Journal of Electronic Imaging

JElectronicImaging.org

Forensic analysis of handwritten documents with GRAPHJ

Luca Guarnera
Giovanni Maria Farinella
Antonino Furnari
Angelo Salici
Claudio Ciampini
Vito Matranga
Sebastiano Battiato

SPIE. IS&T imaging.org

Luca Guarnera, Giovanni Maria Farinella, Antonino Furnari, Angelo Salici, Claudio Ciampini, Vito Matranga, Sebastiano Battiato, "Forensic analysis of handwritten documents with GRAPHJ," *J. Electron. Imaging* **27**(5), 051230 (2018), doi: 10.1117/1.JEI.27.5.051230.

Journal of Electronic Imaging 27(5), 051230 (Sep/Oct 2018)

Forensic analysis of handwritten documents with GRAPHJ

Luca Guarnera,[a] Giovanni Maria Farinella,[a] Antonino Furnari,[a,*] Angelo Salici,[b] Claudio Ciampini,[b] Vito Matranga,[b] and Sebastiano Battiato[a]
[a]University of Catania, Image Processing Laboratory, Dipartimento di Matematica e Informatica, Italy
[b]Raggruppamento Carabinieri Investigazioni Scientifiche, RIS di Messina, Messina, Italy

Abstract. We present GRAPHJ, a forensic tool for handwriting analysis. The proposed tool has been designed to implement the real forensic protocol adopted by the "Reparto Investigazioni Scientifiche" of Carabinieri, Italy. GRAPHJ allows the examiner to (1) automatically detect text lines and words in the document, (2) search for a specific character and detect its occurrences in the handwritten text, (3) measure different quantities related to the detected elements (e.g., heights and widths of characters), and (4) generate a report containing measurements, statistics, and the values of all parameters used during the analysis. The generation of the report helps to improve the repeatability of the whole process. The experiments performed on a set of handwritten documents show that GRAPHJ allows one to extract quantitative measures comparable to those acquired manually by an expert examiner. We also report a study on the use of the relative position of the superscript dot of the "i" characters as a parameter to infer the identity of the writer. The study has been performed using GRAPHJ and illustrates its value as a forensic tool. *© 2018 SPIE and IS&T* [DOI: 10.1117/1.JEI.27.5.051230]

Keywords: handwriting analysis; image forensics; manuscript investigation; writer identification.

Paper 180101SS received Jan. 26, 2018; accepted for publication Jul. 5, 2018; published online Jul. 27, 2018; corrected Oct. 19, 2018.

1 Introduction

The analysis of handwritten documents is the forensic process of detecting the peculiarities of human writing in order to assess the identity of the writer.[1–4] The process aims at detecting the distinctive traits of handwritten text arising from the specific motor patterns of the writer. Such traits include the shape of the stroke and the relative positions and dimensions of words and letters.

To properly reduce the bias of the forensic examiner, handwriting analysis requires the adoption of a quantitative and repeatable approach. Handwriting analysis should be also robust to the different possible variations in writing due to intrinsic and extrinsic causes, such as, for instance, different writing speeds, dissimulation, and available space. This has fostered research on approaches based on graphometry, which take into account different quantifiable features of handwritten text.[2,5–12]

Despite the guidelines offered by the aforementioned studies, forensic handwriting analysis still tends to be a highly unstructured process, always guided, and potentially biased by the experience of the forensic experts. To standardize the process and simplify its documentation and repeatability, we propose GRAPHJ, an automated tool for handwriting analysis. The tool implements several algorithms for the automated detection of the relevant elements of a handwritten document, as well as for the measurement of important quantitative measurements. For instance, GRAPHJ implements the automated detection of text lines and words in a handwritten document and allows one to search for all occurrences of specific characters. It also allows one to generate a report of the performed analysis including quantitative statistics and measurements, thus simplifying the reporting phase of the process. GRAPHJ has been developed to follow and support the handwriting analysis protocol of the RIS (Reparto Investigazioni Scientifiche, Arma dei Carabinieri), Italy. As such, it has been tested and is currently considered in the protocol used by RIS, Carabinieri in Italy, where it has proved to be a powerful tool to improve and automate the objective analysis of handwritten documents.

In the following, we first discuss the related works in Sec. 2. In Sec. 3, we present GRAPHJ, describing the typical workflow and the details of all algorithms therein involved. In Sec. 4, we report experiments to evaluate the performance of GRAPHJ with respect to standard examination techniques carried out by experts. We also investigate whether the relative placement of the superscript dot on "i" characters can be used as a parameter to infer the identity of the writer. The reported analysis has been carried out using GRAPHJ. Section 5 concludes the paper.

2 Related Works

Many researchers have considered the problem of handwritten document analysis to assess the identity of the writer. Hayes[3] highlighted the importance of absolute dimensions, which reflect the movements of hands and fingers influenced by individual expression. For instance, some people produce small writing, whereas others are characterized by a taller one. The work of Koppenhaver[10] revealed that the study of character heights is valuable for the identification of the range of variability of the writer. Morris[11] pointed out the importance of analyzing the dimensional parameters

*Address all correspondence to: Antonino Furnari, E-mail: furnari@dmi.unict.it

comparing absolute and relative quantities. Such an analysis based on the assessment of speed, slope, and style can be useful to identify an attempt of forgery. Kelly and Lindblom[12] studied the ratio of the heights of lowercase and uppercase letters and showed that such value can be used to identify the writer.

Other researchers proposed computational approaches to perform writer identification and verification. Schomaker[13] revised general background and methods for writer identification and verification. Brink et al.[14] proposed to encode a subject's handwriting as a mixture of handwritings of typical writers. Schlapbach et al.[15] proposed a method to identify the author of text handwritten on a white-board. To perform the analysis, the authors collected a dataset comprising samples from 200 different writers (DW). Bulacu and Schomaker[16] designed a method to extract probability distribution functions from handwriting images to characterize writer individuality. The method is developed to be independent of the textual content of the documents. Saad[17] designed a system for writer identification from offline Arabic handwritten text. The method combines fuzzy logic and genetic algorithms to fuse the extracted features and cope with the ambiguity of handwriting similarities. Shivram et al.[18] proposed to model writing styles as a shared component of an individual's handwriting. To this aim, they developed a theoretical framework based on latent Dirichlet allocation. Chahi et al.[19] designed the "blockwise local binary count" operator to tackle writer identification of handwritten documents. The descriptor characterizes the writing style of each writer computing a set of histograms from the connected components detected in the writing. He and Schomaker[20] engineered two textural-based descriptors for writer identification, namely LBPruns and COLD. The features are designed to capture the line information of handwritten texts rather than curvature information. Christlein et al.[21] described a method for writer identification that makes use of densely sampled RootSIFT descriptors and GMM supervectors for feature encoding. Exemplar-SVMs are used to train a document-specific similarity measure. Hannad et al.[22] proposed an approach for writer identification that splits handwriting into small fragments. Each fragment is described using texture-based descriptors including histograms of local binary patterns, local ternary patterns, and local phase quantization. Two documents are hence compared by computing the distance between the extracted descriptors.

