IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 December 2023, accepted 18 December 2023, date of publication 21 December 2023,
date of current version 27 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3345644

RESEARCH ARTICLE

Unveiling the Power of Simplicity: Two Remarkably Effective Methods for Fingerprint Segmentation

RAFFAELE CAPPELLI
Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy
e-mail: raffaele.cappelli@unibo.it

ABSTRACT Accurate fingerprint segmentation is crucial for reliable fingerprint recognition systems. This paper presents two novel segmentation methods, GMFS and SUFS, inspired by the KISS (Keep It Simple and Straightforward) principle. Both methods, evaluated on a public benchmark and compared to eighteen state-of-the-art approaches, excel in terms of accuracy, while maintaining simplicity and computational efficiency. GMFS utilizes a single handcrafted feature for straightforward yet effective fingerprint segmentation, achieving superior performance compared to previously reported traditional methods. SUFS employs a simplified U-net architecture for end-to-end segmentation, demonstrating remarkable performance: it achieves an average classification error rate of 1.51% across the entire benchmark, with an improvement of over 40% compared to the previously best-performing method. Furthermore, despite being trained on a relatively small dataset, it exhibits significant generalization capabilities, effectively segmenting fingerprints from very different acquisition technologies without requiring fine-tuning. An open-source Python implementation of both methods is available, fostering further research and development in the field of fingerprint recognition.

INDEX TERMS Fingerprints, fingerprint segmentation, KISS principle, magnitude of the gradient, U-net.

I. INTRODUCTION

Fingerprint recognition has emerged as a ubiquitous biometric technology, revolutionizing personal identification due to its reliability and accuracy [1], [2]. Its versatility has made it a cornerstone of various applications, including criminal investigations, border control systems, and mobile device authentication. A critical step in fingerprint recognition is segmentation, the process of isolating the fingerprint pattern from the background (Fig. 1). Segmentation is essential as it eliminates noise and irrelevant information, paving the way for accurate fingerprint recognition algorithms.

The field of fingerprint segmentation has witnessed a proliferation of techniques, transitioning from traditional methods based on handcrafted features in the spatial or frequency domain, to more sophisticated approaches utilizing classifiers, genetic algorithms, clustering, and, more recently, deep learning. While these advancements have undoubtedly improved segmentation performance, they have also introduced a degree of complexity that can hinder implementation and comprehension of the underlying mechanisms driving performance gains. This study adopts the KISS (Keep It Simple and Straightforward[1]) principle [3], [4], seeking to develop novel segmentation methods that achieve state-of-the-art performance while maintaining simplicity.

The primary contributions of this paper are as follows:

1. A novel segmentation method based on simple and efficient image processing steps that achieves state-of-the-art performance.
2. A novel segmentation method based on a simplified U-net architecture that surpasses all previous methods evaluated on a public benchmark and is able to

[1]While other interpretations of the KISS acronym are more common, this paper employs a less colloquial but more academically appropriate definition in the interest of maintaining scientific precision and avoiding potential misinterpretations.

The associate editor coordinating the review of this manuscript and approving it for publication was Zahid Akhtar.

deal with fingerprints acquired through very different technologies without requiring fine-tuning.
3. An open-source implementation of both methods to facilitate further research and development in this domain.

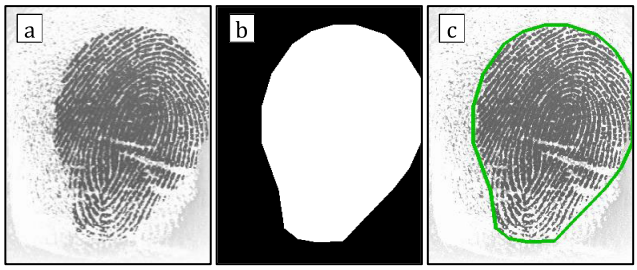

FIGURE 1. An example of fingerprint segmentation. (a) a fingerprint, (b) segmentation mask, (c) segmentation mask contour overlaid on the fingerprint.

The rest of this paper is organized as follows. Section II reviews the main fingerprint segmentation methods proposed in the literature. Section III describes the two novel fingerprint segmentation methods. Section IV reports experiments aimed at evaluating the performance of the proposed methods and comparing them to the state-of-the-art on a public benchmark. Finally, section V draws some concluding remarks.

II. RELATED WORKS

During the last four decades, more than 100 methods for fingerprint segmentation have been published [1], [5]. The earliest methods typically split the image into blocks (e.g., 16×16 pixels), extract some features from each to classify it as foreground or background, and finally perform a few postprocessing operations. Mehtre et al. [6] look for the presence of peaks in histograms of local ridge orientations, and, in [7], consider also the gray-level variance. Ratha et al. [8] use the gray-level variance as well but computing it along the direction orthogonal to the ridge orientation. Bazen and Gerz [9] compute the local mean and variance of gray-level intensities, and the coherence of local gradients; a simple perceptron is trained as a foreground/background classifier from these features. A similar technique is proposed in [10]. Shen et al. [11] and Alonso-Fernandez et al. [12] propose methods based on Gabor filter bank responses, while Wu et al. [13] choose Harris corner response as main feature and Gabor filter responses for postprocessing. Zhu et al. [14] start from a local orientation estimation as the main feature, then a shallow neural network helps to detect wrongly estimated orientations and improves segmentation. Wang et al. [15] propose a method based on Gaussian-Hermite moments. Other methods rely on fuzzy C-means clustering [16], [17], [18].

Given that fingerprints are characterized by an oriented pattern with frequencies only in a specific band of the Fourier spectrum, some authors propose segmentation methods that work in the frequency domain. In [19], Chikkeru et al. perform a local Fourier analysis with the goal of enhancing the fingerprint pattern, obtaining a fingerprint segmentation mask together with an estimation of local ridge orientation and frequency. Hu et al. [20] apply a Log-Gabor filter in the frequency domain and combine it with orientation reliability information. Marques and Thome [21] extract feature vectors based on the Fourier spectrum and the directional consistency of each 32×32 block, to train a shallow neural network. Thai et al. [5] design a segmentation approach based on the directional Hilbert transform of Butterworth bandpass filters. Thai and Gottschlich [22] propose a segmentation method by global three-part decomposition, which decomposes a fingerprint image into cartoon, texture, and noise parts: the foreground mask is obtained from the non-zero coefficients in the texture image using morphological processing.

More recent methods are mostly based on deep learning. Dai et al. [23] first apply the total variation model to decompose the fingerprint image into cartoon and texture components; then the texture component is divided into overlapping blocks, each of which is classified as foreground or background by a convolutional neural network; the final segmentation mask is obtained after a morphology-based postprocessing. Another block-based deep learning method is described in [24], where Serafim et al. experiment two well-known convolutional neural network architectures for block classification: LeNet [25] and AlexNet [26]; as in the previous method, some postprocessing steps produce the final segmentation mask. An end-to-end fingerprint segmentation method, trained on full sized images, is described in [27], where Joshi et al. introduce a recurrent U-Net with dropout, called DRUnet, and compare it to four existing network architectures: Conditional Generative Adversarial Network [28], U-Net [29], Convolution neural network with criss-cross attention module [30], and Recurrent U-Net [31].

This section has not considered latent fingerprint segmentation methods, since this paper focuses on plain fingerprints [1]. Segmentation of latent fingerprints requires specifically-designed approaches that are outside the aims of this study: interested readers may refer to [32], [33], and [34] and the references therein.

III. PROPOSED METHODS

Given a grayscale image $\mathbf{F}$ containing a fingerprint image, a segmentation method must classify each pixel in $\mathbf{F}$ as *foreground* or *background*: the result is a binary image $\mathbf{S}$ with the same size of $\mathbf{F}$ (Fig. 1).

The following sections describe the two fingerprint segmentation methods proposed in this paper. Both methods expect fingerprint images with a resolution of 500 dpi, which is the typical resolution of most fingerprint scanners [1].

A. GMFS

GMFS (Gradient-Magnitude Fingerprint Segmentation) is the result of a research effort aimed at designing a fingerprint segmentation method based on traditional image processing techniques and inspired by the KISS principle. In particular,

during its development, the author's goal was to obtain a method that:

- uses a small number of simple features (ideally only one),
- consists in a short sequence of well-known and efficient image processing steps, and
- requires a small number of parameters to be configured.

Fig. 2 shows a functional schema of GMFS and Fig. 3 an example with all the intermediate processing steps. Feature extraction simply consists of gradient magnitude estimation for each pixel, and is performed as follows. Let $\frac{\partial \mathbf{F}}{\partial x}$ and $\frac{\partial \mathbf{F}}{\partial y}$ be the two images which, at each pixel, contain the horizontal and vertical derivative approximations of the input fingerprint $\mathbf{F}$. $\frac{\partial \mathbf{F}}{\partial x}$ and $\frac{\partial \mathbf{F}}{\partial y}$ can be computed by convolution with the Sobel filters $\mathbf{S}_x$ and $\mathbf{S}_y$ [35]: $\frac{\partial \mathbf{F}}{\partial x} = \mathbf{F} * \mathbf{S}_x$, $\frac{\partial \mathbf{F}}{\partial y} = \mathbf{F} * \mathbf{S}_y$. The magnitude of the gradient at each pixel is $\mathbf{M} = \sqrt{\left(\frac{\partial \mathbf{F}}{\partial x}\right)^2 + \left(\frac{\partial \mathbf{F}}{\partial y}\right)^2}$. The gradient magnitude is typically high at the transition points between ridges and valleys (see Fig. 3.b).

