Finger Region Estimation by Boundary Curve Modeling and Bezier Curve Learning

Masakazu Fujio[1], Keiichiro Nakazaki[2], Naoto Miura[2], Yosuke Kaga[1] and Kenta Takahashi[1]
[1]*Research and Development Group Hitachi, Ltd., Yokohama, Kanagawa, Japan*
[2]*Research and Development Group Hitachi, Ltd., Kokubunji, Tokyo, Japan*

Keywords: Bezier Curve, Semantic Segmentation, Shape-Aware Method, Finger Region Segmentationsed.

Abstract: This paper presents a shape-aware finger region segmentation method from hand images for user authentication. The recent development of encoder-decoder network-based deep learning technologies dramatically improved image segmentation accuracy. Although those methods predict the probability of belonging to each object pixel by pixel, it is impossible to consider whether the estimated region has a finger-like shape. We adopted a deep learning-based Bezier curve estimation method to realize shape-aware model training. We improved the accuracy with the case of warm color, complex background, and finger touching that would be difficult to estimate target regions using color-based heuristics or traditional pixel-by-pixel methods. We prepared ground truth data for each finger region (index finger, middle finger, ring finger, little finger), then trained both the conventional pixel-by-pixel estimation method and our Bezier curve estimation methods. Quantitative results showed that the proposed models outperform traditional methods (pixel-wise IOU 0.935) and practical speed.

1 INTRODUCTION

As the COVID-19 pandemic is occurring globally, the need for contactless user authentication is rising for applications such as entrance management, system login, and contactless payment at stores. In response, we have developed finger vein authentication technology that uses visible-light cameras typically built into mobile devices. This technology can perform biometric identification using finger images taken with a general-purpose camera. Given that various objects could appear in the background of finger images, eliminating background elements and accurately detecting fingers are critical technical challenges. Background removal and finger detection can be considered the finger region segmentation from an image. The accuracy of image segmentation methods has been dramatically improved with the recent development of deep learning technologies. Among them, encoder-decoder networks have successfully performed an image segmentation task (Ronneberger et al., 2015; Chen et al., 2018; Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019; Zhang et al., 2017; Ma et al., 2018; Minaee et al., 2020). U-Net (Ronneberger et al., 2015) is a standard and well-known encoder-decoder network for image segmentation. DeepLab v3 (Chen et al., 2018) introduced the atrous convolution into the encoder and PPM(Pyramid Pooling Module) to control the features' resolution and modified the decoder's structure.

In the context of finger region extraction from the scene image, those encoder-decoder-based image segmentation methods estimate the probability of belonging to each finger pixel by pixel. Then the multiple finger regions are extracted according to the estimated probability. However, the pixel-wise image segmentation approach is challenging to consider whether the estimated region has a finger-like shape. It is vulnerable to warm color backgrounds or complex backgrounds. When the fingers are thick or closed, the estimation of the finger-valley position becomes inaccurate by attaching the adjacent fingers. Finger regions are generally bell-shaped and can be represented by combinations of curves. Then compared to normal semantic segmentation, the target geometry is restricted. Suppose we can reflect the shape model in the training process more directory. In that case, we can expect to acquire more robust finger region extraction models with complex backgrounds or finger-touching cases.

This paper proposes a finger region estimation method using the deep-learning-based finger boundary curve regression. As an efficient and robust curve representation tool, we used the Bezier curve in vector

Fujio, M., Nakazaki, K., Miura, N., Kaga, Y. and Takahashi, K.
Finger Region Estimation by Boundary Curve Modeling and Bezier Curve Learning.
DOI: 10.5220/0011684400003411
In *Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2023)*, pages 371-378
ISBN: 978-989-758-626-2; ISSN: 2184-4313

graphics to draw smooth curves. We can reconstruct finger boundary curves and the finger regions by estimating Bezier control points. Developing lightweight finger region extraction models for practical finger vein authentication on edge devices is also vital. The contributions of our paper are as follows:

- We propose to model finger boundaries by Bezier curves and train its control points by using a deep learning regression for fingers region extraction.
- We defined boundary curve loss and experimentally showed the advantage of adding this loss function.
- We showed that the proposed method outperforms pixel-wise IOU (0.935) in the scenario of using edge devices such as smartphones and keeps the finger's shape despite the case with warm color, complex background and finger-touching cases.