Handwriting analysis tasks different from author identification have also been investigated. For instance, Bhardwaj et al.[23] tackled content-based retrieval of handwritten document images by searching of similar handwriting styles corresponding to a given query image. Ramaiah et al.[24] estimated the approximate age of historical handwritten documents learning a distribution over different styles across centuries.

While the aforementioned works investigated methods for automatic analysis of handwritten documents, expert examination still plays an important role in the forensic practice. Manual and automatic tools have been proposed to aid this analysis. Among the others, Fabiańska et al.[25] proposed Graphlog. The tool allows the operator to select all relevant elements of the text (e.g., words and characters) in order to extract quantitative measurements such as distances, angles, and proportions. Other commercial tools have also been proposed,[26] in contrast to previous tools, GRAPHJ has been designed to follow the well-established protocol for handwritten document analysis employed by RIS, Carabinieri, Italy. Moreover, GRAPHJ allows for the automated detection of elements in order to reduce the amount of required manual intervention. GRAPHJ has been designed to be a multimedia forensics tool[27] aimed at assisting the examiner in analyzing digitalized handwritten documents. To the best of our knowledge and considering the cooperation with RIS, GRAPHJ is the first tool that follows a real protocol. Note that this tool has been now adopted by RIS for their scientific analysis of forensic cases. Table 1 summarizes the differences between GRAPHJ and other two related tools: Graphlog[25] and Masquerade.[26]

Table 1 Comparison of GRAPHJ with Graphlog and Masquerade. A, automatic; M, manual; NA, not available.

	GRAPHJ	Graphlog[25]	Masquerade[26,28,29]
Search of text lines	A	NA	A
Search of words	A	NA	A
Character search	A	NA	A
Search of writing areas	A	NA	A
Height characters	A	M	A
Width characters	A	M	A
Distance between words	A	M	A
Distance between characters	A	M	A

3 GRAPHJ

GRAPHJ allows one to automate many procedures for handwriting analysis. In particular, it allows one to carry out three automated procedures that are relevant to forensic analysis: (1) detection of elements of handwritten documents such as text lines and words, (2) search of instances of a specific characters throughout the document, and (3) measurement of quantities such as the distances between words and characters, and the heights and widths of characters. All procedures are automated in order to limit the required amount of manual intervention. Nevertheless, the examiner can adjust the relevant parameters at any moment. GRAPHJ has been developed as a plugin for the ImageJ framework,[30] which is a standard tool to perform image processing tasks. It presents a graphical user interface that guides the operator through the analysis of a handwritten document, allowing to perform the required operations, monitor the output of the various steps, and adjust the relevant parameters. The tool also allows the generation of a report detailing the analysis performed and reporting the values of all measured quantities. The generated report allows one to improve the repeatability of the analysis, which is a key point in forensic science. Figure 1 shows the typical workflow suggested to perform handwriting analysis using GRAPHJ. A video demo of GRAPHJ is available at our web page.[31]

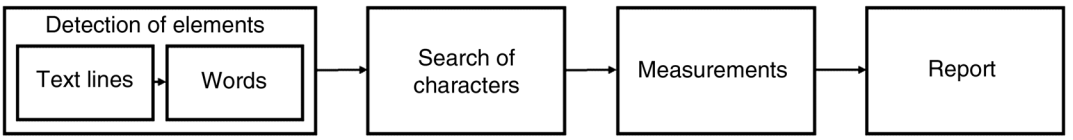

Fig. 1 The typical workflow of handwriting analysis in GRAPHJ.

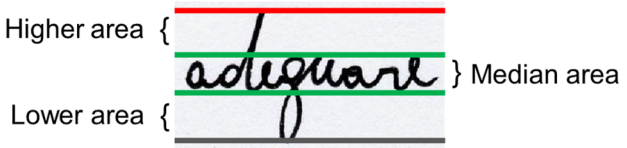

Fig. 2 Higher, median, and lower areas of a text line.

3.1 *Automated Search of Text Lines*

We assume text lines to be composed by three areas: higher, lower, and median areas, as shown in Fig. 2. GRAPHJ detects text lines in two steps: (1) detection of median areas and (2) detection of the lower and higher areas corresponding to each previously detected median area.

Figure 3 shows the procedure employed to search for median areas while Algorithm 1 reports the related pseudocode. The procedure is described in the following:

1. The image is binarized setting to 0 all pixels exceeding a given threshold $\mathcal{T}$ and setting to 1 (white color) all other pixels. The resulting binary image is denoted by B;
2. A histogram H_r, counting the number of black pixels contained in each row is computed (the histogram is calculated through the *histRows()* function, see Algorithm 1, line 3). The histogram is a vector of N integers, where N is the number of rows in the image (i.e., the height of the image). $H_r[i]$ contains the number of black pixels in the i'th row of the binary image B;
3. The positions of the central lines of median areas are detected considering all the peaks of the histogram which values are above a threshold s_1 specified by the user (Algorithm 1, line 6). The peaks are computed using the *findPeaks()* function, which returns both position and values of the peaks, see Algorithm 1, line 5. The threshold s_1 is chosen by the operator to minimize the influence of noise;
4. The algorithm proceeds to detect the starting (j) and ending (k) row indexes of each median area.