Starting from this feature, a threshold t is computed as $t = \mathit{percentile}\,(\mathbf{M}, 95) \cdot \tau$, where $\mathit{percentile}\,(\mathbf{M}, 95)$ is the 95^{th} percentile of matrix $\mathbf{M}$, and τ is a parameter of the method (see section IV-B). Using the 95^{th} percentile, instead of simply choosing the maximum value in $\mathbf{M}$, helps to mitigate the impact of outliers, which are often present due to noise in the image.

The gradient magnitude is then averaged by convolving $\mathbf{M}$ with a Gaussian filter $\mathbf{G}_\sigma$ with size $g_s \times g_s$: $\bar{\mathbf{M}} = \mathbf{M} * \mathbf{G}_\sigma$. Filter $\mathbf{G}_\sigma$ is obtained by discretizing the 2D Gaussian function $G_{2D}\,(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ over a $g_s \times g_s$ grid and normalized by dividing each element by the sum of all the filter values. σ is a parameter of the method (see section IV-B), while the filter size is set to $g_s = \lceil 3 \cdot \sigma \rceil \cdot 2 + 1$, to contain most of the Gaussian values according to the three-sigma rule. This smoothing step is very important: it reduces the effect of noise and "fills" most of the inner regions of ridges and valleys where the gradient magnitude is low (see Fig. 3.c).

The initial segmentation mask $\mathbf{S}_t$ is obtained by a simple thresholding operation on the average gradient magnitude:

$$\mathbf{S}_t = \left[S_{i,j}\right], \text{ with } S_{i,j} = \begin{cases} \mathit{foreground} & \mathit{if}\ \bar{M}_{i,j} > t \\ \mathit{background} & \mathit{otherwise} \end{cases}$$

Note that threshold t is computed from the non-averaged gradient magnitude $\mathbf{M}$, while thresholding is performed on the averaged gradient magnitude $\bar{\mathbf{M}}$.

The final segmentation mask $\mathbf{S}$ is obtained from $\mathbf{S}_t$ after the following postprocessing steps:

1. Morphological closing [35] (dilation followed by erosion) with a simple 3×3 disc-shaped structuring element (each erosion and dilation step is repeated n_c times, where n_c is a parameter of the method).
2. If the foreground contains more than one connected component, only the largest one is considered; the connected component labeling algorithm [35] can be used to perform this step efficiently.
3. Filling any holes, provided they are not adjacent to an image border; this step can be efficiently carried out by applying the connected components labeling algorithm to the background.
4. Morphological opening [35] (erosion followed by dilation) with the same structuring element used at step 1 (each dilation and erosion step is repeated n_o times, where n_o is a parameter of the method).
5. If the previous step creates more than one connected component, step 2 is executed again.

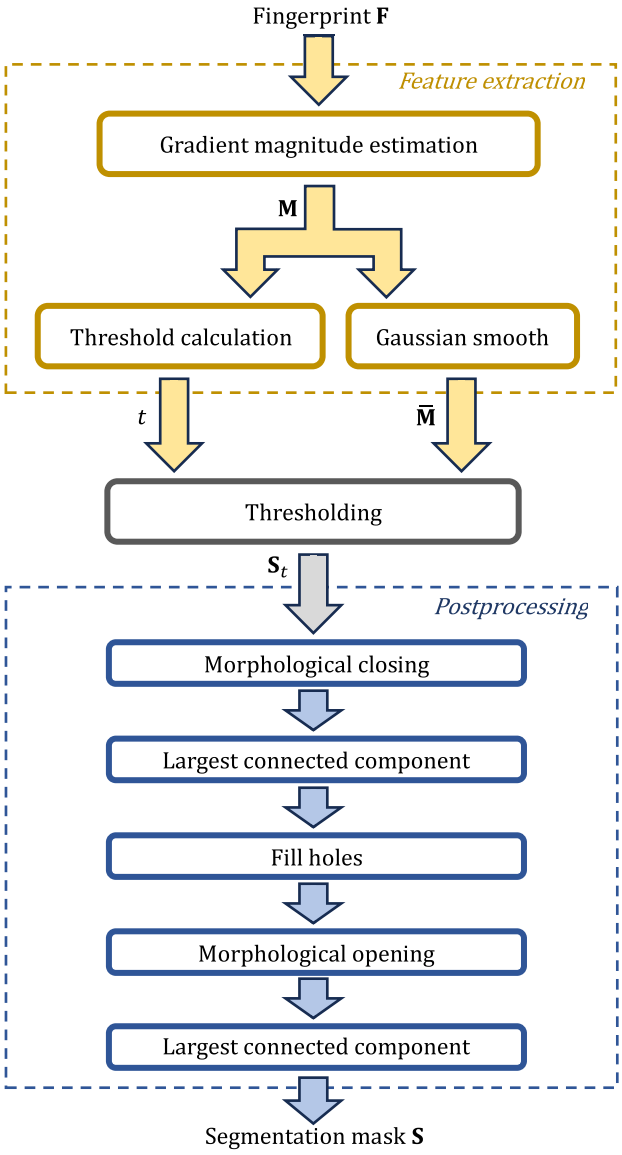

FIGURE 2. A visual summary of the proposed GMFS method.

Fig. 3.e-h illustrate the effects of the first four postprocessing steps: the first step partially fills the holes caused by noise (e), the second one removes the noise artifact at the bottom-left

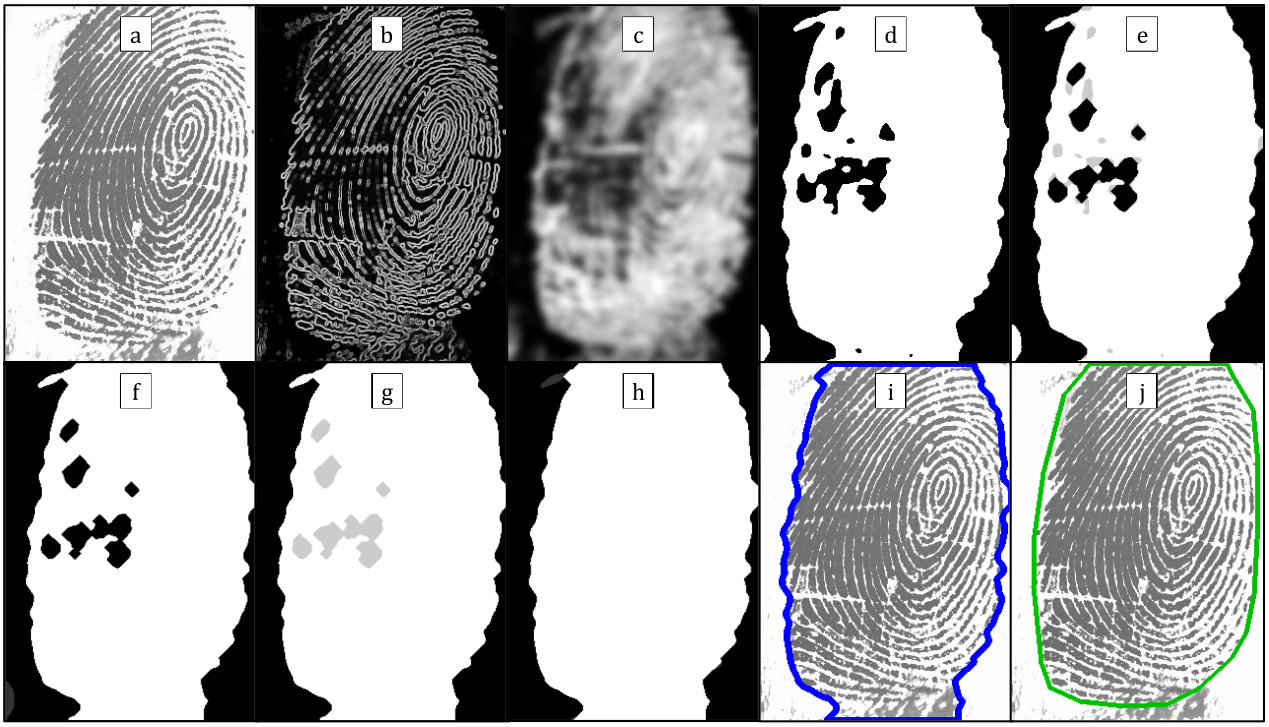

FIGURE 3. An example of fingerprint segmentation using GMFS: (a) Fingerprint image F, (b) Gradient magnitude M, (c) Averaged gradient magnitude $\bar{\mathbf{M}}$, (d) Initial segmentation mask $\mathbf{S_t}$, (e)-(h) Results of the first four postprocessing steps with the previous mask superimposed in semitransparency to highlight changes, (i) Final segmentation mask, and (j) Ground truth.

corner of the image (f), the third one fills the remaining holes (g), and the fourth postprocessing step removes the protrusion at the top-left of the fingerprint (h).

GMFS satisfies the goals stated at the beginning of this section:

- it is based on a single feature (the gradient magnitude),
- it consists of a sequence of well-known image processing operations that can be efficiently computed (convolution with Sobel filters and Gaussian filter, thresholding, morphology operations and connected component labelling algorithm),
- only four parameters regulate it: τ, σ, n_c, and n_o (see section IV-B).

B. SUFS

SUFS (Simplified U-net Fingerprint Segmentation) is a fingerprint segmentation method based on deep learning, with the following general characteristics:

- It uses a single network for the end-to-end segmentation task (from fingerprint **F** to segmentation mask **S**), except for straightforward preprocessing and postprocessing operations.
- The network architecture is inspired by U-Net [29], but with some modifications that make the network simpler, more symmetrical, and more suitable for the task of segmenting fingerprints (Table 1).

TABLE 1. Main differences between SUFS network and U-Net [29].