2 RELATED WORKS

In the literature, there are several approaches and related works that claim finger/hand segmentation for cluttered/complex background images (Mohan et al., 2015)(Bapat and Kanhangad, 2017)(Ungureanu et al., 2018) (Al-Nima et al., 2017)(Priesnitz et al., 2021). S. Zhao (Zhao et al., 2018) used a fully convolutional network for hand region segmentation and fine-tuned the version of VGG16 model in ILSVRC-2014 competition to their collected hand segmentation datasets. However, they say that when the background is brown wood, results are unsatisfactory for some parts of the background or woods are identified as hands. P. Mohan (Mohan et al., 2015) uses sequential frame subtraction and post-processing for hand gesture recognition by assuming that the background should be static and the hand should not be stationary. However, flickering light and some objects are moving behind. In addition to that, the hand should not be moved while authenticating to improve accuracy. Bapat (Bapat and Kanhangad, 2017) combines skin color detection and shape filtering for hand segmentation. They define skin color values for HSV and RGB and then remove false positives using shape characteristics. They use the change of the number of skin pixels per row to judge whether that area is a hand region. Their method depends on whether the contour of the segment by the skin color detection is sufficiently correct, which may not be justified skin-colored background or illumination change in hand area such as a shade. Ungureanu (Ungureanu et al., 2018) used SegNet and original encoder-decoder dcnn for hand segmentation with complex backgrounds. They used standard palmprint image databases CASIA and HKPU with a monotone color background for the training. Then they augmented the image dataset by replacing the simple background with complex images with various texture properties. The evaluation of (Ungureanu et al., 2018) needs to be revised because they used only synthetic data, which does not include actual complex data. Furthermore, the evaluated models' size or inference speed is not apparent. Al-Nima et al. (Al-Nima et al., 2017) use IIT Delhi, PolyU3D2D, and spectral 460 from the CASIA Multi-Spectral Palmprint. That DB has a monotone color background. Priesnitz(Priesnitz et al., 2021) uses DeepLabV3+ for hand segmentation. However, their parameter sizes and FLOPs is not practical for edge devices such as smartphone without GPUs. Practical inference speed is one of our concerns in this paper.

Besides, there are researches on lightweight backbone networks, such as MobileNet (Howard et al., 2017), ShuffleNet (Zhang et al., 2017), MobileNet-v2 (Sandler et al., 2018), ShuffleNetv2 (Ma et al., 2018) and MobileNet-v3 (Howard et al., 2019). These models decompose normal convolution into depth-wise convolution and point-wise convolution to reduce parameters and FLOPs. For practical finger vein authentication that uses visible-light cameras on edge devices, it is crucial to develop lightweight and accurate finger region extraction from the camera images.

3 PROPOSED METHOD

3.1 Finger Region Segmentation by Bezier Curve Learning

Recently, the image segmentation approach by deep learning models has exhibited many successes. However, pixel-wise image segmentation can not directly incorporate the segmented regions' shape. In the literature, some studies estimate the parameters of the spline curve by regression-based deep learning [14, 15]. Finger regions are generally bell-shaped and can be represented by combinations of simple curves. Thus, it is straightforward to detect spline curves representing fingers' boundaries. As an efficient and robust curve representation tool, we used the Bezier curve in vector graphics to draw smooth curves.

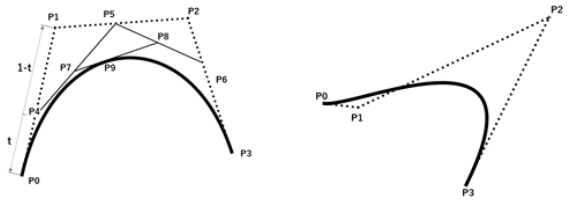

Figure 1: Examples of cubic Bezier curves (with four control points). The diagram on the left shows a drawing process of a Bezier curve from its parameters (control points, $P_0 \sim P_3$).