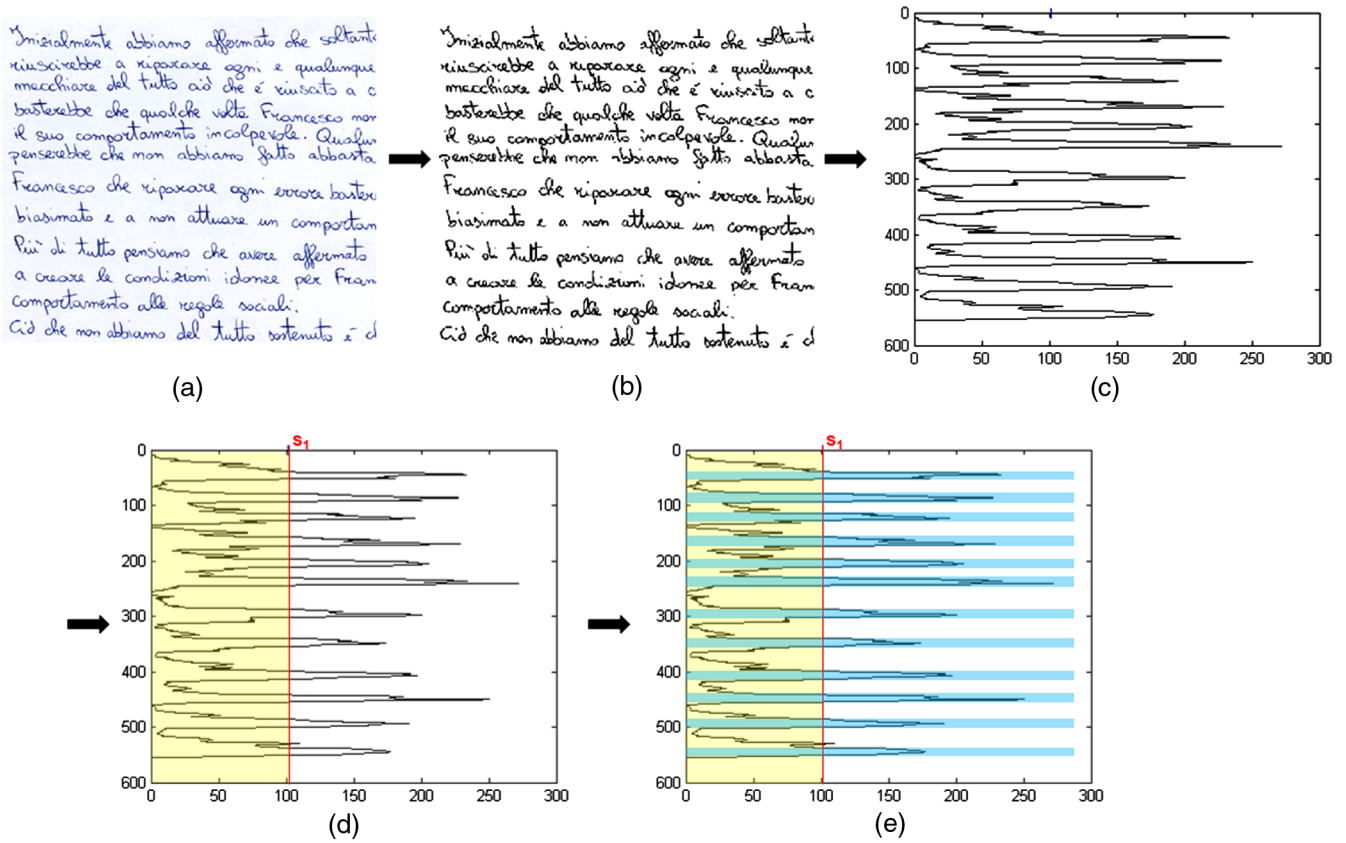

Fig. 3 Automated search of text lines. (a) Input image. (b) Binary image. (c) Per-row histogram of the binary image. (d) Search of median areas (dependent on threshold s_1). (e) Search of the starting and the ending rows of higher and lower areas.

Algorithm 1 Detection of median areas.

1: **Input:** Binary image B, threshold s_1
2: **Output:** Histogram H_r, indices *ind*
3: H_r = histRows(B)
4: $i = 0$
5: **for all** $[indMax, valMax] \in$ findPeaks(H_r) do
6: **if** $valMax > s_1$ then
7: $val = valMax/4$
8: **for** $j = indMax$; $j \geq 0$ & &$H_r[j] > val$; $j = j - 1$ do
9: **end for**
10: **for** $k = indMax$; $k < length(H_r)$ & &$H_r[k] > val$; $k = k + 1$ do
11: **end for**
12: $ind[i] = j$
13: $ind[i + 1] = k$
14: $i = i + 2$
15: **end if**
16: **end for**

Algorithm 2 Detection of corresponding higher and lower areas.

1: **Input:** Indices *ind*, histogram H_r
2: **Output:** List *areas*
3: Set *areas* to an empty list
4: **for** $i = 0$; $i < length(ind)$; $i = i + 2$ **do**
5: **for** $j = ind[i]$; $j > 0$ & &$H_r[j] \neq 0$; $j = j - 1$ **do**
6: **end for**
7: **for** $k = ind[i + 1]$; $k < length(H_r) - 1$ & &$H_r[k] \neq 0$; $k = k + 1$ **do**
8: **end for**
9: **if** $i > 0$ & &$(H_r[j] \neq 0 || j \leq ind[i - 1])$ **then**
10: j = arg $\min_h(H_r[h = ind[i - 1]], H_r[h = ind[i]])$
11: **end if**
12: **if** $i < length(ind) - 2$ & &$(H_r[k] \neq 0 || k \geq ind[i + 2])$ **then**
13: k = arg $\min_h(H_r[h = ind[i + 1]], H_r[h = ind[i + 2]])$
14: **end if**
15: Append tuple $(j, ind[i], ind[i + 1], k)$ to list *areas*
16: **end for**

Considering that the values of the histogram are expected to decay gradually in a neighborhood of the corresponding peak, the detection of starting and ending points of the median areas is done by searching for the nearest lower and higher rows whose value is over $\frac{1}{4}$ of the peak value (see Algorithm 1, lines 7–14);

5. The algorithm returns the position of the median areas in the form of a list of starting and ending row indexes *ind*.

After the detection of median areas, Algorithm 2 is employed to detect the starting (j) and ending (k) points of higher and lower areas. For each pair of indexes contained in the variable *ind* (lines 4 to 16 of Algorithm 2), the procedure searches for the indexes of the nearest upper (lines 5 and 6 of Algorithm 2), and lower (lines 7 and 8 of Algorithm 2) pixel rows containing only zero valued pixels. Since the histogram H_r contains the number of nonzero valued pixels in each row of the binary image, the aforementioned pixel rows can be easily detected by searching for the corresponding elements of H_r which value is equals to zero (see the conditions of the for loops in lines 5 and 7 of Algorithm 2). The algorithm returns a list of tuples containing four values: starting row index of the higher area, starting row index of the median area, ending row index of the median area, and ending row index of the lower area. Each tuple represents the position of a text line.