SUFS network architecture	Original U-Net architecture
Segmentation mask with the same size of the input fingerprint (512 × 512)	Segmentation map smaller than the input image (e.g., 572 × 572 input image and 388 × 388 segmentation map)
Convolutions with padding	Convolutions without padding
No cropping operations	Cropping operations needed
Single convolution at each level	Two subsequent convolutions
Batch normalization at each level	No batch normalization
Simple upsampling layers in the decoding path	Transposed convolutions used for upsampling

- Thanks to the use of standard layers and a simple loss function, the network can be easily implemented in any deep learning framework.
- Network training is carried out with a simple procedure using basic augmentation techniques on the learning data.

The network, like the traditional U-Net, has an encoding path and a decoding path. Both paths consist of six levels: at each level there is an encoder (decoder) block with the same structure.

An encoder block (Fig. 4) takes $\frac{f}{2}$ feature maps of size $d \times d$ as input (except for the first encoder, which takes a single-channel image: the input fingerprint), then applies a 3×3 convolution with padding and ReLU activation,

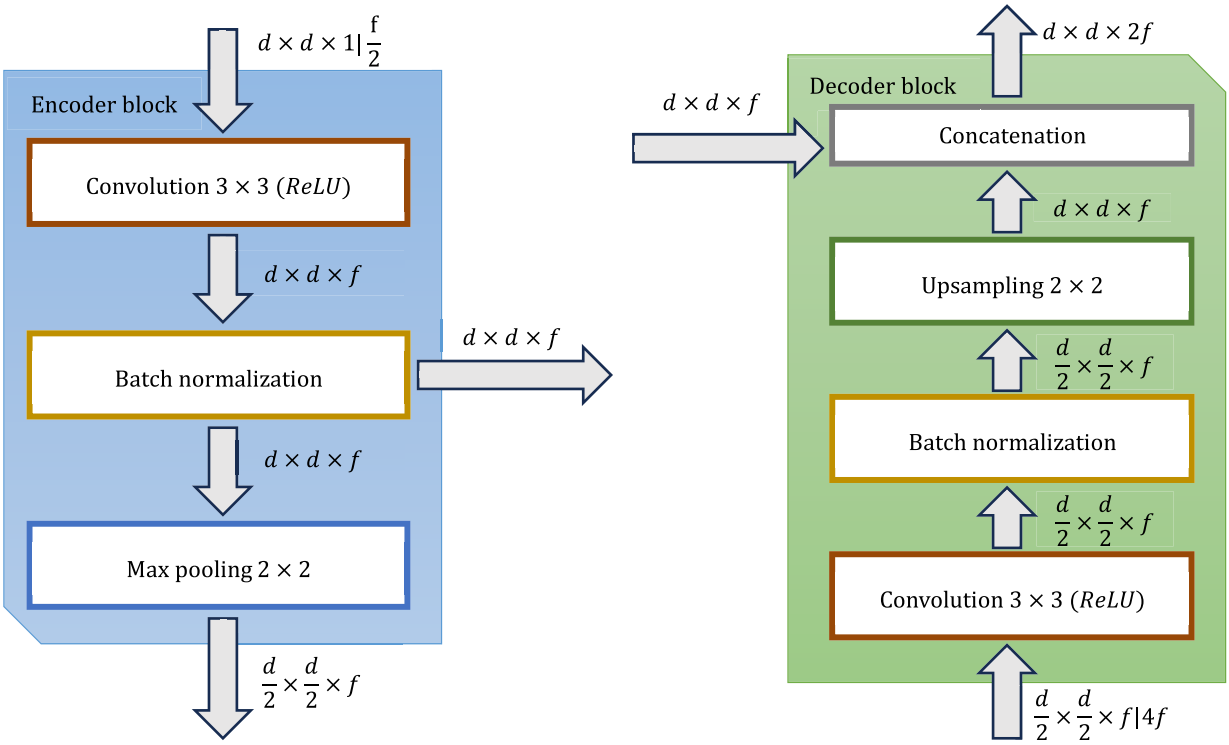

FIGURE 4. SUFS: the two building blocks of the network with their inputs, outputs, and intermediate layers.

batch normalization, and a 2×2 max pooling operation for downsampling. An encoder block produces two types of outputs: f downsampled feature maps of size $\frac{d}{2} \times \frac{d}{2}$ for the next level in the encoding path, and f feature maps at the original $d \times d$ size to be provided, through a skip connection, to the decoder block operating at the same resolution.

A decoder block (Fig. 4) takes $4 \cdot f$ feature maps of size $\frac{d}{2} \times \frac{d}{2}$ as input (except for the first decoder block, which takes f feature maps from the last encoder block), then applies a 3×3 convolution with padding and ReLU activation, batch normalization, and a 2×2 upsampling layer to produce f upsampled feature maps of size $d \times d$, which are concatenated to the same number of feature maps from the input skip connection, resulting in a final output of $2 \cdot f$ feature maps of size $d \times d$.

Fig. 5 provides a visual summary of the whole SUFS method. The preprocessing step adjusts the image size by adding or cropping borders. It is important to emphasize that the image is not resized, as doing so would alter its resolution. The network expects a 512×512 grayscale image as input: at the first level of the encoding path, 16 downsampled feature maps are extracted; each subsequent encoding level doubles the number of feature maps, while halving the size. The output of the last encoding block consists of 512 feature maps of size 8×8, which are provided as input to the first decoding block, together with 512 feature maps at the previous size through the skip connection. The first decoding level produces 1024 feature maps of size 16×16 and each subsequent decoding level halves the number of feature maps while doubling the size. The last decoding level produces 32 feature maps at the original 512×512 size, which are converted into a single feature map by a 3×3 convolution with padding and sigmoid activation. A final postprocessing step converts the network output to a binary image using 0.5 as a threshold and adjusts the image size by adding or cropping borders to obtain a segmentation map $\mathbf{S}$ with the same size of the input fingerprint $\mathbf{F}$.

The network architecture exhibits some symmetries. For instance, at each level, both the encoder block and its corresponding decoder block share identical f and d parameters. Additionally, the feature maps passing through the skip connections span from $512 \times 512 \times 16$ to $16 \times 16 \times 512$.

The above network architecture was selected as the most promising among some possible alternatives that were examined during a round of preliminary experiments on separate datasets (see section IV-C). In particular, the following options were considered:

- the standard U-net architecture and some of its variants,
- a greater or smaller number of feature maps f in each level,
- a greater or smaller number of levels,
- transposed convolution instead of normal convolution for the decoder block,
- transposed convolution with stride two in the decoder block, instead of the upsampling layer.

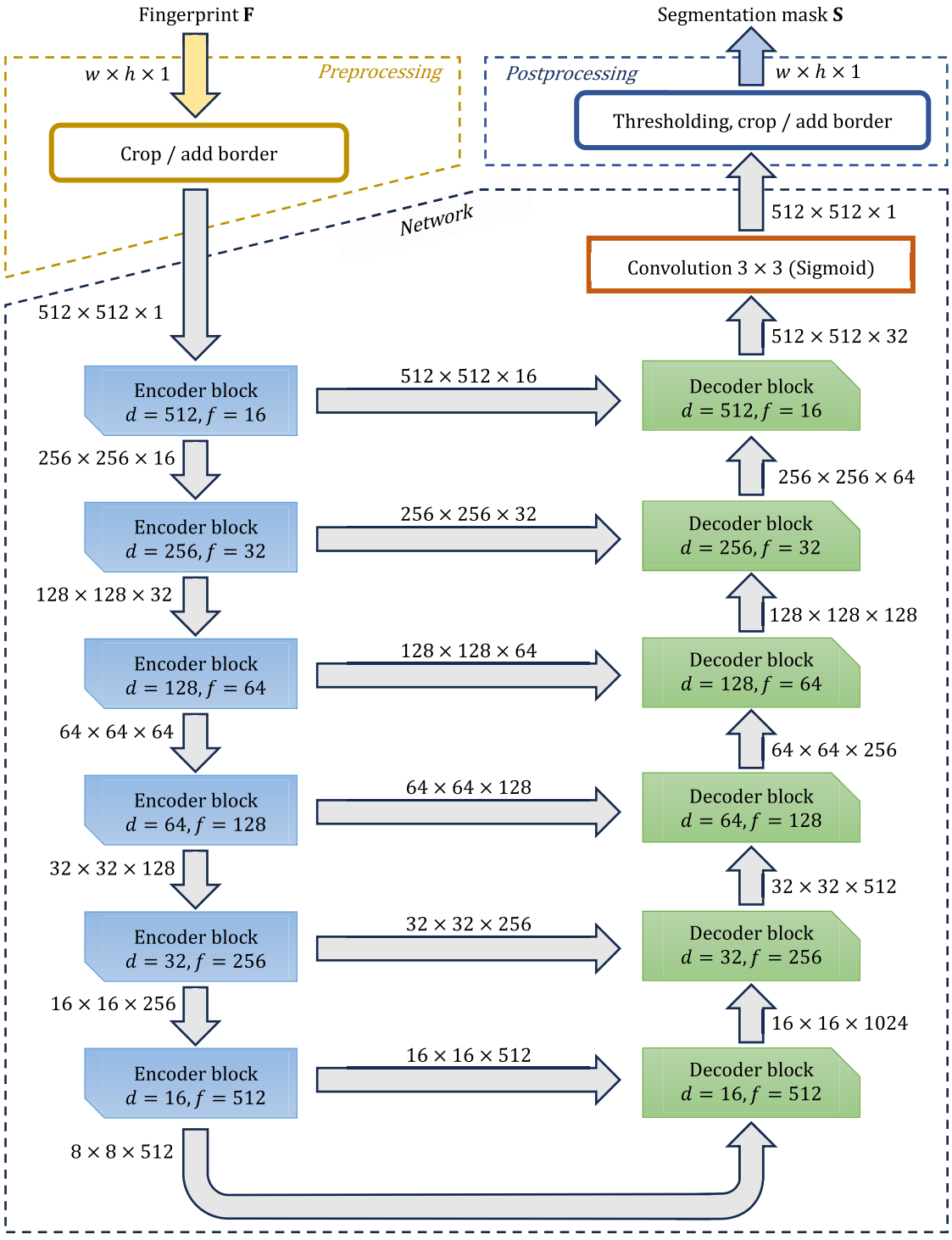

FIGURE 5. A visual summary of SUFS, including preprocessing, network architecture, and postprocessing.