3.2 Finger Region Representation by Bezier Curves

A Bezier curve is based on n control points and is defined by the following equation:

$$B(t) = \sum_{i=0}^{n} b_{i,n}(t) P_i, \qquad 0 \le t \le 1 \qquad (1)$$

where the polynomials

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \qquad i = 0, \ldots, n \qquad (2)$$

are known as Bernstein basis polynomials of degree n. $\binom{n}{i}$ are the binomial coefficients. $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ The points P_i are called control points for the Bezier curve. In this study, each point P_i is an XY coordinate.

By changing the value of t from 0 to 1, we can get the sequence of the coordinates of the Bezier curve. In other words, once you get the parameter value (XY coordinates) of $P_i \sim P_n$, you can draw the Bezier curve. The left diagram in Figure 1 illustrates this process. The control points $P_0 \sim P_3$ are the parameters of the Bezier Curve. Additional points, $P_4 \sim P_9$, are calculated iteratively from the starting control points $P_0 \sim P_3$. At the first iteration, we calculate $P_4 \sim P_6$. P_4 is the $t : 1-t$ division point of a straight line P_0P_1. P_5 is the $t : 1-t$ division point of a straight line P_1P_2. P_6 is the $t : 1-t$ division point of a straight line P_2P_3. At the second iteration, we calculate $P_7 \sim P_8$. P_7 is the $t : 1-t$ division point of a straight line between P_4P_5. P_8 is the $t : 1-t$ division point of a straight line between P_5P_6. At the last iteration, we calculate P_9. P_9 is the $t : 1-t$ division point of a straight line between P_7P_8. This recursive calculation is repeated n times. The n is the dimension of the Bezier Curve. Examples of Figure 1 are n=3 (cubic Bezier curve). $B(t)$ in (1) corresponds to the XY coordinate of the calculated point P_9. Because the additional points $P_4 \sim P_9$ are expressed by the control points $P_0 \sim P_3$ formulas, we can calculate $B(t)$ as equation (1).

We investigated for the number of control points required for representing finger boundaries by Bezier curves. As a result of trial and error, we confirmed that the finger shape could be adequately reproduced by dividing each finger into two regions, left and right, and using six control points for each divided curve. It was difficult to estimate control points from a whole finger boundary curve by least square estimation of P_i (equation (3); Large discrepancy between an estimated curve and original curve). The contour of a finger is not a simple bell shape but often has a complicated curve, which requires many control points. By estimating XY coordinates of the control points of each piece of curves (a total of twenty-four control points), we can reproduce the original curves, then extract finger regions.

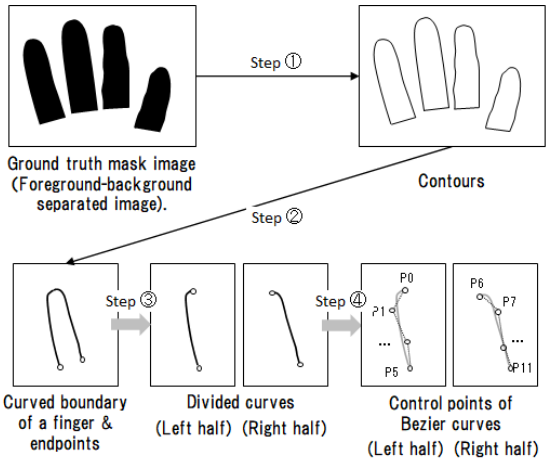

Figure 2: Overview of the ground truth data conversion process. We convert binary mask-represented finger region ground truth data into the Bezier curve control points.

We convert finger region ground truth data (e.g., binary mask represented) into the Bezier curve control points. Figure 2 shows the conversion process step by step.

