods, palmprint features are usually extracted to calculate matching scores. To extract feature of ROIs better, we designed a siamese network named ***SiameseMobileNet***. Specifically, ***SiameseMobileNet*** takes a pair of palmprint ROIs as input, and these two input images will be sent to the subnet respectively. Each subnet is a MobileNetV2, sharing weights with each other. Using these subnets, we can get two feature vectors of the size 1280×1 from input ROI images. After that, two feature vectors will be concatenated and sent to three fully connected layers with normalization [23] to reduce the size. Finally, the feature vector will be sent to a fully connected layer, and the result will be given as a matching score. We took scores less than 0.5 as matching success, otherwise palmprint matching fail. The details of this siamese network are shown in Fig. 4, and details of ***SiameseMobileNet***'s training set will be shown in Sect. 4. In Sect. 5, we will quantitatively evaluate its performance.

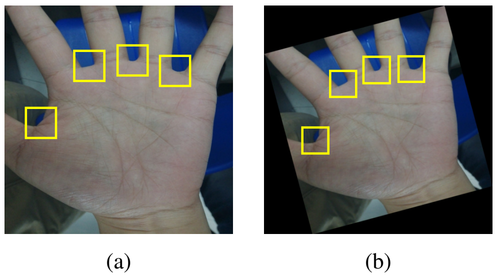

Fig. 5: In the phase of preparing training samples, to make the finger-gap-point detector rotation invariant, each original labelled image was rotated to generate a set of its rotated versions. (a) is the original labelled image, and (b) is its rotated version generated from (a).

4. *MPD*: A LARGE-SCALE PALMPRINT DATASET COLLECTED BY MOBILE PHONES

In order to provide a reasonable performance evaluation benchmark for palmprint verification on mobile platforms, we have established and released a large-scale dataset named *MPD*, in which palm images were collected from a variety of backgrounds and lighting environments. For the purpose of eliminating the influence of camera parameters of different brands of mobile devices, we used two kinds of smartphones: Huawei and Xiaomi. To avoid the influence of season or time on photographs, we took 2 rounds of palm image collection with the same two mobile phones of the same group of people according to the same standard half in 2 periods. *MPD* comprises 16,000 palmprint images from 200 sessions (400 different hands). We have kept 10 photos of one's hand with each phones and each time period and labelled all 16,000 palm images in our dataset.

In the training stage for finger-gap-point detector ***D***, in order to make it be scale invariant, we augmented the training set by rotating each original labelled image to generate a number of its rotated versions as shown in Fig. 5. In detail, for a given original labelled image **I**, we could gain **N** rotated labelled images from it with rotate angle $\theta_N = \frac{360}{\mathbf{N}}$. To make the operation of rotation easier, we resized the original labelled images to the fixed size $s_f \times s_f$. Fig. 5a is an original labelled image and Fig. 5b is its rotated version generated by Fig. 5a with the rotation of 15 degrees. We separated the augmented dataset according to the following ratio: training set:validation set:test set = 8:1:1.

In the training phase of point pair pattern classifier ***C***, based on the labelled data, we can obtain a dataset named *dotPair* comprising all the point pair pattern images. As shown in Fig. 3, the ratio of sample number of class "right-point-pair" to sample number of class "wrong-point-pair" is 1:11. To balance the number of samples in two classes, we reserved the samples of class "right-point-pair' and randomly select the same number of samples in class "wrong-point-

Table 1: Training settings

DCNN structure	framework	Learning rate	batch size	epoch
PeleeNet	caffe	0.0005	32	16
MobileNetV2	keras	0.01	64	5
SiameseMobileNet	keras	0.01	64	5

Table 2: Evaluation result of finger-gap-point detection

DCNN structure	mAP (%)	Speed (ms/image)
Tiny Yolo v2	98.63	50
MobileNetV1+SSD	99.57	45
PeleeNet	99.99	43

pair". Finally we separated them according to the following ratio: training set:validation set:test set = 8:1:1.