To train the network, a loss function based on the Tversky index [36] was chosen:

$$loss\left(\hat{y}, y\right) = 1 - \frac{1 + \hat{y} \cdot y}{1 + \hat{y} \cdot y + \alpha\left(1 - y\right)\hat{y} + \left(1 - \alpha\right) y\left(1 - \hat{y}\right)} \tag{1}$$

where $\hat{y}$ is the true value, y is the predicted outcome, α is a parameter that weights false negatives, and one is added in numerator and denominator to ensure that *loss* is not undefined in edge cases. During the preliminary experiments, other loss functions were considered [37], including the Binary Cross-Entropy, the Focal Loss, the Dice Loss,

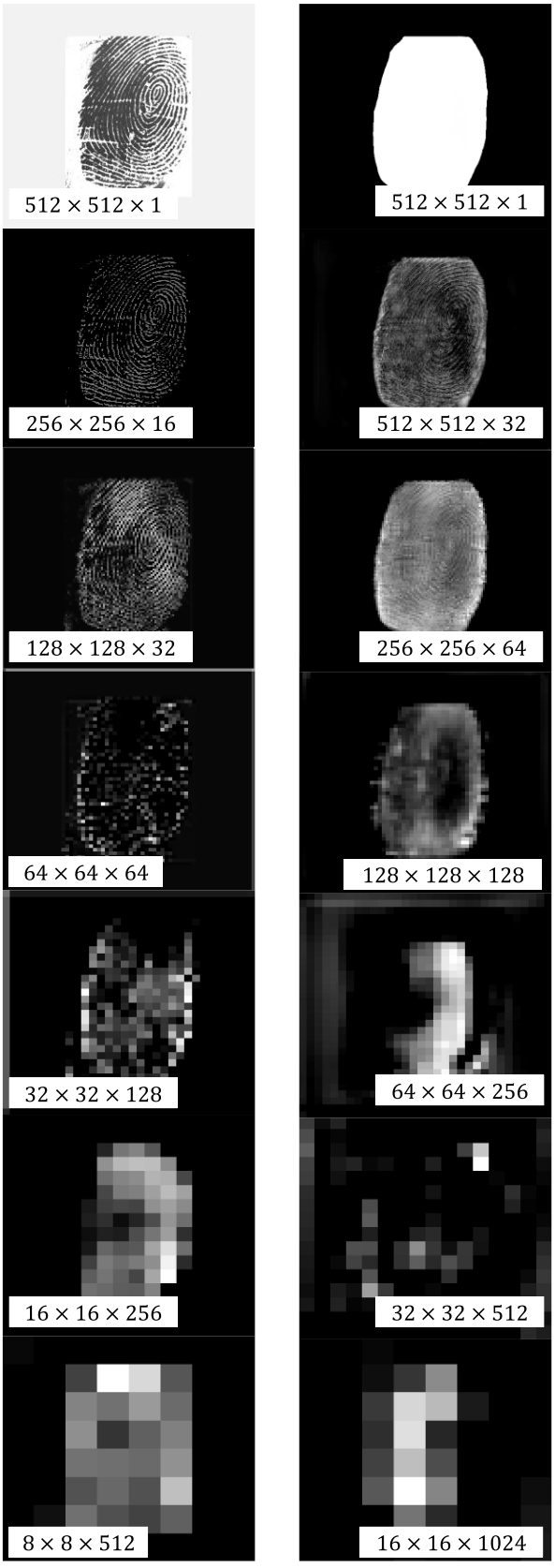

FIGURE 6. An example of input, feature maps, and output of the SUFS network. One of the feature maps for each level is shown.

and the Focal Tversky Loss. The loss function (1) was chosen as the most promising according to the results obtained, with $\alpha = 0.7$, the same value suggested in [38]. Network training is carried out with a batch size of 16, using the Adam optimizer [39] with a learning rate of 10^{-3}, which is progressively reduced to 10^{-5}. Additionally, simple data augmentation techniques are employed to artificially expand training data, introducing horizontal flips, small translations and rotations, and varying contrast and scale.

Fig. 6 shows an example of fingerprint segmentation with SUFS, from the network input to the corresponding output. For each encoder and decoder block, one of the feature maps is shown. Note that the size of the feature maps varies from 512 × 512 to 8 × 8, although they are all resized at the same dimensions for visualization purposes. Along the encoding path, the feature maps evolve from capturing fine-grained details and local patterns to representing higher-level contextual information and global structures. This process allows the network to extract progressively more abstract and meaningful features from the input fingerprint. In contrast, the decoding path reverses this trend, transforming feature maps from abstract spatial representations back into precise pixel-level segmentation masks.

IV. EXPERIMENTAL RESULTS

A. BENCHMARK AND METRICS

The publicly available fingerprint databases from the first three Fingerprint Verification Competitions (FVC2000 [40], FVC2002 [41], and FVC2004 [42]) are an established benchmark for fingerprint comparison algorithms [2], [43]. Thanks to the work of Thai, Huckemann, and Gottschlich, who made publicly available a manually marked segmentation ground truth for all those fingerprints [5], these databases are also a suitable benchmark for fingerprint segmentation, already adopted by recent published works in the field. Table 2 reports some general information on the benchmark databases; there are four databases for each competition: three are acquired from real fingers and the fourth one is generated using SFinGe [44], [45], a synthetic fingerprint generation method. Fig. 7 shows a sample fingerprint from each database, together with the corresponding segmentation ground truth. The benchmark covers a wide range of acquisition technologies and image sizes. As can be seen in Fig. 7, fingerprints from the various databases are highly heterogeneous with respect to their size, appearance, contrast, noise type, etc. This contributes to make the benchmark quite challenging. The image resolution is 500 dpi for all databases except two: FVC2002 DB2 and FVC2004 DB3. Since GMFS and SUFS are designed to work with 500 dpi images, the fingerprints of these two databases are resized to bring their resolution to 500 dpi before being provided as input to the two proposed methods. The resulting segmentation masks are then resized back to the original resolution before being compared to the ground truth.

TABLE 2. The twelve databases that comprise the benchmark (four for each competition).

FVC Database	Source	Image size	Resolution (dpi)	Set A	Set B
2000 1	Low-cost optical sensor	300×300	500	800 (100×8)	80 (10×8)
2000 2	Low-cost capacitive sensor	256×364	500	800 (100×8)	80 (10×8)
2000 3	Optical sensor	448×478	500	800 (100×8)	80 (10×8)
2000 4	Synthetic generator [44]	240×320	500*	800 (100×8)	80 (10×8)
2002 1	Optical sensor	388×374	500	800 (100×8)	80 (10×8)
2002 2	Optical sensor	296×560	569	800 (100×8)	80 (10×8)
2002 3	Capacitive sensor	300×300	500	800 (100×8)	80 (10×8)
2002 4	Synthetic generator [44]	288×384	500*	800 (100×8)	80 (10×8)
2004 1	Optical sensor	640×480	500	800 (100×8)	80 (10×8)
2004 2	Optical sensor	328×364	500	800 (100×8)	80 (10×8)
2004 3	Thermal sweeping sensor	300×480	512	800 (100×8)	80 (10×8)
2004 4	Synthetic generator [44]	288×384	500*	800 (100×8)	80 (10×8)
			Total number of fingerprints	9600	960