In step (1), we extract four contours from binary images using (Suzuki and be, 1985). The resulting contours are represented by sequence of coordinates (polygons). Each polygon represents one of four fingers. In step (2), we search finger valleys points based on the coordinates of each polygon. In step (3), we divide each piece of the polygon into two parts (left half and right half) at the point with the smallest y-coordinate of the polygon. Each piece consists of a curve from a finger valley to a fingertip. Finally, in step (4), we obtain the optimal parameters for Bezier curves (XY coordinate P_i in equation (1)) from the sampling points (coordinates in the divided polygon). Given the m sampling points $\{p_i\}_{i=1}^{m}$ from the curved boundary, where p_i represents the ith sampling point, we can use the standard least square method to achieve this, as shown in equation (3) (Liu et al.,

2020). $t_1 \sim t_m$ are the m sample values of t in equation (1).

$$\begin{bmatrix} b_{0,5}(t_0) & \dots & b_{5,5}(t_0) \\ b_{0,5}(t_1) & \dots & b_{5,5}(t_1) \\ \vdots & \ddots & \vdots \\ b_{0,5}(t_m) & \dots & b_{5,5}(t_m) \end{bmatrix} \begin{bmatrix} P_{x_0} & P_{y_0} \\ P_{x_1} & P_{y_1} \\ \vdots & \vdots \\ P_{x_5} & P_{y_5} \end{bmatrix} = \begin{bmatrix} p_{x_0} & p_{y_0} \\ p_{x_1} & p_{y_1} \\ \vdots & \vdots \\ p_{x_m} & p_{y_m} \end{bmatrix} \tag{3}$$

These four steps can convert mask annotation to a parameterized Bezier curve. In this study, we directly use the endpoint of the divided polygons in the above step (4) as the first (P_0 and P_6) and the last (P_5 and P_{11}) control points, respectively. Then, we train the model to estimate those control points for four fingers (a total of twelve control points).

3.3 Bezier Curve Parameter Estimation by Regression Based Deep Learning

Once the trained model estimates the control points for eight curves of four fingers (3.1), we can reconstruct the regions of four fingers. In this subsection, we briefly explain the model used to regress the XY coordinates of the control points.

3.3.1 Model Structure

Figure 5 shows the overall model structure used for the Bezier control points regression. This study used FCOS (Fully Convolutional One-Stage Object Detection) Net (Tian et al., 2019) as a regression estimation model for control points. FCOS Net constructs multi-scale semantic feature maps through Feature Pyramid Network (FPN) (Lin et al., 2016). On top of each scale feature map, the attached Head layer has a classification branch, regression branch, and center-ness branch. The classification branch predicts the class of target objects. The bounding box branch predicts base pixel coordinates and four relative distances (left, top, right, and bottom from the base pixel coordinates). The center-ness branch predicts a pixel's deviation to the center of its corresponding bounding box.

To achieve sufficient inference speed in the edge devices such as smartphones, we used Mobilenet-v2 as the backbone network of FCOS Net. In addition, the channel size of the backbone network and feature maps are half the original (Tian et al., 2019). It also reduced the number of feature map layers from $F3 \sim F7$ into $F3$ and $F5$. The bounding box regression mechanism of FCOS Net is used to estimate the control point coordinates of the Bezier curve (Liu et al., 2020).

3.3.2 Bezier Control Points Regression

In our study, we define four class labels, index finger, middle finger, ring finger, and little finger. We define the ground-truth bounding box from the endpoints of the divided polygons (P_0, P_5, P_6, P_{11}) in step (4) in Figure 2. Note that our definition of a bounding box is not necessarily equal to the bounding box of the finger region. Figure 3 illustrates the resulting ground-truth bounding box for an index finger. The regression branch in our model predicts the base coordinate and four relative distances, $\Delta_l = base_x - left, \Delta_t = base_y - top, \Delta_r = base_x - right, \Delta_b = base_y - bottom$. In addition, the Regression branch also predicts twenty-four (12 points for each finger) relative distances from the same base coordinate for each control point $P_0 \sim P_{11}$, $\Delta_x = P_{x_i} - base_x, \Delta_y = P_{y_i} - base_y$. Figure 4 illustrates the resulting Bezier control points for an index finger.

The detected bounding boxes are scored by multiplying the predicted center-ness with the corresponding classification score to filter out bounding boxes and Bezier control points far from the center. This filter out is done by the bounding box non-maximum suppression (NMS) process.