In the training stage for ***SiameseMobileNet***, based on labelled data, we generated *ROI* dataset with 16,000 images from 400 hands. We regarded ROI image pairs from the same hand as positive samples, and ROI images pairs from different hands as negative samples. In order to balance the number of positive and negative samples, we kept all positive samples and randomly select the same number of negative samples. We picked 160 sessions as training set, 20 sessions as validation set and 20 sessions as test set.

5. EXPERIMENTAL RESULTS

5.1. Training settings

The training settings of finger-gap-point detector ***D***, point pair pattern classifier ***C*** and ***SiameseMobileNet*** are shown in Table 1.

5.2. Performance evaluation of finger-gap-point detection

In our palmprint verification system ***DeepMPV***, finger-gap-point detection is a crucial step. In this experiment, we evaluated the performance of finger-gap-point detector ***D*** and two DCNN-based methods in the field of object detection, Tiny Yolo v2 and MobileNet+SSD. From Table 2, we can see that the efficiency and performance of model based on PeleeNet is better than the one based Tiny Yolo v2. Hence, we used the PeleeNet-based model to detect finger-gap-points in ***DeepMPV***.

5.3. Performance evaluation of point pair pattern classification

In ***DeepMPV***, when the finger-gap-points are detected, we need to classify each point pair pattern defined by a pair of finger-gap-points as one of the two predefined classes. To achieve this goal, we used the classifier based on MobileNetV2 to classify patterns. Considering that this classifier ***C*** will be transplanted to mobile devices, we evaluated the performance of classifiers based on MobileNetV2 architecture with different width in this experiment.

As the width of MobileNetV2 architecture decreases, the number of channels in each layer decreases, and the model becomes lighter and more suitable for running on the mobile phone. From Table 3, we can know that when the model becomes lighter, its performance does not decline too much, and some even perform better. The lightest model in Table 3 have the best performance of 99.70% accuracy. Moreover, the lighter model can process pictures faster, which improves the efficiency of our system on the premise of guaranteeing accuracy.

Table 3: Performance of pre-trained MobileNetV2 models with different width

Width Multiplier	Accuracy-Top1 (%)	Speed (ms/image)
1.00	97.75	36
0.75	99.62	33
0.50	98.47	29
0.35	99.70	26

Table 4: Performance of pre-trained SiameseMobileNet models with different width and output shape

Width Multiplier	Output Shape	Accuracy-Top1 (%)	Speed (ms/image)
1.00	512	94.42	76
1.00	128	93.03	73
0.75	128	92.19	67
0.50	128	91.54	60
0.35	128	89.91	57

5.4. Performance evaluation of palmprint verification

When we find the correct point pair in palm images, we can extract ROIs by constructing local coordinate. In this experiment, we tried to find the best model for palmprint verification by changing the value of width multiplier and the output shape of fully connected layers. As we can see in Table 4, models with lager output shape have better performance. Besides, when the model becomes lighter, the accuracy of palmprint verification decreases and the speed of image processing becomes faster. However, when the width of the model is reduced by half, its accuracy is reduced by around 2%, while its speed is increased a lot. It is appropriate to use the model with half width in the development of mobile phone applications during practical utilisation.

Table 5: Performance of SiameseMobileNet and CompCode

Feature Extractor	Accuracy(%)
MobileNetV2(alpha=0.35)	89.91
CompCode	81.12

Furthermore, we also compared the performance of SiameseMobileNet and other palmprint verification methods like CompCode in our dataset ***MPD***. From Table 5, we can see that the performance of the worst-performing model in SiameseMobileNet is also much better than that of CompCode.

6. CONCLUTION AND FUTURE WORK

In this paper, we made two major contributions to the field of vision-based palmprint verification. Firstly, we collected and labelled a large-scale palmprint dataset including 16,000 palm images, which is the largest one in this field. And we have made it publicly available. Such a dataset will for sure benefit the study of palmprint verification. Secondly, we proposed a DCNN-based solution for palmprint verification on mobile platforms. Its high efficiency and performance have been corroborated by experiments. In near future, we will try to refine our DCNN-based palmprint verification solution and to continuously enlarge MPD to include more palmprint samples.