*Resolution simulated by the synthetic generator

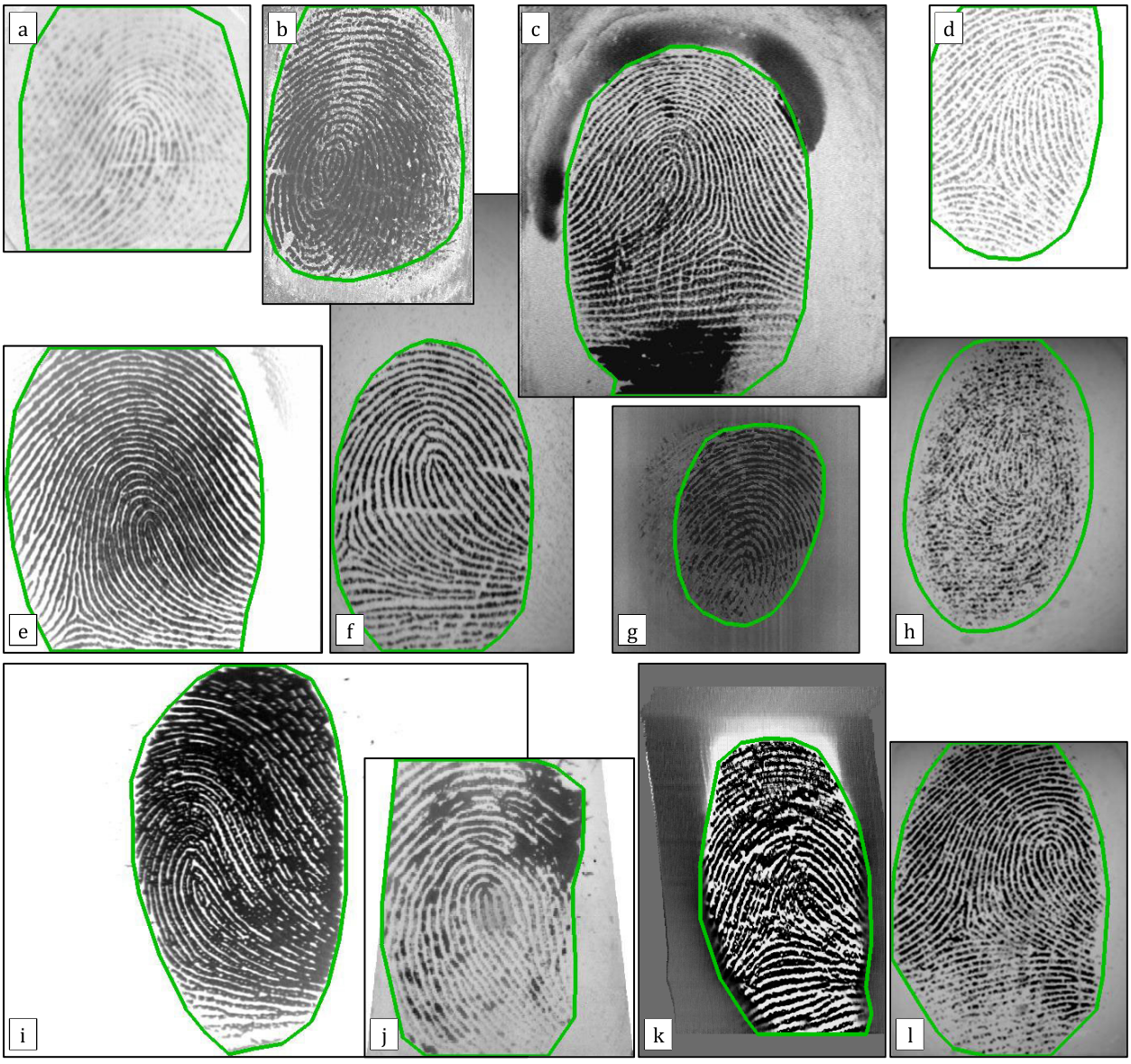

FIGURE 7. A fingerprint image for each database in FVC2000 [40] (a)-(d), FVC2002 [41] (e)-(h), and FVC2004 [42] (i-l): all images are at the same scale factor and with the contour of the corresponding segmentation mask provided by Thai, Huckemann, and Gottschlich [5].

TABLE 3. GMFS parameters chosen for each database.

FVC Database	τ	3σ	n_c	n_o
2000 1	0.18	7	3	12
2000 2	0.16	11	6	9
2000 3	0.22	13	6	6
2000 4	0.07	19	2	6
2002 1	0.03	11	4	6
2002 2	0.13	11	8	12
2002 3	0.25	15	4	9
2002 4	0.08	25	2	12
2004 1	0.02	7	8	15
2004 2	0.18	23	2	12
2004 3	0.12	15	4	12
2004 4	0.13	19	2	6

Each database is divided into two sets: A and B. Each set A contains 800 fingerprints from 100 different fingers (there are eight impressions for each finger), each set B contains 80 fingerprints from 10 different fingers. Following the same approach of previous papers, sets A are reserved for testing, while sets B are used for parameter tuning, model training and validation.

The accuracy of a segmentation mask $\mathbf{S}$ with respect to the corresponding ground truth $\mathbf{S}_{GT}$ is evaluated in terms of:

- True positives (TP) – the number of foreground pixels in $\mathbf{S}$ that are foreground in $\mathbf{S}_{GT}$.
- True negatives (TN) – the number of background pixels in $\mathbf{S}$ that are background in $\mathbf{S}_{GT}$.
- False positives (FP) – the number of foreground pixels in $\mathbf{S}$ that are background in $\mathbf{S}_{GT}$.
- False negatives (FN) – the number of background pixels in $\mathbf{S}$ that are foreground in $\mathbf{S}_{GT}$.

Three evaluation metrics are used in this paper: the classification error rate ER (2) is the percentage of incorrectly-classified pixels with respect to the total number of pixels in the image, the Dice coefficient DC (3) is a standard metric to quantify segmentation performance [46], and the Jaccard similarity coefficient JC (4) is a measure of similarity between finite sample sets [47], also referred to as intersection-over-union.

$$\mathrm{ER} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2}$$

$$\mathrm{DC} = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{3}$$

$$\mathrm{JC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{4}$$

B. GMFS PARAMETER SELECTION

GMFS is controlled by four parameters (see section III-A):

- τ and σ are related to the thresholding operation and should be tuned according to specific characteristics of the fingerprint acquisition sensor, such as background and image contrast.
- n_c and n_o are related to the postprocessing steps and should be tuned according to the typical amount of noise present in the fingerprints.

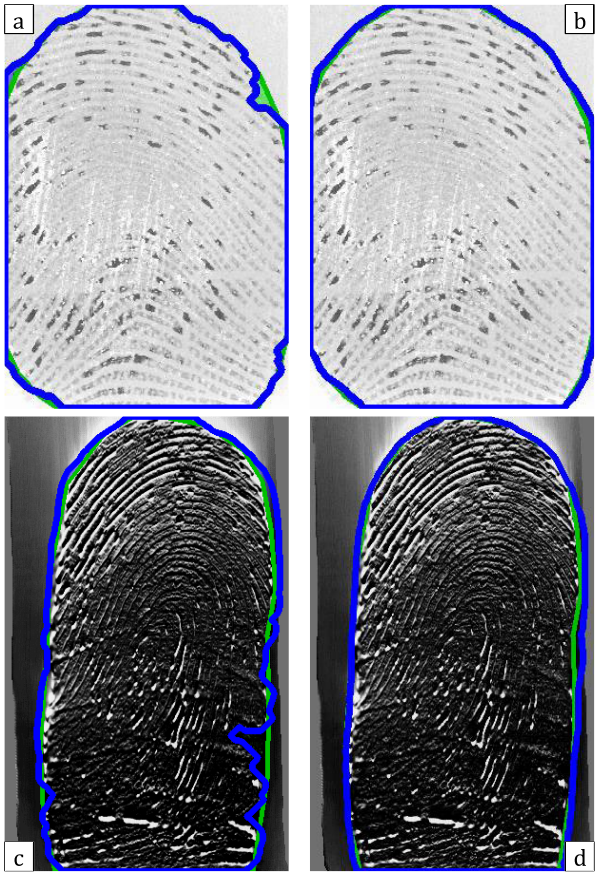

FIGURE 8. Examples of successful fingerprint segmentations using GMFS (a, c) and SUFS (b, d). The green contour represents the ground truth mask, while the blue contour represents the proposed segmentation.

In this experimentation, the values of the above parameters are chosen on set B of each database. The rationale for choosing the parameters for each database is that the twelve databases have been acquired (generated) with twelve different sensors (generation settings) and their images have specific properties (see Fig. 7 and table 2). Table 3 reports the parameter values used: for each database, the values that minimize the average *ER* over the corresponding set B have been chosen among a set of reasonable combinations of values.

It is worth noting that the smallest values of parameter τ (0.02 and 0.03) are chosen for the databases where the background tends to be uniform and less noisy (FVC2004 DB1 and FVC 2002 DB1, see Fig. 7.i and Fig. 7.e).

C. SUFS MODEL, HYPERPARAMETERS AND TRAINING

Unfortunately, the size of each set B (80 fingerprints from 10 fingers) is too small to train a deep neural network and the only reasonable option is to train a single network using all the fingerprints from B sets (960 in total). In fact, for SUFS

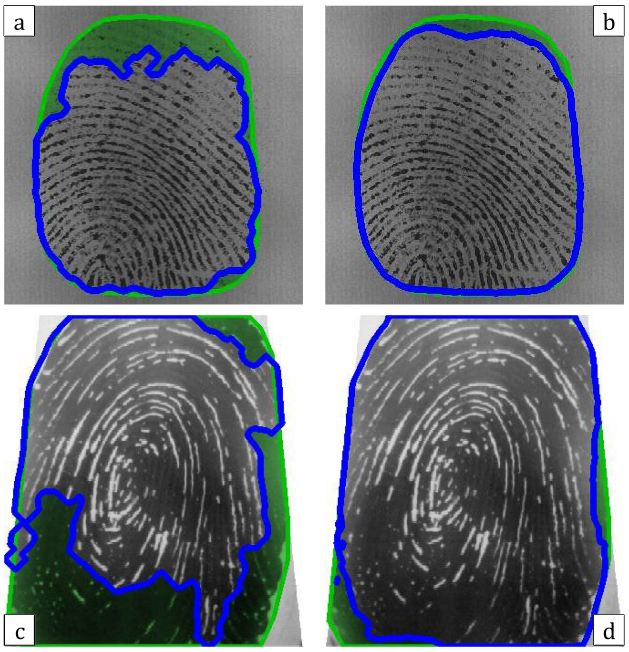

FIGURE 9. Examples of cases where GMFS (a, c) is less precise than SUFS (b, d). The green contour represents the ground truth mask, while the blue contour represents the proposed segmentation. The green areas correspond to false negatives.

preliminary experiments and training, these fingerprints are split into the following sets:

- B1 – 96 fingerprints (the eight fingerprints of the finger with index 101[2] from all B sets),
- B2 – 96 fingerprints (the eight fingerprints of the finger with index 102 from all B sets),
- B3 – 768 fingerprints (the eight fingerprints of the fingers with indices 103-110 from all B sets).

During the preliminary experiments to evaluate various options for the network architecture, and to choose the loss function and the hyperparameters, B3 is used as a training set, B1 as a validation set, and B2 as a test set to choose the most promising configuration.

The final training, with the network architecture described in section III-B, is carried out on B2 + B3, using B1 as a validation set.