3.3.3 Loss Function

In a text spotting method ABCNet (which is based on FCOS Net) (Liu et al., 2020), they define three types of loss functions, classification loss (L_{cls}), bounding boxes regression loss (L_{loc}), and Bezier control points regression loss (L_{cpt}) for text spotting. In this paper, we define boundary curve loss by sampling contour points from the estimated/target Bezier control points by using equation (1).

$$L_{bcl} = \frac{1}{m} \sum_{i=0}^{m} |B_{estimate}(t_i) - B_{target}(t_i)| \qquad 0 \le t_i \le 1 \tag{4}$$

We take the value of t_i equally spaced sampling with values from 0 to 1. Our model minimizes the following total loss function (L_{total}).

$$L_{total} = L_{cls} + L_{loc} + L_{cpt} + \lambda_1 L_{bcl} \tag{5}$$

λ_1 is the hyperparameter for the boundary curve loss L_{bcl}. We set this value to 0.1.

4 EVALUATION

We performed deep learning model training for finger region estimation using the models proposed in the previous section and evaluated finger region estimation accuracy and inference speeds. First of all,

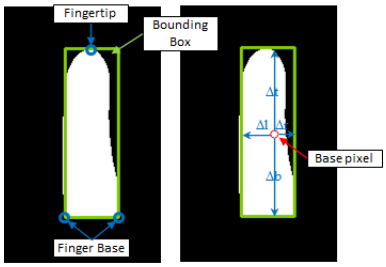

Figure 3: (Left) We define the ground-truth bounding box from the endpoints of the divided polygons in step (4) in Figure 2 (P_0, P_5, P_6, P_{11}). P_0, P_6 corresponds to the fingertips (smallest y-coordinate of the polygon), and P_5, P_{11} corresponds to the finger bases; (right) Bounding box regression examples. The regression branch in our model predicts the base coordinate and four relative distances, $\Delta_l = base_x - left, \Delta_t = base_y - top, \Delta_r = base_x - right, \Delta_b = base_y - bottom$.

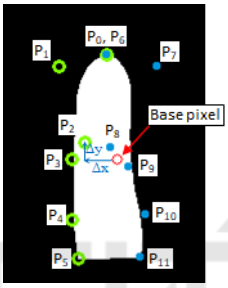

Figure 4: The regression branch also predicts twenty-four (12 points for each finger) relative distances for each control point $P_0 \sim P_{11}$ ($\Delta_{x_i} = P_{x_i} - base_x, \Delta_{y_i} = P_{y_i} - base_y$); (green circle) Control points for the left split curve in Figure 2; (blue circle) Control points for the right split curve in Figure 2.

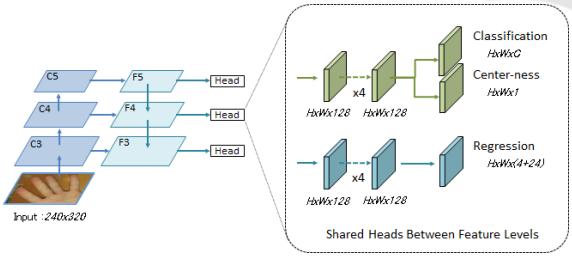

Figure 5: Examples of FCOS net. C3, C4, and C5 denote the feature maps of the backbone network, and F3, F4, and F5 are the feature levels used for the final prediction. The original (Tian et al., 2019) used five feature maps ($F3 \sim F7$).

we explain the data preparation procedure for model training.

DB: Two public hand image databases (PolyU-IITD ver.3 (Kumar, 2019), NTU-CP-v1 (Matkowski et al., 2020)) and one in-house databases. We collected smartphone rear camera captured hand images mainly in work-from-home environments (total of 418 hands (left and right), 16,603 images) (We call collected in-house database "ARC." Four thousand images were held-out for validation). The images are resized into 240x320. That dataset has no ground-truth hand masks. Then we need to prepare ground-truth masks.

Ground-Truth Data: Binary mask image for each finger (index finger, middle finger, ring finger, little finger). The target finger region is defined from the fingertip to the finger valleys. The conventional method extracted the ground truth data (color and shape-based heuristics).

Data Cleansing: The image's ground-truth labeling of correct finger regions was done using a rule-based finger region segmentation algorithm based on color and shape. However, since some images fail to be authenticated if the extracted finger region's size is less than a certain threshold, they are excluded from the labeled images. In addition, we adopt the following criteria for the ground truth data cleansing: (1) Successful extraction of 4 fingers. (2) The height of the finger valleys along with the finger's long axis is less than one-fifth of the height of the middle finger. (3) Finger area is more than half of the bounding box size. (4) Aspect ratio threshold (exclude objects with a large pitch or objects that may be in the middle of the finger.)