7. ACKNOWLEDGMENT

This research was funded in part by the Natural Science Foundation of China under Grant No. 61672380, in part by the National Key Research and Development Project under Grant No. 2017YFE0119300, and in part by the Fundamental Research Funds for the Central Universities under Grant No. 2100219068.

8. REFERENCES

[1] A. A. Ross, K. Nandakumar, and A. K. Jain, "Handbook of biometrics," *Springer*, 2008.

[2] Huawei Co., "Huawei mate 20 pro," `https://consumer.huawei.com/en/phones/mate20-pro/`, In-screen Fingerprint.

[3] Apple Co., "Face id: Your face is your password.," `https://www.apple.com/iphone-xs/face-id/`, Apple Face ID.

[4] D. Zhang, W. Kong, J.. You, and M Wong, "Online palmprint identification," *IEEE Trans. PAMI*, vol. 25, no. 9, pp. 1041–1050, 2003.

[5] A. K. Jain and J. Feng, "Latent palmprint matching," *IEEE Trans. PAMI*, vol. 31, no. 6, pp. 1032–1047, 2009.

[6] L. Zhang and H. Li, "Encoding local image patterns using riesz transforms: With applications to palmprint and finger-knuckle-print recognition," *Image Vis. Comput.*, vol. 30, no. 12, pp. 1043–1051, 2012.

[7] L. Zhang, Y. Shen, H. Li, and J. Lu, "3D palmprint identification using block-wise features and collaborative representation," *IEEE Trans. PAMI*, vol. 37, no. 8, pp. 1730–1736, 2015.

[8] Y. Han, T. Tan, Z. Sun, and Y. Hao, "Embedded palmprint recognition system on mobile devices," in *ICB*. Springer, 2007, pp. 1184–1193.

[9] N. Brown, "Mobile verification by palmprint biometrics," `https://pdfs.semanticscholar.org/6daf/52b66c179a0e8862ceb692d410e36f6783b0.pdf`.

[10] S. Aoyama, K. Ito, T. Aoki, and H. Ota, "A contactless palmprint recognition algorithm for mobile phones," in *IWAIT*, 2013, pp. 409–413.

[11] M. Aykut and M. Ekinci, "Developing a contactless palmprint authentication system by introducing a novel roi extraction method," *Image Vis. Comput.*, vol. 40, pp. 65–74, 2015.

[12] M. Franzgrote, C. Borg, B. J. T. Ries, S. Bussemaker, X. Jiang, M. Fieleser, and L. Zhang, "Palmprint verification on mobile phones using accelerated competitive code," in *ICHB*. IEEE, 2011, pp. 1–6.

[13] Q. Zhao, W. Bu, and X. Wu, "Sift-based image alignment for contactless palmprint verification," in *ICB*. IEEE, 2013, pp. 1–6.

[14] A. Kong and D. Zhang, "Competitive coding scheme for palmprint verification," in *ICPR*. IEEE, 2004, vol. 1, pp. 520–523.

[15] W. Jia, D. Huang, and D. Zhang, "Palmprint verification based on robust line orientation code," *Pattern Recognit.*, vol. 41, no. 5, pp. 1504–1513, 2008.

[16] S. Iitsuka, K. Ito, and T. Aoki, "A practical palmprint recognition algorithm using phase information," in *ICPR*. IEEE, 2008, pp. 1–4.

[17] R. J. Wang, X. Li, S. Ao, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," *arXiv:1804.06882*, 2018.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *arXiv:1801.04381*, 2018.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*. Springer, 2016, pp. 21–37.

[22] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *CVPR*. IEEE, 2017, pp. 6517–6525.

[23] J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv:1607.06450*, 2016.