D. RESULTS

GMFS is implemented in Python using the OpenCV library [48]. The time required to segment a fingerprint depends on the image size: on a PC with an Intel® Xeon® Silver 4112 CPU at 2.60GHz, the average segmentation time ranges from 7ms for FVC2000 DB4 images (the smallest ones) to 22ms for FVC2004 DB1 images (the largest).

SUFS is implemented in Python with the Keras library [49]. On a PC with an NVIDIA GeForce RTX™ 3080 Ti GPU, training the network requires about 25 minutes, while the average segmentation time is about 4ms, using batches of 32 fingerprints.

[2]In each FVC set B, the index of the first finger is 101, because the numbering continues from set A, which contains fingerprints from 100 fingers.

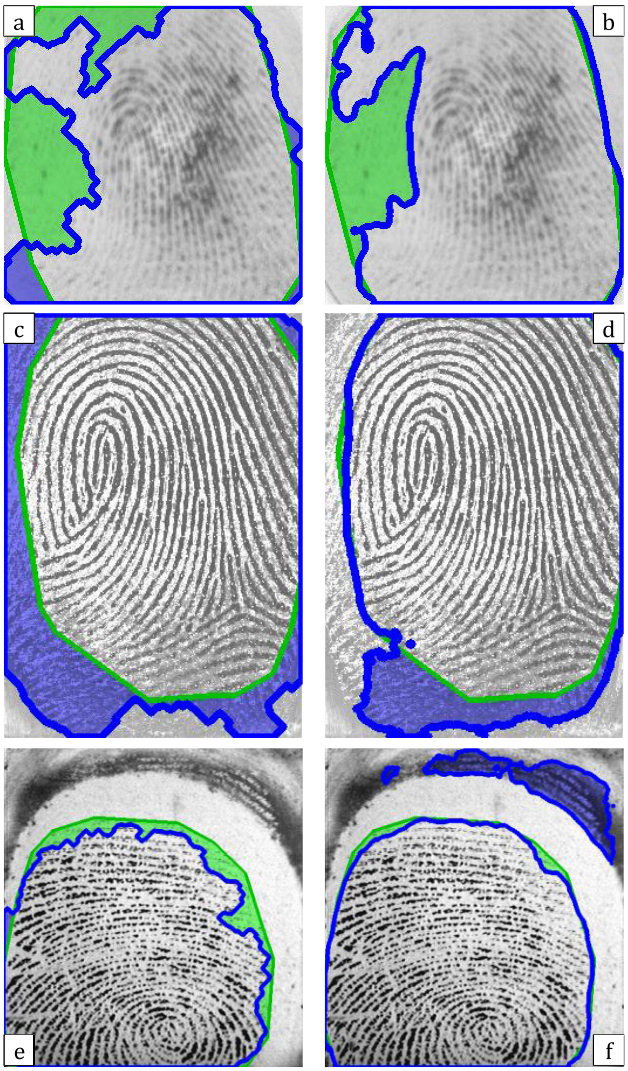

FIGURE 10. Examples of failure cases using GMFS (a, c, e) and SUFS (b, d, f). The green contour represents the ground truth mask, while the blue contour represents the proposed segmentation. The green areas correspond to false negatives, while the blue areas correspond to false positives.

Fig. 8 illustrates sample successful segmentation cases using GMFS and SUFS. Both methods exhibit robustness across the large variety of background and sensor noise that characterizes the benchmark (Fig. 7).

Fig. 9 shows two examples where GMFS produces less satisfactory results than SUFS: in Fig. 9.a, GMFS falsely classifies the top portion of the fingerprint as background due to its low contrast. In Fig. 9.c, GMFS misclassifies the bottom and right portions of the fingerprint due to their poor quality (the valleys are almost impossible to locate). In both cases, SUFS achieves significantly better results.

Fig. 10 shows three cases where both methods produce unsatisfactory results. In Fig. 10.a-b, low-contrast portions of the fingerprint are misclassified as background.

TABLE 4. A summary of the state-of-the-art methods considered and the two new proposed ones.

Method	Description	Deep learning	Available metrics
GFB	Gabor filter bank-based approach [11]		ER (from [5])
HCR	Harris corner response-based method [13]		ER (from [5])
MVC	Method based on local mean/variance and gradient coherence [9]		ER (from [5])
STFT	Short time Fourier transform based method [19]		ER (from [5])
FDB	Factorized directional bandpass method [5]		ER
G3PD	Method based on global three-part decomposition [22]		ER
NFIQ2	Segmentation step of the NFIQ2 software [51]		ER, DC, and JC (from [24])
Mind	Segmentation step of the fingerprint recognition software MindTCT [50]		ER, DC, and JC (from [24])
SAFIS	Segmentation step of the fingerprint recognition software SourceAFIS [52]		ER, DC, and JC (from [24])
FJ	Segmentation step of the fingerprint recognition software FingerJetFX [53]		ER, DC, and JC (from [24])
LNet	Patch-based segmentation with LeNet [25] [24]	✓	ER, DC, and JC
ANet	Path-based segmentation with AlexNet [26] [24]	✓	ER, DC, and JC
PCnet	Patch-based segmentation with a convolutional neural network [23]	✓	ER
CGan	Conditional Generative Adversarial Network [28]	✓	DC and JC (from [27])
Unet	U-Net [29]	✓	DC and JC (from [27])
CCnet	Convolution neural network with criss-cross attention module [30]	✓	DC and JC (from [27])
RUnet	Recurrent U-Net [31]	✓	DC and JC (from [27])
DRUnet	Recurrent U-Net with dropout [27]	✓	DC, and JC
GMFS	The proposed gradient magnitude-based segmentation method		ER, DC, and JC
SUFS	The proposed simplified U-Net segmentation method	✓	ER, DC, and JC

TABLE 5. Average ER computed on set a of each database in the benchmark*.

FVC DB	GFB	HCR	MVC	STFT	FDB	G3PD	NFIQ2	Mind	SAFIS	FJ	LNet	ANet	PCnet	GMFS	SUFS
2000 1	13.26	11.15	10.01	16.70	5.51	5.69	23.25	9.84	20.91	38.93	4.22	4.14	4.57	4.37	**1.71**
2000 2	10.27	6.25	12.31	8.88	3.55	4.10	8.00	10.80	5.39	6.31	4.39	4.31	3.32	4.02	**1.57**
2000 3	10.63	7.80	7.45	6.44	2.86	2.68	8.19	25.97	7.14	8.68	3.13	3.01	2.31	3.63	**1.27**
2000 4	5.17	3.23	9.74	7.19	2.31	2.06	10.51	5.63	7.60	13.85	2.51	2.57	2.37	2.04	**1.61**
2002 1	5.07	3.71	4.59	5.49	2.39	1.72	6.89	3.45	5.61	4.61	1.81	1.77	1.92	1.87	**1.60**
2002 2	7.76	5.72	4.32	6.27	2.91	2.83	12.20	5.58	5.18	7.42	3.33	3.35	2.28	2.65	**1.20**
2002 3	9.60	4.71	5.29	5.13	3.35	3.27	5.27	8.44	6.23	10.89	4.04	3.90	3.19	4.07	**2.09**
2002 4	7.67	6.85	6.12	7.70	4.49	3.63	15.46	6.59	8.85	15.51	4.96	5.09	2.50	3.28	**2.18**
2004 1	5.00	2.26	2.22	2.65	1.40	0.88	4.21	2.24	8.42	16.33	1.07	1.10	1.10	1.05	**0.75**
2004 2	11.18	7.54	8.06	9.89	4.90	4.62	8.82	10.62	12.15	10.57	3.53	3.39	2.97	3.93	**1.22**
2004 3	8.37	4.96	3.42	9.35	3.14	2.77	8.01	16.68	4.91	5.13	2.67	2.61	2.64	2.46	**1.50**
2004 4	5.96	5.15	4.58	5.18	2.79	2.53	13.03	5.86	5.24	8.94	3.50	3.40	2.26	2.39	**1.43**
Average	*8.33*	*5.78*	*6.51*	*7.57*	*3.30*	*3.06*	*10.32*	*9.31*	*8.14*	*12.26*	*3.26*	*3.22*	*2.62*	*2.98*	***1.51***

*Best values are in bold

In Fig. 10.c-d, fingerprint-like noise (likely a ghost-fingerprint left on the sensor by a previous acquisition) is misclassified as foreground. Similarly, in Fig. 10.f, a ghost-fingerprint portion at the top tricks SUFS into producing a false-positive region; this does not happen for GMFS (Fig. 10.e) because of postprocessing steps 2 and 5 (see section III-A). On the other hand, GMFS produces more false negatives on the same fingerprint.

E. COMPARISON WITH THE STATE-OF-THE-ART

The two proposed segmentation methods are compared with 18 state-of-the-art approaches. These are all the relevant

TABLE 6. Average DC computed on set a of each database in the benchmark*.