Data Augmentation: To expand the variation of the training data, we conducted the following image data augmentation. (1) RandomColorJitter, (2) RandomPerspective, (3) RandomCrop, (4) Image synthesis from hand images and background images. In step (4), we used Place365 [1] database for the various background scene.

The model can predict various background environments that the conventional heuristics method can not predict correctory by adopting data augmentation.

4.1 Region Estimation Accuracy

We used the dataset (Table 1) for model training and evaluation. From the dataset, about 4,000 images are held out for evaluation. We used Intersection Over Union (IOU) for the pixel-wise classification accuracy metrics for region estimation accuracies. The evaluation of authentication accuracies is out of the scope of this study. Still, improving the region estimation improves the finger biometrics authentication accuracies.

Table 3 shows the prediction accuracy (mIoU) for each model. The first column shows the model name

[1] http://places2.csail.mit.edu/download.html

Table 1: Hand image dataset used for this study.

DB Name	DB Type	Hand IDs	Images
PolyU-IITD ver.3 (Kumar, 2019)	Public	1,220	12,200
NTU-CP-v1 (Matkowski et al., 2020)	Public	655	2,478
ARC	In house	936	25,603

Table 2: Hand image dataset used for model robustness comparison. ICF: Indoor Closed Finger, IWB: Indoor Warm Background.

DB Name	DB Type	Images
ICF	In house	1,841
IWB	In house	110

(SHFL:Shufflenet, SHFLV2:ShufflenetV2, MNV1: MobileNet, MNV2: MobilenetV2, MNV3: MobilenetV3). Each model name represents the backbone network model except for the FBCR (ours). We evaluated two cases of FBCR with and without loss function L_{bcl} in equation (5). We adopted lightweighted models[2] that can practically be used in edge devices such as smartphones. The second column shows the accuracy for each finger. The third column shows the average accuracy for all fingers (mIoU). We used inference speed in a GPU-less environment to select models for comparison. One criterion is that the inference time on a smartphone is less than 700 msec (see Table 4). As can be observed, the proposed model FBCR w/o L_{bcl} outperforms other pixel-by-pixel estimation methods, such as the MobilenetV3-based segmentation model (MNV3) (Howard et al., 2019). Furthermore, FBCR w/ L_{bcl} improves the accuracy by about 0.3 %.

Table 4 shows the result of processing time on the CPU for each model. We used a smartphone ($QualcommSnapdragon^{TM}835$) for the performance evaluation. The inference speed of the FBCR model is a bit slower than the MNV3 model (Table 4). We can improve the performance by restricting the adaptable object resolutions of the Feature Pyramid Network(FPN) in FCOS.

4.2 Robustness in Background Variations

Table 5 shows the prediction accuracy (mIoU) for warm color backgrounds (Table 2:IWB). As can be observed, the proposed model FBCR outperforms MNV2 (the second-best model in Table 3).

Figure 6 shows the extraction results with different models for complex background scenes. We compared three models, Heuristics, MNV2, and ours. The first column is the case for almost all warm color backgrounds. In this case, the heuristics method fails to predict all finger regions. MNV2 predicts all finger regions, but the shape of the first, third, and fourth fingers is irregular.

On the other hand, FBCR (ours) model predicts all fingers except the fourth finger. Instead of outputting an unclear finger region, output nothing. The second column is the results for the warm color background (a red bar is behind the finger). In this case, the heuristics method fails to predict except for the little finger. MNV2 predicts all finger regions, but the index finger region includes part of the red bar region.

On the other hand, the FBCR model predicts all finger regions correctory. The third column is a complex background beside the balcony on a sunny day, with shadows on the hand. The results of the heuristics method appear to be affected by a difference in luminance in finger regions. In this case, both MNV2 and FBCR predict all fingers correctly. The last column is another case for warm color background (wood-grained table). In this case, both MNV2 and FBCR predict all finger regions correctory.