3.2 *Automated Detection of Words*

The detection of words is carried out starting from the position of text lines detected in the binary image B. To this aim, for each line, we extract an image patch L from the binarized image B. The detection of words is carried out on an image patch L in two steps:

1. The boundaries (starting and ending pixel columns) of each word are detected using a threshold chosen by the operator. The threshold is referred to as s_2 in Algorithm 3;
2. The correct orientation of each word is determined and the positions of higher and lower areas are refined for each word.

The first step of the procedure is summarized in Algorithm 3 and shown in Fig. 4. A column histogram H_c counting the number of black pixels contained in each column of L is computed (the histogram is calculated using the *histCols()* function, see line 4 of Algorithm 3). Note that the computation of H_c is similar to the computation of H_r. To find the boundaries of each word, the algorithm searches for sequences of contiguous columns containing only white pixels (lines 7 to 16 of Algorithm 3). This is achieved considering the bins of H_c that contain zero values. If the detected gap is larger than a given threshold s_2, then the starting and ending column indexes of a new word are appended to the words list (lines 12 to 15 of Algorithm 3). The algorithm

Algorithm 3 Detection of words.

1: **Input:** Image patch of a text line L, threshold s_2

2: **Output:** List *words*

3: Set *words* to an empty list

4: H_c = histCols(L)

5: $i_s = 0$

6: $i_e = 0$

7: **While** $i_s < length(H_c)$ & &$i_e < length(H_c)$ **do**

8: **for**; $H_c[i_s] == 0$; $i_s = i_s + 1$ **do**

9: **end for**

10: **for** $i_e = i_s + 1$; $H_c[i_e] \neq 0$; $i_e = i_e + 1$ **do**

11: **end for**

12: **if** $i_e - i_s \geq s_2$ then

13: Append tuple (i_s, i_e) to list *words*

14: $i_s = i_e + 1$

15: **end if**

16: **end while**

eventually returns a list of tuples containing the starting and ending indexes (i_s, i_e).

Once the boundaries of each word have been detected, the algorithm refines the position of median, higher, and lower areas for each word. This step is necessary because words on the same text line may have different size and orientation. The procedure is reported in Algorithm 4 and described in the following. To perform the refinement procedure, an image patch w is extracted for each word from the binary image B considering the detected row and column indexes. The rotation correction procedure in Ref. 32 is used to detect the correct orientation of a given word. The procedure works as follows: the corresponding image patch w is rotated by different angles α sampled from the interval $[-N, N]$ at step k (lines 7 and 14 of Algorithm 4, function rotate(w, α) rotates word w by α degrees). For each rotated patch w, we compute a row histogram H_w using the function *histRows* (line 8 of Algorithm 4). The correct orientation is retrieved by selecting the angle α for which the value $\max(H_w)$ is maximized (lines 9 to 12 of Algorithm 4). This outlined procedure arises from the observation that, if the word is aligned horizontally, the histogram H_w will be strongly peaked. Once the correct orientation has been determined, the positions of the correct median, higher, and lower areas are computed using Algorithms 1 and 2 (see lines 15 and 16 of Algorithm 4).

3.3 *Automated Search of Characters*

GRAPHJ allows one to search for all instances of a given character in the handwritten document under analysis. To this aim, the graphical interface allows the examiner to select a bounding box around the desired character to be searched. The definition of the bounding box allows one to select an image patch T that serves as a template. The algorithm hence performs a sliding window search over the whole document in order to locate all possible occurrences of selected

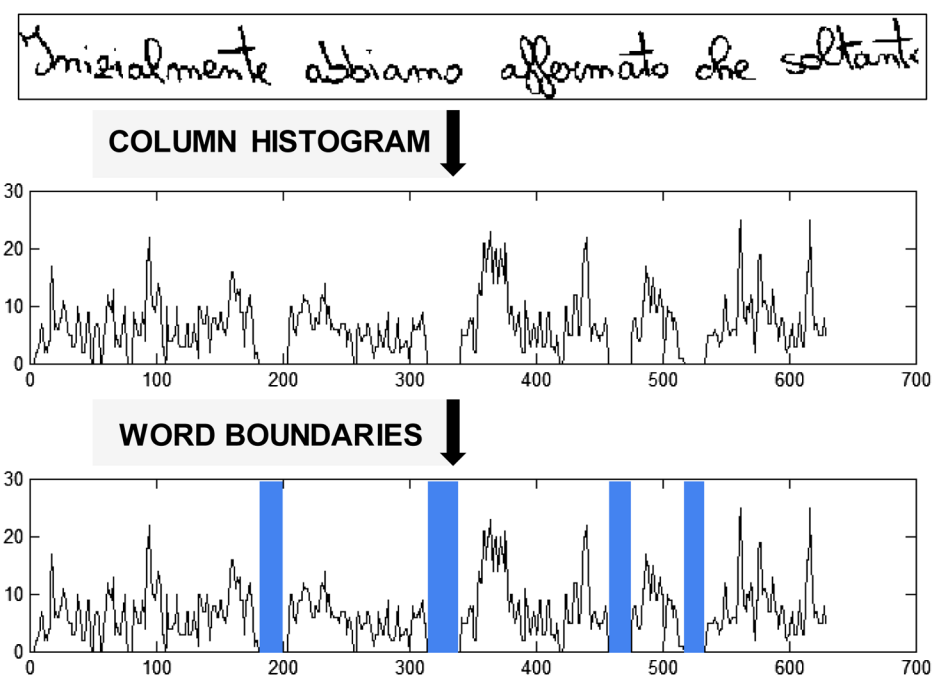

Fig. 4 Automated detection of words.

Algorithm 4 Detection of median, higher, and lower areas for each word.

Input: Array of image patches *words*, angle range width N, angle step k
Output: List of orientations and line indexes *word_areas*
  for all $w \in words$ do
    maxValue = 0
    $\beta = 0$
    for $\alpha = -N$; $\alpha <= N$; $\alpha = \alpha + k$ **do**
      w_r = rotate(w, α)
      H_w = histRows(w_r)
      if max$(H_w) >$ *maxValue* then
        maxValue = max(H_w)
        $\beta = \alpha$
      end if
    end for
    w_r = rotate(w, β)
    Determine i_1, i_2, i_3, i_4, indexes of median, higher and lower areas of w_r using Algorithms 1 and 2
    Append tuple $(\beta, i_1, i_2, i_3, i_4)$ to list *word_areas*
  end for

Algorithm 5 Scoring function for automated search of characters

Input: Template T, window W, and image height *im_height*
Output: Score *sc*
completed = False
$sc = 0$
while not completed **do**
  x_T, y_T = find coordinates of next black pixel in template T
  x_W, y_W = find coordinates of next black pixel in template W
  $y_T^{prev} = 0$
  $y_W^{prev} = 0$
  while $y_W <$ *im_height* && $y_T <$ *im_height* **do**
    if $||y_T^{prev} - y_T| - |y_W^{prev} - y_W|| < |y_T^{prev} - y_T|/2$ **then**
      $sc = sc + 1$
    end if
    $y_T^{prev} = y_T$
    $y_W^{prev} = y_W$
    $y_T = y_T + 1$
    $y_W = y_W + 1$
  end while
  if no more black pixels to analyze in T or W **then**
    completed = True
  end if
end while

character. The size of the search window W is chosen to be equal to the one of the template T. In order to join robustness to small rotations, during the search, two additional candidate patches are generated by rotating the content of each search window by 10 deg and -10 deg. For each candidate search window, a score S_W is computed using the procedure outlined in Algorithm 5 and discussed in the following. Search windows with scores larger than a threshold set by the operator are highlighted in the document as correctly detected character instances.

The scoring function reported in Algorithm 5 counts the number of black pixels contained in template T, which are present in window W. The scoring is performed scanning columnwise both the template T and the search window W.

3.4 *Measures*

GRAPHJ also allows one to measure some quantitative information about words and characters in an automated fashion. In particular, the algorithm implements two functions:

- automatic computation of the biaxial proportions and their average;
- automatic computation of the side expansions and their average.

3.4.1 *Automatic computation of the biaxial proportions and their average*

Biaxial proportions are the width and the height of the oval characters (see Fig. 5). To convert such measures from pixels to millimeters, we use the dedicated functions offered by ImageJ. For each character, GRAPHJ computes the average $\rho_i = \frac{w_i}{h_i}$, where w_i and h_i are width and height of the i'th characters, respectively.

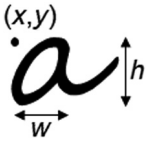

Fig. 5 Coordinates (x, y) of a given character and biaxial proportions (width and height).

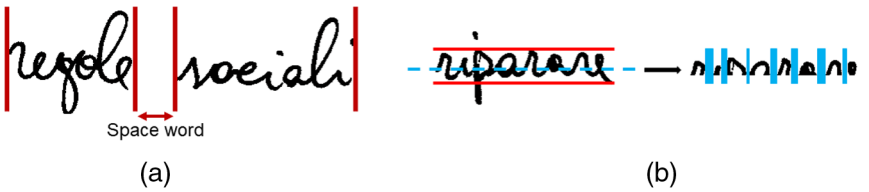

Fig. 6 (a) Computation of the space between words. (b) Computation of the spaces between characters. Only the median part of the words is retained to facilitate character segmentation.

3.4.2 *Automatic computation of the side expansions and their average*

The side expansions are the distances between the characters of a word and the distances between words. Distances between words are easily computed using the previously retrieved starting and ending indexes computed using Algorithm 3. The distances between characters are computed following a similar procedure. To remove the influence of the lower and upper termination of characters, only the median areas of the words are used to segment characters. Figure 6(b) shows the computation of such quantities.

For each computed distance between characters (denoted as $\mathcal{D}_j^{(C)}$) and words (denoted as $\mathcal{D}_j^{(W)}$), GRAPHJ calculates the following ratios:

$$R_j^{(C)} = \frac{\mathcal{D}_j^{(C)}}{\overline{w}} \qquad R_j^{(W)} = \frac{\mathcal{D}_j^{(W)}}{\overline{w}}, \tag{1}$$

where $\overline{w}$ is the average width of oval characters.

4 Experimental Analysis and Results

In the following sections, we first discuss the complexity of the algorithms in Sec. 4.1, then report experiments to assess the performance of GRAPHJ as compared to standard manual examination techniques in Sec. 4.2. In Sec. 4.3, we report a study on the effectiveness of the relative placement of the superscript dots of lowercase, handwritten "i" letters as a parameter to identify the writer. The proposed analysis illustrates a real case use of GRAPHJ as a forensic tool.

4.1 *Time Complexity*

In this section, we discuss the computational complexity of the algorithms presented in the previous sections. All complexities are reported using the O-notation and considering worst case scenarios. We denote the input binary image as B, its width as w, and its height as h.

The complexity of Algorithm 1 is $O(h^2) + O(w \cdot h)$. This is assessed considering the following complexities. The complexity of function *histRows* (line 3) is $O(w \cdot h)$. The complexity of the external for loop (lines 5 to 16) is $O(\beta)$, where β is the number of peaks in the histogram. β is in the order of h in the worst case scenario, therefore the complexity of the for loop is $O(h)$. Function *findPeaks* scans all peaks in the histogram H_r, hence its complexity is $O(h)$ in the worst case scenario. The complexities of the internal for loops (lines 8 and 9 and lines 10 and 11) are both $O(h)$.

The complexity of Algorithm 2 is $O(h^2)$. Specifically, the complexity of the external for loop (lines 4 to 16) is $O(\beta)$, hence $O(n)$ in the worst case scenario. The complexities of the for loops (lines 5–6 and lines 7–8) are $O(h)$.

The complexity of Algorithm 3 is $O(w \cdot h_L)$, where h_L is the height of the input text-line image patch L. Please note that usually $h_L \ll h$. The complexity of Algorithm 3 is obtained by considering that of function *histCols* (line 4) has complexity $O(w \cdot h_L)$ in the worst case and the complexity of the while loop (lines 7 to 16) is $O(w)$ in the worst case.

The complexity of Algorithm 4 is $O(\Delta \cdot w_w \cdot h_w)$, where Δ is the number of word patches received as input, and w_w and h_w are the largest width and height of input image patches, respectively. The complexity of the external for loops (lines 3 to 17) is $O(\Delta)$. The complexity of the internal for loop (line 6) is constant because α, N, k are constants in our implementation. The complexities of functions *rotate* and *histRows* are both $O(w_w \cdot h_w)$. In line 15, Algorithms 1 and 2 are applied. In the worst case, the complexity of such operations is $O(h_w)$ since in Algorithm 1 the *findPeaks* function will find only one peak (hence there will be a single iteration of the external for loop) and, similarly, Algorithm 2 performs a single iteration of the external for loop.

The complexity of Algorithm 5 is $O(w \cdot h \cdot p \cdot q)$, where p and q are the dimensions of template T.

It should be noted that in practice the overall time needed to perform automatic operations on real documents is in the order of seconds and hence acceptable for the domain experts.

4.2 *Assessment of the Performance of GRAPHJ*

To assess the performance of GRAPHJ, we compare the results obtained with an automated analysis of a handwritten text, with the results of a classic analysis carried out manually by an expert examiner. The analysis was carried out on 10 different writing samples, written voluntarily by 10 different right-handed subjects. All documents have been written in cursive writing and using similar ink and paper. Each subject has been asked to write the same long paragraph of text under dictation. The dictated text included all letters of the Italian alphabet, as well as sentences with different lengths and complexity.

Each sample has been manually analyzed by a forensics expert of RIS, who measured the heights of two groups of 40 different letters. The letters have been analyzed in a sequential way with a degree of precision of 0.1 mm. The first group comprises measurements of the heights (U) of letters with an upper elongated stroke on the right or left side, i.e., "l", "t", "d", "f", "t", ... The second group comprises the measurements of the body in the median zone (M) of letters without elongated strokes, i.e., "a", "c", "o", "m", ...

The same analysis has been carried out using the tools provided by GRAPHJ such as: detection of text lines, detection of words, search of characters, and measurements of quantities. Table 2 reports the mean μ and standard deviation σ for the two groups of letters. The table compares the

Table 2 Mean and standard deviation of the heights of the two groups of letters analyzed by experts examiners using classic manual protocols and using GRAPHJ. All reported measures are expressed in millimeters.

Sample	Analysis	μ_M	σ_M	μ_U	σ_U
1	Manual	1.99	±0.59	4.23	±0.58
1	GRAPHJ	1.91	±0.40	4.51	±0.71
2	Manual	2.00	±0.40	5.26	±0.50
2	GRAPHJ	1.81	±0.39	5.18	±0.60
3	Manual	2.15	±0.55	4.87	±0.76
3	GRAPHJ	2.13	±0.39	4.82	±0.81
4	Manual	1.81	±0.53	6.66	±0.94
4	GRAPHJ	1.84	±0.45	6.48	±0.84
5	Manual	2.03	±0.32	4.65	±0.86
5	GRAPHJ	2.04	±0.41	4.32	±0.78
6	Manual	2.14	±0.47	4.96	±0.89
6	GRAPHJ	2.05	±0.37	5.22	±1.08
7	Manual	1.70	±0.55	4.84	±0.90
7	GRAPHJ	1.58	±0.34	4.37	±0.59
8	Manual	2.47	±0.84	6.17	±1.38
8	GRAPHJ	2.24	±0.55	5.66	±1.03
9	Manual	2.09	±0.59	5.85	±0.82
9	GRAPHJ	1.97	±0.36	5.87	±0.79
10	Manual	2.38	±0.43	7.16	±0.69
10	GRAPHJ	2.19	±0.37	7.21	±0.51

measurements performed by the forensics expert to those obtained using GRAPHJ on the 10 documents considered for the analysis. Table 3 reports the mean absolute percentage error of the measurements obtained using GRAPHJ as compared to the reference measurements obtained by the examiner. The small error reported in the results highlights the compliance of the analysis carried out with GRAPHJ with respect to the standard analysis obtained using classic techniques. Moreover, GRAPHJ allows one to automatically generate a report of the different operations performed by the examiner, thus improving the repeatability of the process.

4.3 *On the Relative Placement of the Superscript Dot of "i" Characters as a Parameter to Infer the Identity of the Writer*

In this section, we investigate whether the relative placement of the superscript dot on "i" characters can be used as a parameter to infer the identity of the writer. The analysis is carried out using GRAPHJ and shows its potential as a forensic examination tool. The proposed study has been carried out analyzing 120 different writing samples. The manuscript samples have been written in two separate sessions by 60 subjects. 79% of the subjects are males, while the remaining 21% are females. Subjects are aged between 20 and 29. 95% of the subjects have a high school diploma, 4% of them have a graduate or undergraduate degree, while the remaining 1% have a middle school diploma. 92% of the subjects are righthanded, the remaining 8% are lefthanded. Each subject produced under dictation two manuscript samples in two different sessions. In the first session, the subject was asked to write on a blank A4 sheet without any marks, while in the second session, manuscripts were produced on ruled A4 sheets. The distance between lines was about 8 mm. The second writing session took place at least 5 h after the first one. We refer to the manuscripts produced in the first session as "sheet 1," while we refer to manuscripts produced in the second session as "sheet 7."

Each of the 60 handwritten documents has been analyzed using GRAPHJ in order to extract, for 20 different lowercase "i" characters the ratio between the height of the "i" without superscript dot (denoted as h_1) and the height of the "i" including the superscript dot (denoted as h). The ratio is hence computed as

$$r = \frac{h}{h_1}. \tag{2}$$

The computation of the h and h_1 values is shown in Fig. 7. The total number of r values measured from the collected manuscript amounts to 1200 instances (20 measurements for each of the 60 manuscripts).

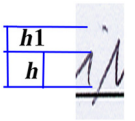

Fig. 7 Measures of the height of an "i" character with (h) and without ($h1$) the superscript dot.

Table 3 Mean absolute percentage error for the two analyzed groups of letters. We report results for each of the 10 documents in the dataset and with respect to the two investigated groups of letters (M and U).

Sample	1	2	3	4	5	6	7	8	9	10
Err% (M)	4.0	9.5	0.9	1.7	0.5	4.2	7.1	9.3	5.7	8.0
Err% (U)	6.6	1.5	1.0	2.7	7.1	5.2	9.7	8.3	0.3	0.7

Table 4 Mean and standard deviations of the r values measured from the "sheet 1" and "sheet 7" manuscripts. All values are measured in millimeters.

	μ	σ
r_1	0.46	0.08
r_7	0.45	0.08

4.3.1 *Consistence of the measurements across different writing sessions*

Table 4 reports the mean μ and standard deviation σ for the samples of r values measured from the "sheet 1" and "sheet 7" manuscripts. These values are referred to as r_1 and r_7 respectively. As can be noted, the mean and standard deviation values for the two different groups ("sheet 1" and "sheet 7") are almost equal. This suggests that the measurement is consistent across different writing sessions and conditions (i.e., using blank or ruled sheets). To verify this observation, we performed a t-student test on the two samples and obtained a t-statistic equal to $t_r = 0.8$, which is below the critical value for a p-value $p < 0.05$ (i.e., 1.65).

To further assess consistence across different writing sessions, for each writer i, we measured the following value:

$$\varepsilon_i = \frac{\overline{r}_1^i - \overline{r}_7^i}{\overline{r}_1^i}, \tag{3}$$

where $\overline{r}_1^i$ is the mean r value measured on "sheet 1" for writer i and $\overline{r}_7^i$ is the mean r value measured on "sheet 7" for writer i. The sample of the computed ε_i values presents a low mean equal to 0.02 mm and a low standard deviation of 0.1 mm, which confirms the consistence of writing across different sessions.

4.3.2 *Use of the measurements to assess the identity of the writer*

In this section, we investigate whether the measured r values can be used to assess the identity of the writer. The problem is usually framed as follows: given two manuscript samples, one from a suspected writer S and the other one from an anonymous writer A, establish if the documents have been written by the same writer (SW) (i.e., if S and A are the same person). In the forensic practice, this is generally assessed by means of the likelihood ratio (LR),[33] which is measured as

$$\mathrm{LR} = \frac{P(E|H_p, X)}{P(E|H_d, X)}, \tag{4}$$

where E is the new evidence (i.e., a manuscript of known origin), X is a reference population (i.e., the population of all manuscripts), H_p is the hypothesis that the two documents have been written by the SW, and H_d is the hypothesis that the two documents have been written by DW. A value of LR larger than one suggests that the documents have been written by the SW. A value of LR smaller than one suggests that the two documents have been written by DW. A value equal to one denotes the neutrality of the tests, suggesting that no conclusion can be drawn from the measurements.

Assuming that the measured r values follow a Gaussian distribution, we quantify the LR value as follows:

$$\mathrm{LR} \approxeq \frac{p_s(\mu_a)}{p_p(\mu_a)}, \tag{5}$$

where

$$p_p(x) = \frac{1}{\sqrt{2\pi\sigma_p^2}} e^{-\frac{(x-\mu_p)^2}{2\sigma_p^2}} \tag{6}$$

is the Gaussian distribution of the r values across all manuscripts (μ_p and σ_p are its mean and standard deviation)

$$p_s(x) = \frac{1}{\sqrt{2\pi\sigma_s^2}} e^{-\frac{(x-\mu_s)^2}{2\sigma_s^2}} \tag{7}$$

is the Gaussian distribution of r values measured across the manuscript of the suspected writer S (μ_s and σ_s are its mean and standard deviation), and μ_a is the mean r value measured from the anonymous writer A. In order to compute the LR value, in all our tests we set μ_p and σ_p to the mean and variance of all samples (i.e., $\mu_p = 0.46$ and $\sigma_p = 0.08$), μ_s and σ_s to the mean and variance of the given suspected writer, computed from the manuscript of "sheet 1," and μ_a to the mean of the measurements obtained from the anonymous author, computed from the manuscript of "sheet 7."

To evaluate the discriminative power of the LR, we considered two sets of pairs containing respectively documents from the SW and documents from DW. Figure 8 shows the Tippet plot (the inverse of the cumulative distribution functions) with the graphical representations of the PMEH_p and PMEH_p probabilities. PMEH_p is the probability that LR values are smaller than 1 knowing that the writers are the same, while PMEH_s is the probability that LR values are larger than 1 knowing that the writers are different. Ideally, these two probabilities should be equal to 0, while in our case we have $\mathrm{PMEH}_p \approx 0.4$ and $\mathrm{PMEH}_d \approx 0.3$.

The proposed analysis suggests the following findings: (1) the measured values are consistent across different writing sessions and conditions (i.e., blank or ruled sheets); (2) in accordance with previous results,[34] the ratio between the height of "i" characters and the positions of the superscript dots is not reliable to assess the identity of the writer.

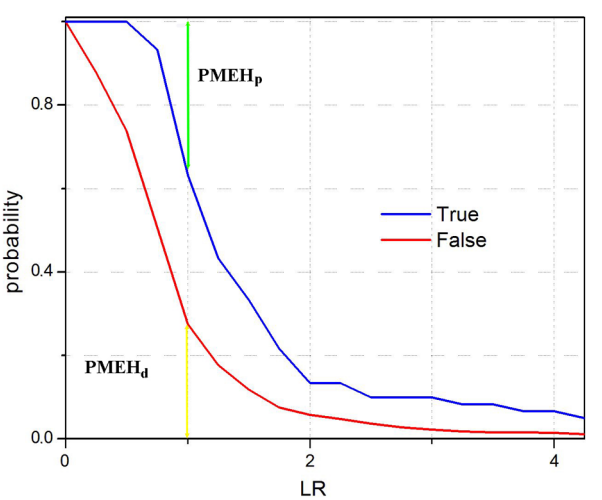

Fig. 8 Tippet plot.

5 Discussion and Conclusion

This work has presented GRAPHJ, a forensic tool for the analysis of handwritten documents that follows the real protocol adopted by RIS of Carabinieri in Italy. The tool allows one to automate many operations including the detection of elements such as text lines and words, the search of characters, and the measurement of quantities such as the heights and widths of characters. Our experimental analysis shows that: (1) measures retrieved with GRAPHJ are compatible with those obtained through a classic analysis by forensics experts, and (2) GRAPHJ can be effectively used to perform the analysis of handwritten documents. The main limit of GRAPHJ is that algorithms are semiautomatic, which, at the moment, requires few manual interventions of the operator to obtain the expected results. Future works will focus on a further automation of the selection thresholds and on the use of GRAPHJ as a tool to obtain a labeled dataset useful to train and test machine learning algorithms, which can be useful to improve the overall framework in order to perform an automatic analysis.

References

1. J. Orliaguet, S. Kandel, and L. Bo, "Visual perception of motor anticipation in cursive handwriting: influence of spatial and movement information on the prediction of forthcoming letters," *Perception* **26**(7), 905–912 (1997).
2. R. A. Huber and A. M. Headrick, *Handwriting Identification: Facts and Fundamentals*, CRC Press, Boca Raton, Florida (1999).
3. R. C. Hayes, *Forensic Handwriting Examination: A Definitive Guide*, ReedWrite Press, Honolulu, Hawaii (2006).
4. L. Schomaker, "Writer identification and verification," in *Advances in Biometrics*, N. K. Ratha and V. Govindaraju, Eds., pp. 247–264, Springer, London (2008).
5. I. W. Evett and R. N. Totty, "A study of the variation in the dimensions of genuine signatures," *J. Forensic Sci. Soc.* **25**(3), 207–215 (1985).
6. S. E. Abbey, "Natural variation and relative height proportions," *Int. J. Forensic Doc. Exam.* **5**, 108–116 (1999).
7. J. Maciaszek, "Natural variation in measurable features of initials," *Prob. Forensic Sci.* **85**, 25–39 (2011).
8. B. Found and D. Rogers, "The forensic investigation of signature complexity," in *Handwriting and Drawing Research: Basic and Applied Issues*, M. L. Simner, C. G. Leedham, and A. J. W. M. Thomassen, Eds., pp. 483–492, IOS Press, Amsterdam (1996).
9. B. Found and D. Rogers, "A consideration of the theoretical basis of forensic handwriting examination," *Int. J. Forensic Doc. Exam.* **4**, 109–118 (1998).
10. K. M. Koppenhaver, *Forensic Document Examination: Principles and Practice*, Springer Science and Business Media, Heidelberg (2007).
11. R. Morris, *Forensic Handwriting Identification: Fundamental Concepts and Principles*, Academic Press, San Diego (2000).
12. J. S. Kelly and B. S. Lindblom, *Scientific Examination of Questioned Documents*, CRC Press, Boca Raton, Florida (2006).
13. L. Schomaker, "Advances in writer identification and verification," in *Ninth Int. Conf. on Document Analysis and Recognition (ICDAR)*, Vol. 2, pp. 1268–1273, IEEE (2007).
14. A. Brink, L. Schomaker, and M. Bulacu, "Towards explainable writer verification and identification using vantage writers," in *Ninth Int. Conf. on Document Analysis and Recognition (ICDAR)*, Vol. 2, pp. 824–828, IEEE (2007).
15. A. Schlapbach, M. Liwicki, and H. Bunke, "A writer identification system for on-line whiteboard data," *Pattern Recognit.* **41**(7), 2381–2397 (2008).
16. M. Bulacu and L. Schomaker, "Text-independent writer identification and verification using textural and allographic features," *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(4), 701–717 (2007).
17. S. M. Saad, "Application of fuzzy logic and genetic algorithm in biometric text-independent writer identification," *IET Inf. Secur.* **5**(1), 1–9 (2011).
18. A. Shivram, C. Ramaiah, and V. Govindaraju, "A hierarchical bayesian approach to online writer identification," *IET Biom.* **2**(4), 191–198 (2013).
19. A. Chahi et al., "Block wise local binary count for off-line text-independent writer identification," *Expert Syst. Appl.* **93**, 1–14 (2018).
20. S. He and L. Schomaker, "Writer identification using curvature-free features," *Pattern Recognit.* **63**, 451–464 (2017).
21. V. Christlein et al., "Writer identification using gmm supervectors and exemplar-svms," *Pattern Recognit.* **63**, 258–267 (2017).
22. Y. Hannad, I. Siddiqi, and M. E. Y. El Kettani, "Writer identification using texture descriptors of handwritten fragments," *Expert Syst. Appl.* **47**, 14–22 (2016).
23. A. Bhardwaj et al., "Retrieving handwriting styles: a content based approach to handwritten document retrieval," in *Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pp. 265–270, IEEE (2010).
24. C. Ramaiah, G. Kumar, and V. Govindaraju, "Handwritten document age classification based on handwriting styles," *Proc. SPIE* **8297**, 82970Q (2012).
25. E. Fabiańska et al., "Graphlog-computer system supporting handwriting analysis," *Prob. Forensic Sci.* **68**, 394–408 (2006).
26. NITe. Masquerade, http://www.nitesrl.com/products/masquerade/ (12 June 2018).
27. S. Battiato, O. Giudice, and A. Paratore, "Multimedia forensics: discovering the history of multimedia contents," in *Proc. of the 17th Int. Conf. on Computer Systems and Technologies*, pp. 5–16, ACM (2016).
28. A. Marcelli, A. Parziale, and C. De Stefano, "Quantitative evaluation of features for forensic handwriting examination," in *IEEE 13th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 1266–1271 (2015).
29. A. Parziale et al., "An interactive tool for forensic handwriting examination," in *IEEE 14th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pp. 440–445 (2014).
30. M. D. Abràmoff, P. J. Magalhães, and S. J. Ram, "Image processing with ImageJ," *Biophotonics Int.* **11**(7), 36–42 (2004).
31. IPLAB. GRAPHJ, http://iplab.dmi.unict.it/GRAPHJ/ (13 July 2018).
32. S. Battiato et al., "Adaptive techniques for microarray image analysis with related quality assessment," *J. Electron. Imaging* **16**(4), 043013 (2007).
33. D. Lucy, *Introduction to Statistics for Forensic Scientists*, John Wiley and Sons, Chichester, England (2013).
34. V. Matranga, "La persistenza di gestualità complesse nella dissimulazione grafica," "Scrittura," *Riv. Probl. Grafologici* **172**, 77–95 (2016).

Luca Guarnera is a PhD student in computer science at the University of Catania in collaboration with the spin-off ICTLab. He received his master's degree in computer science (cum laude) from the University of Catania in 2018. He joined IPLAB in 2015. His research interests are multimedia forensics and related fields.

Giovanni Maria Farinella received his PhD in computer science in 2008. He is a tenure track associate professor at the Department of Mathematics and Computer Science, University of Catania, Italy. His research interests lie in the fields of computer vision, pattern recognition, and machine learning. He founded (in 2006) and currently directs the International Computer Vision Summer School. He was awarded the PAMI Mark Everingham Prize 2017.

Antonino Furnari is a postdoctoral fellow at the University of Catania. He received his PhD in mathematics and computer science in 2016 from the University of Catania. He is author or coauthor of 17 papers in international book chapters, international journals, and international conference proceedings. His research interests are computer vision, pattern recognition, and machine learning, with focus on first person vision.

Angelo Salici received his Italian master's degree in electronic engineering from the University of Catania in 2005 and his PhD in advanced technologies for information engineering from the University of Messina in 2012. He is a forensic expert and laboratory director of the Carabinieri Force and his current research interests are in the areas of ballistic, biometrics, and digital forensic.

Claudio Ciampini received his Italian master's degree in physics from the University of Pisa. He is a forensic expert and laboratory director of the Carabinieri Force. From 2002 to 2014, he worked in the field of face and speaker recognition. His current research interests are in the areas of ballistic and biometric recognition.

Vito Matranga received his Italian degree in law from the University of Palermo and his second-level university master's degree in forensic science from the University of Messina. He is a senior forensic document examiner of the Carabinieri Force and deals with forensic handwriting analysis.

Sebastiano Battiato is a full professor of computer science at the University of Catania. He is involved in research and directorship of the IPLab research lab. His research interests include computer vision, imaging technology and multimedia forensics. He is also director (and cofounder) of the International Computer Vision Summer School, Sicily, Italy, and a senior member of the IEEE.