FVC DB	NFIQ2	Mind	SAFIS	FJ	LNet	ANet	CGan	Unet	CCnet	RUnet	DRUnet	GMFS	SUFS
2000 1	82.48	93.34	83.62	63.74	97.30	97.36	90.58	91.28	73.98	93.34	93.14	97.25	**98.92**
2000 2	94.72	93.31	96.54	95.78	97.14	97.20	91.10	88.53	79.66	92.39	93.58	97.43	**98.98**
2000 3	92.61	80.73	93.81	92.68	97.26	97.40	96.23	96.74	92.52	96.50	97.57	96.77	**98.89**
2000 4	90.93	95.39	93.47	86.86	97.99	97.93	95.11	95.44	83.03	97.04	97.36	98.35	**98.70**
2002 1	92.48	96.44	94.16	94.91	97.37	98.18	98.15	**98.50**	93.58	98.44	98.38	98.10	98.37
2002 2	89.74	95.72	95.89	93.79	97.37	97.37	95.18	95.84	90.88	97.28	97.40	97.93	**99.08**
2002 3	95.10	92.63	94.18	87.93	96.12	96.26	95.91	96.61	87.70	95.53	96.73	96.06	**98.02**
2002 4	86.02	94.14	92.00	84.33	95.75	95.65	93.69	94.02	90.08	95.32	95.54	97.20	**98.13**
2004 1	91.40	95.68	84.66	76.27	97.95	97.88	99.18	99.24	97.33	99.38	**99.49**	98.02	98.59
2004 2	92.29	91.19	89.48	88.46	96.84	96.97	97.58	96.60	92.34	96.69	97.93	96.57	**98.93**
2004 3	93.80	88.35	96.18	96.17	97.91	97.97	96.93	96.34	87.96	97.17	97.55	98.07	**98.84**
2004 4	88.18	95.14	95.54	91.77	97.14	97.20	97.14	96.95	91.65	97.21	98.03	97.97	**98.80**
Average	*90.81*	*92.67*	*92.46*	*87.72*	*97.18*	*97.28*	*95.57*	*95.51*	*88.39*	*96.36*	*96.89*	*97.48*	***98.69***

*Best values are in bold

TABLE 7. Average JC computed on set a of each database in the benchmark*.

FVC DB	NFIQ2	Mind	SAFIS	FJ	LNet	ANet	CGan	Unet	CCnet	RUnet	DRUnet	GMFS	SUFS
2000 1	76.46	78.61	89.98	60.51	95.78	95.86	83.56	84.79	62.08	88.15	87.97	94.72	**97.88**
2000 2	91.99	94.61	89.19	93.69	95.60	95.69	84.27	80.72	68.05	86.40	88.43	95.18	**98.00**
2000 3	91.80	92.85	74.02	91.32	96.87	96.98	92.96	93.78	87.06	93.74	95.39	93.99	**97.81**
2000 4	89.48	92.39	94.37	86.15	**97.49**	97.42	90.75	91.33	72.86	94.28	94.89	96.77	97.45
2002 1	93.11	94.39	96.54	95.38	96.66	**98.22**	96.39	97.07	88.37	96.95	96.83	96.29	96.82
2002 2	87.79	94.82	94.41	92.58	96.66	96.64	91.35	92.32	84.02	94.88	95.13	96.02	**98.18**
2002 3	94.73	93.77	91.55	89.10	95.95	96.10	92.38	93.56	80.05	91.83	93.87	92.72	**96.17**
2002 4	84.53	91.14	93.41	84.49	95.03	94.91	88.23	88.78	82.58	91.17	91.53	94.60	**96.35**
2004 1	95.78	91.57	97.76	83.66	98.92	98.89	98.38	98.50	94.91	98.78	**98.98**	96.15	97.25
2004 2	91.17	87.85	89.38	89.43	96.46	96.60	95.37	93.50	86.12	93.94	95.98	93.47	**97.89**
2004 3	91.99	95.09	83.31	94.86	97.33	97.39	94.12	93.03	80.29	94.62	95.29	96.23	**97.71**
2004 4	86.96	94.76	94.14	91.05	96.50	96.59	94.53	94.15	84.92	94.73	96.18	96.14	**97.69**
Average	*89.65*	*91.82*	*90.67*	*87.69*	*96.60*	*96.77*	*91.86*	*91.79*	*80.94*	*93.29*	*94.21*	*95.19*	***97.43***

*Best values are in bold

approaches for which results have been reported on the FVC benchmark. Table 4 lists all the twenty methods and specifies which evaluation metrics are available for each of them. Nine methods (including SUFS) are based on deep learning, while the other eleven (including GMFS) use traditional image processing operations and handcrafted features. Four methods come from software projects available in the public domain: Mind (from the MindTCT software developed by NIST [50]), NFIQ2 (from the NFIQ2 software [51]), SAFIS (from the open source fingerprint recognition software SourceAFIS [52]), and FJ (from the open source version of the FingerJetFX software [53]). The remaining methods

TABLE 8. Average results of traditional methods*.

Method	Average ER	Average DC	Average JC
GFB	8.33	-	-
HCR	5.78	-	-
MVC	6.51	-	-
STFT	7.57	-	-
FDB	3.30	-	-
G3PD	3.06	-	-
NFIQ2	10.32	89.65	90.81
Mind	9.31	91.82	92.67
SAFIS	8.14	90.67	92.46
FJ	12.26	87.69	87.72
GMFS	**2.98**	**95.19**	**97.48**

*Best values are in bold

TABLE 9. Average results of deep-learning-based methods*.

Method	Average ER	Average DC	Average JC
LNet	3.26	96.60	97.18
ANet	3.22	96.77	97.28
PCnet	2.62	-	-
CGan	-	91.86	95.57
Unet	-	91.79	95.51
CCnet	-	80.94	88.39
RUnet	-	93.29	96.36
DRUnet	-	94.21	96.89
SUFS	**1.51**	**97.43**	**98.69**

*Best values are in bold

are among the most often cited fingerprint segmentation methods in the scientific literature [5], [9], [11], [13], [19], [22], [23], [24], [27] or are based on well-known network architectures [25], [26], [28], [29], [30], [31].

Tables 5, 6, and 7 report all available results with ER, DC, and JC metric, respectively. Tables 8 and 9 report the average results over all databases for traditional and deep-learning-based methods, respectively.

SUFS exhibits impressive performance, surpassing the other methods in all the three metrics. Specifically, it outperforms the other methods on the ER metric in all databases, on the DC metric in ten out of twelve databases, and on the JR metric in nine out of twelve databases. Additionally, it outperforms the other methods on all metrics when considering the average over the twelve databases.

However, the excellent performance of SUFS must not overshadow the remarkable performance of GMFS, which achieves the second-best performance in two databases on the ER metric and in five databases on the DC metric. It also achieves the third-best performance on the average ER over all databases and the second-best performance on the average DC over all databases. Furthermore, if the comparison is limited to the other traditional (non-deep-learning-based) methods, GMFS obtains the best result on the ER metric in seven out of twelve databases, on the DC metric in all databases, and on the JR metric in nine out of twelve databases. Finally, when considering the average metrics over the twelve databases, it outperforms any other traditional method (Table 8). A further experiment has been carried out to evaluate the impact of GMFS post-processing steps: without them, its average ER grows from 2.98% to 4.01%, thus confirming the importance of GMFS post-processing.

Although comparing performance metrics is essential, the overall complexity of the methods must also be considered to assess their practical utility. The following paragraphs explore the complexity of the two top-performing traditional methods and deep-learning-based approaches, according to the average ER.

Among traditional methods (Table 8), GMFS (average *ER* 2.98%) and G3PD (average ER 3.06%) stand out for the lowest average ER. GMFS, described in detail in Section III-A, employs simple convolutions (with Sobel and Gaussian filters), thresholding, and minimal morphological operations: using an image processing library like OpenCV [48], it can be implemented in about 30 lines of Python code. This simplicity contrasts with G3PD, which requires solving an optimization problem to decompose the fingerprint into cartoon, texture, and noise components, followed by morphological operations to extract the segmentation from the texture coefficients [22]. While the source code is not publicly available (the authors only provide an obfuscated MATLAB implementation), based on the paper [22], it is reasonable to infer that G3PD implementation is significantly more complex than that of GMFS.

Turning to deep-learning-based methods (Table 9), SUFS (average ER 1.51%) and PCnet (average *ER* 2.62%) exhibit the lowest average ER. SUFS, detailed in Section III-B, utilizes an end-to-end neural network with minimal preprocessing and postprocessing steps. Using Keras [49], implementing the SUFS network architecture requires about 15 lines of Python code, and implementing the SUFS segmentation method, including pre- and post-processing, can be accomplished in about 20 lines. PCnet, in contrast, involves a multi-step process [23]: 1) decomposing the fingerprint into texture and cartoon components using a method similar to G3PD, 2) dividing the texture component into overlapping patches, 3) classifying each patch as foreground or background using a specifically trained neural network, and 4) applying morphological operations to obtain the final segmentation. Unfortunately, the source code is not available, but it can be concluded from the paper [23] that PCnet implementation is substantially more complex than that of SUFS.

V. CONCLUSION

This paper introduces two novel fingerprint segmentation methods, GMFS and SUFS, inspired by the KISS principle. Both methods are evaluated on a public benchmark, achieving

state-of-the-art performance while maintaining simplicity and computational efficiency.

GMFS, a straightforward method based on a single handcrafted feature, exhibits superior performance compared to all other traditional methods with available results on the benchmark, including some considerably more complex approaches. Notably, GMFS achieves comparable performance to a range of deep learning-based methods. Given its minimal computational resource requirements, GMFS is particularly well-suited for applications where computing power and memory are constrained.

SUFS leverages the power of deep learning using a simplified U-net architecture for the task of end-to-end fingerprint segmentation and exhibits impressive performance despite being trained on a relatively limited dataset. It can effectively segment fingerprints from databases acquired using a variety of technologies without requiring fine-tuning or specific parameter adjustments for each image type. SUFS surpasses all existing state-of-the-art methods, achieving an average ER of 1.51% across the entire benchmark. This represents a substantial improvement of over 40% compared to the previously best-performing method.

An open-source Python implementation of both methods is available at https://github.com/raffaele-cappelli/pyfing.

ACKNOWLEDGMENT

The author is grateful to Dr. Annalisa Franco for her insightful comments and to Sara Cappelli for proofreading the manuscript.

REFERENCES

[1] D. Maltoni, D. Maio, A. K. Jain, and J. Feng, *Handbook of Fingerprint Recognition*. Switzerland: Springer, 2022.