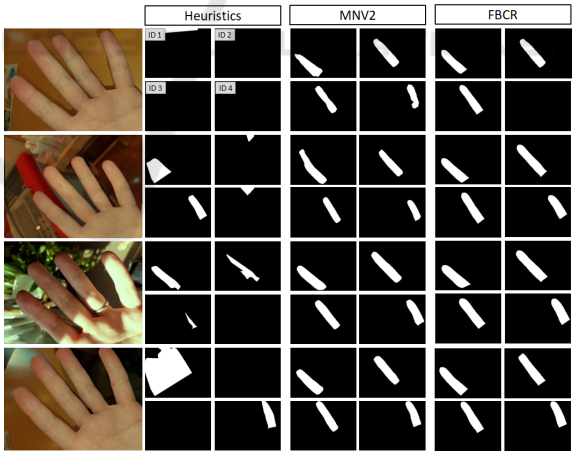

Figure 6: Finger region extraction example for each model.

4.3 Robustness for Finger Touching

Table 6 shows the prediction accuracy (mIoU) for all finger-touching images(Table 2:ICF). We did not include finger-touching hand images in the training datasets in this study. Then the prediction accuracy of both FBCR and MNV2 degrades. As can be observed, the proposed model FBCR remarkably outper-

[2]Lightweight Model for Real-Time Semantic Segmentation: https://github.com/Tramac/Lightweight-Segmentation

Table 3: Prediction accuracy for each model.

Model Name	mIoU for each Finger ID				mIoU for all fingers
	INDEX	MIDDLE	RING	LITTLE	
SHFL	0.907	0.930	0.924	0.921	0.922
SHFLV2	0.906	0.924	0.929	0.928	0.922
MNV1	0.907	0.927	0.926	0.923	0.921
MNV2	0.910	0.934	0.932	0.927	0.926
MNV3	0.908	0.929	0.922	0.916	0.919
FBCR(ours) w/o L_{bcl}	0.912	0.937	0.937	0.934	0.932
FBCR(ours) w/ L_{bcl}	**0.923**	**0.940**	**0.941**	**0.935**	**0.935**

Table 5: Prediction accuracy for warm color backgrounds.

Model Name	mIoU for each Finger ID				mIoU for all fingers
	INDEX	MIDDLE	RING	LITTLE	
MNV2	0.722	0.776	0.731	0.679	0.727
FBCR(ours)	**0.774**	**0.781**	**0.787**	**0.781**	**0.781**

Table 6: Prediction accuracy for finger touching images for each model.

Model Name	mIoU for each Finger ID				mIoU for all fingers
	INDEX	MIDDLE	RING	LITTLE	
MNV2	0.448	0.320	0.263	0.127	0.289
FBCR(ours)	**0.547**	**0.624**	**0.525**	**0.548**	**0.560**

Table 4: Processing time for each model. (Qualcomm Snapdragon (TM) 835).

Model Name	Inference Speed (msec)
SHFL	580
SHFLV2	565
MNV1	496
MNV2	673
MNV3	163
FBCR(ours)	230

forms MNV2 (the second-best model in Table 3) for all fingers.

Figure 7 shows the extraction results with different models for finger posture differences. Since one of our purpose of finger region extraction is user authentication, the posture is restricted to the range that the finger surface can be seen from the camera. Therefore a small amount of rotation or pitching is the target of this comparison.

The top column of Figure 7 shows the extraction results for finger touching in the middle. The tails of the middle finger region and the ring finger regions on MNV2 are estimated short because of the fingers' touch in the middle. On the other hand, finger region and length are correct on the FBCR model. The second column shows the extraction results for four-finger touching. The shape of the third and fourth fingers is crumbling, and the boundary of the middle finger is corrupted. The third column of Figure 7 shows a hand rolling case (rotation in the y-axis). In this case, the index finger overlaps the region of the middle finger area a little. The index finger region on the MNV2 model is crumbling, and other finger regions are not correct either.

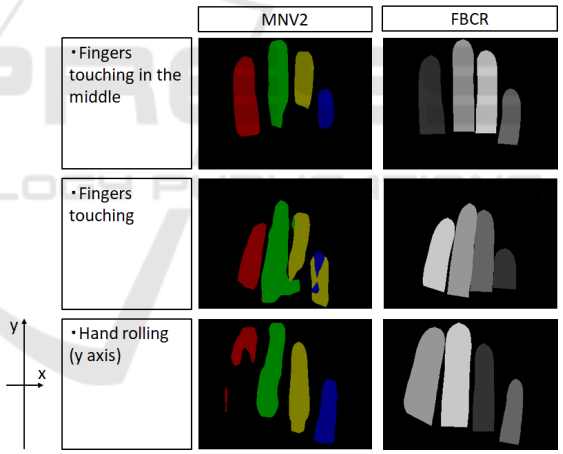

Figure 7: (Left) estimation results with MNV2; (right) estimation results with the model FBCR (ours).

The drawback of the FBCR is that when the model fails to detect a finger, there is no output of region estimation for that finger.

5 CONCLUSIONS

This paper proposed a shape-aware finger region segmentation method for user authentication from hand

images. We adopted a deep-learning-based Bezier curve estimation model to realize shape-aware model training. Finger regions generally have a bell shape and can be represented by parametric spline curves such as Bezier curve. Our model estimates the set of control points and then reconstructs the curved boundary of fingers. We prepared ground truth data for each finger (index finger, middle finger, ring finger, little finger). Then trained, a conventional encoder-decoder-based deep learning network and proposed Bezier curve estimation model. We showed that the proposed method outperforms other models by pixel-wise IOU (0.935) in using edge devices such as smartphones and keeps the finger's shape despite the case with warm color and complex background. We plan to improve the inference speed for the application to edge devices in future work.

REFERENCES

Al-Nima, R. R. O., Dlay, S. S., Woo, W. L., and Chambers, J. A. (2017). Efficient finger segmentation robust to hand alignment in imaging with application to human verification. In *2017 5th International Workshop on Biometrics and Forensics (IWBF)*, pages 1–6.

Bapat, A. and Kanhangad, V. (2017). Segmentation of hand from cluttered backgrounds for hand geometry biometrics. In *2017 IEEE Region 10 Symposium (TENSYMP)*, pages 1–4.

Chen, L., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611.

Howard, A., Sandler, M., Chu, G., Chen, L., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H. (2019). Searching for mobilenetv3. *CoRR*, abs/1905.02244.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

Kumar, A. (2019). Toward more accurate matching of contactless palmprint images under less constrained environments. *IEEE Transactions on Information Forensics and Security*, 14(1):34–47.

Lin, T.-Y., Doll?r, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2016). Feature pyramid networks for object detection.

Liu, Y., Chen, H., Shen, C., He, T., Jin, L., and Wang, L. (2020). Abcnet: Real-time scene text spotting with adaptive bezier-curve network. *CVPR*, abs/2002.10200.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Matkowski, W. M., Chai, T., and Kong, A. W. K. (2020). Palmprint recognition in uncontrolled and uncooperative environment. *IEEE Transactions on Information Forensics and Security*, 15:1601–1615.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., and Terzopoulos, D. (2020). Image segmentation using deep learning: A survey. *CoRR*, abs/2001.05566.

Mohan, P., Srivastava, S., Tiwari, G., and Kala, R. (2015). Background and skin colour independent hand region extraction and static gesture recognition. In *2015 Eighth International Conference on Contemporary Computing (IC3)*, pages 144–149.

Priesnitz, J., Rathgeb, C., Buchmann, N., and Busch, C. (2021). Deep learning-based semantic segmentation for touchless fingerprint recognition. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10-15, 2021, Proceedings, Part VIII*, page 154?168, Berlin, Heidelberg. Springer-Verlag.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381.

Suzuki, S. and be, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46.

Tian, Z., Shen, C., Chen, H., and He, T. (2019). FCOS: fully convolutional one-stage object detection. *CoRR*, abs/1904.01355.

Ungureanu, A.-S., Bazrafkan, S., and Corcoran, P. (2018). Deep learning for hand segmentation in complex backgrounds. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–2.

Zhang, X., Zhou, X., Lin, M., and Sun, J. (2017). Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083.

Zhao, S., Yang, W., and Wang, Y. (2018). A new hand segmentation method based on fully convolutional network. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 5966–5970.