[2] D. Maltoni and R. Cappelli, "Fingerprint recognition," in *Handbook of Biometrics*, A. K. Jain, P. Flynn, and A. A. Ross, Eds. Boston, MA, USA: Springer, 2008, pp. 23–42.

[3] T. Dalzell, *The Routledge Dictionary of Modern American Slang and Unconventional English*. Evanston, IL, USA: Routledge, 2009.

[4] Wikipedia Contributors. (2023). *KISS Principle—Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/w/index.php?title=KISS_principle

[5] D. H. Thai, S. Huckemann, and C. Gottschlich, "Filter design and performance evaluation for fingerprint image segmentation," *PLoS ONE*, vol. 11, no. 5, pp. 1–31, May 2016.

[6] B. M. Mehtre, N. N. Murthy, S. Kapoor, and B. Chatterjee, "Segmentation of fingerprint images using the directional image," *Pattern Recognit.*, vol. 20, no. 4, pp. 429–435, Jan. 1987.

[7] B. M. Mehtre and B. Chatterjee, "Segmentation of fingerprint images—A composite method," *Pattern Recognit.*, vol. 22, no. 4, pp. 381–385, Jan. 1989.

[8] N. K. Ratha, S. Chen, and A. K. Jain, "Adaptive flow orientation-based feature extraction in fingerprint images," *Pattern Recognit.*, vol. 28, no. 11, pp. 1657–1672, Nov. 1995.

[9] A. M. Bazen and S. H. Gerez, "Segmentation of fingerprint images," in *Proc. Workshop Circuits, Syst. Signal Process. (ProRISC)*, 2001, pp. 276–280.

[10] X. Chen, J. Tian, J. Cheng, and X. Yang, "Segmentation of fingerprint images using linear classifier," *EURASIP J. Adv. Signal Process.*, vol. 2004, no. 4, pp. 1–15, Apr. 2004.

[11] L. Shen, A. Kot, and W. Koo, "Quality measures of fingerprint images," in *Audio- and Video-Based Biometric Person Authentication*, Berlin, Germany: Springer, 2001.

[12] F. Alonso-Fernandez, J. Fierrez-Aguilar, and J. Ortega-Garcia, "An enhanced Gabor filter-based segmentation algorithm for fingerprint recognition systems," in *Proc. 4th Int. Symp. Image Signal Process. Anal. (ISPA)*, 2005, pp. 239–244.

[13] C. Wu, S. Tulyakov, and V. Govindaraju, "Robust point-based feature fingerprint segmentation algorithm," in *Advances in Biometrics*, Berlin, Germany, 2007.

[14] E. Zhu, J. Yin, C. Hu, and G. Zhang, "A systematic method for fingerprint ridge orientation estimation and image segmentation," *Pattern Recognit.*, vol. 39, no. 8, pp. 1452–1472, Aug. 2006.

[15] L. Wang, H. Suo, and M. Dai, "Fingerprint image segmentation based on Gaussian-Hermite moments," in *Advanced Data Mining and Applications*, Berlin, Germany: Springer, 2005.

[16] P. M. Ferreira, A. F. Sequeira, and A. Rebelo, "A fuzzy C-means algorithm for fingerprint segmentation," in *Pattern Recognition and Image Analysis*, Cham, Switzerland: Springer, 2015.

[17] J. Kang and W. Zhang, "Fingerprint image segmentation using modified fuzzy C-means algorithm," in *Proc. 3rd Int. Conf. Bioinf. Biomed. Eng.*, Jun. 2009, pp. 1–4.

[18] W. Lei and Y. Lin, "A novel dynamic fingerprint segmentation method based on fuzzy C-means and genetic algorithm," *IEEE Access*, vol. 8, pp. 132694–132702, 2020.

[19] S. Chikkerur, V. Govindaraju, and A. N. Cartwright, "Fingerprint image enhancement using STFT analysis," in *Pattern Recognition and Image Analysis*, Berlin, Germany: Springer, 2005.

[20] C. Hu, J. Yin, E. Zhu, H. Chen, and Y. Li, "A composite fingerprint segmentation based on log-Gabor filter and orientation reliability," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2010, pp. 3097–3100.

[21] A. C. Pais Barreto Marques and A. C. Gay Thome, "A neural network fingerprint segmentation method," in *Proc. 5th Int. Conf. Hybrid Intell. Syst. (HIS)*, 2005, pp. 385–392.

[22] D. H. Thai and C. Gottschlich, "Global variational method for fingerprint segmentation by three-part decomposition," *IET Biometrics*, vol. 5, no. 2, pp. 120–130, Jun. 2016.

[23] X. Dai, J. Liang, Q. Zhao, and F. Liu, "Fingerprint segmentation via convolutional neural networks," in *Biometric Recognition*, Cham, Switzerland: Springer, 2017.

[24] P. B. S. Serafim, A. G. Medeiros, P. A. L. Rego, J. G. R. Maia, F. A. M. Trinta, M. E. F. Maia, J. A. F. Macêdo, and A. V. L. Neto, "A method based on convolutional neural networks for fingerprint segmentation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA, vol. 1, 2012, pp. 1097–1105.

[27] I. Joshi, R. Kothari, A. Utkarsh, V. K. Kurmi, A. Dantcheva, S. D. Roy, and P. K. Kalra, "Explainable fingerprint ROI segmentation using Monte Carlo dropout," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. Workshops (WACVW)*, Jan. 2021, pp. 60–69.

[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5967–5976.

[29] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, Cham, Switzerland: Springer, 2015.

[30] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-cross attention for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 603–612.

[31] W. Wang, K. Yu, J. Hugonot, P. Fua, and M. Salzmann, "Recurrent U-Net for resource-constrained segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2142–2151.

[32] J. Zhang, R. Lai, and C.-C. J. Kuo, "Latent fingerprint segmentation with adaptive total variation model," in *Proc. 5th IAPR Int. Conf. Biometrics (ICB)*, Mar. 2012, pp. 189–195.

[33] Y. Zhu, X. Yin, X. Jia, and J. Hu, "Latent fingerprint segmentation based on convolutional neural networks," in *Proc. IEEE Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2017, pp. 1–6.

[34] D.-L. Nguyen, K. Cao, and A. K. Jain, "Automatic latent fingerprint segmentation," in *Proc. IEEE 9th Int. Conf. Biometrics Theory, Appl. Syst. (BTAS)*, Redondo Beach, CA, USA, Oct. 2018, pp. 1–9.

[35] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4rd ed. London, U.K.: Pearson, 2018.
[36] A. Tversky, "Features of similarity," *Psychol. Rev.*, vol. 84, no. 4, pp. 327–352, Jul. 1977.
[37] S. Jadon, "A survey of loss functions for semantic segmentation," in *Proc. IEEE Conf. Comput. Intell. Bioinf. Comput. Biol. (CIBCB)*, Oct. 2020, pp. 1–7.
[38] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky loss function for image segmentation using 3D fully convolutional deep networks," in *Machine Learning in Medical Imaging*, Cham, Switzerland: Springer, 2017.
[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.
[40] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "FVC2000: Fingerprint verification competition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 402–412, Mar. 2002.
[41] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "FVC2002: Second fingerprint verification competition," in *Proc. Int. Conf. Pattern Recognit.*, 2002, pp. 811–814.
[42] R. Cappelli, D. Maio, D. Maltoni, J. L. Wayman, and A. K. Jain, "Performance evaluation of fingerprint verification systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 1, pp. 3–18, Jan. 2006.
[43] R. Cappelli, M. Ferrara, and D. Maltoni, "Minutia cylinder-code: A new representation and matching technique for fingerprint recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2128–2141, Dec. 2010.
[44] R. Cappelli, "Fingerprint synthesis," in *Handbook of Fingerprint Recognition*. Springer, 2022, pp. 385–426.
[45] R. Cappelli, D. Maio, and D. Maltoni, "An improved noise model for the generation of synthetic fingerprints," in *Proc. 8th Control, Autom., Robot. Vis. Conf. (ICARCV)*, 2004, pp. 1250–1255.
[46] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, Jul. 1945.
[47] P. Jaccard, "The distribution of the flora in the Alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, Feb. 1912.
[48] *OpenCV: Open Source Computer Vision Library*. Accessed: Dec. 21, 2023. [Online]. Available: https://opencv.org/
[49] *Keras*. Accessed: Dec. 21, 2023. [Online]. Available: https://keras.io/
[50] C. I. Watson, M. D. Garris, E. Tabassi, C. L. Wilson, R. M. McCabe, S. Janet, and K. Ko, "User's guide to NIST biometric image software (NBIS)," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST IR 7392, 2007.
[51] E. Tabassi, M. Olsen, O. Bausinger, C. Busch, A. Figlarz, G. Fiumara, O. Henniger, J. Merkle, T. Ruhland, C. Schiel, and M. Schwaiger, "NFIQ 2 NIST fingerprint image quality," Nat. Inst. Standards Technol., Tech. Rep. NIST IR 8382, 2021.
[52] R. Vazan. *SourceAFIS*. Accessed: Dec. 21, 2023. [Online]. Available: https://github.com/robertvazan/sourceafis-java
[53] DigitalPersona. *Fingerprint Feature Extractor, Open Source Edition*. Accessed: Dec. 21, 2023. [Online]. Available: https://github.com/FingerJetFXOSE/FingerJetFXOSE

RAFFAELE CAPPELLI received the degree in computer science from the University of Bologna, in 1998, and the Ph.D. degree, in 2002.

He is currently an Associate Professor with the Department of Computer Science and Engineering, University of Bologna, and a member of the Biometric System Laboratory, Cesena. He is one of the organizers of the fingerprint verification competitions (FVC2000-FVC2006 and FVC-onGoing) and one of the authors of the synthetic fingerprint generator SFinGe.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement