Tools for Historical Handwritten Document Analysis

Ioannis Pratikakis[0000-0002-4124-3688]

Democritus University of Thrace, Department of Electrical and Computer Engineering, Kimmeria Campus, 67100 Xanthi, Greece
ipratika@ee.duth.gr

Abstract. Accessing historical documents requires an effective toolbox that comprises several technologies within the area of document image analysis. In this paper, we focus on Handwritten Text recognition (HTR) and Keyword spotting (KS) for which we present representative approaches that aim for effectiveness and efficiency.

Keywords: Handwritten Text Recognition, Keyword Spotting, Historical Handwritten Documents.

1 Introduction

The potential to access our written past, which stimulates the interest of not only researchers but also the general public, makes relevant technologies from document image analysis research area highly appealing. In this paper, the focus is on tools for effective historical handwritten document analysis, namely Handwritten Text recognition (HTR) and Keyword spotting (KS).

Several challenges are present for those tools targeting the relevant historical period, caused by the age of the historical manuscripts that affects the clarity of the writing and the image quality in general. The language used in the writing results in increased complexity due to the multitude of diacritics, punctuation and abbreviating symbols that were used, leading to an increased character set compared to modern languages. The complexity is further increased by the fact that the content of such documents is unconstrained and might have been created by multiple writers.

The structure of this paper is built upon the description of distinct methodologies that relate to distinct tools for HTR and KS that are situated in a toolbox dedicated to Handwritten Document Analysis. Example toolbox is shown in Figure 1.

2 Handwritten Text Recognition

The overall proposed architecture consists of an image preprocessing module that feeds an OctCNN-BGRU. The proposed architecture, as shown in Figure 2, consists of a

CNN stage for feature extraction and a recurrent stage for feature decoding into a probability vector corresponding to the different character classes. Each text line of the document is presegmented and processed separately, in a bidirectional manner. Each of the stages is presented in detail in the following sections. For a comprehensive presentation the interested readers can consult the work in (Tsochatzidis et al., 2021).

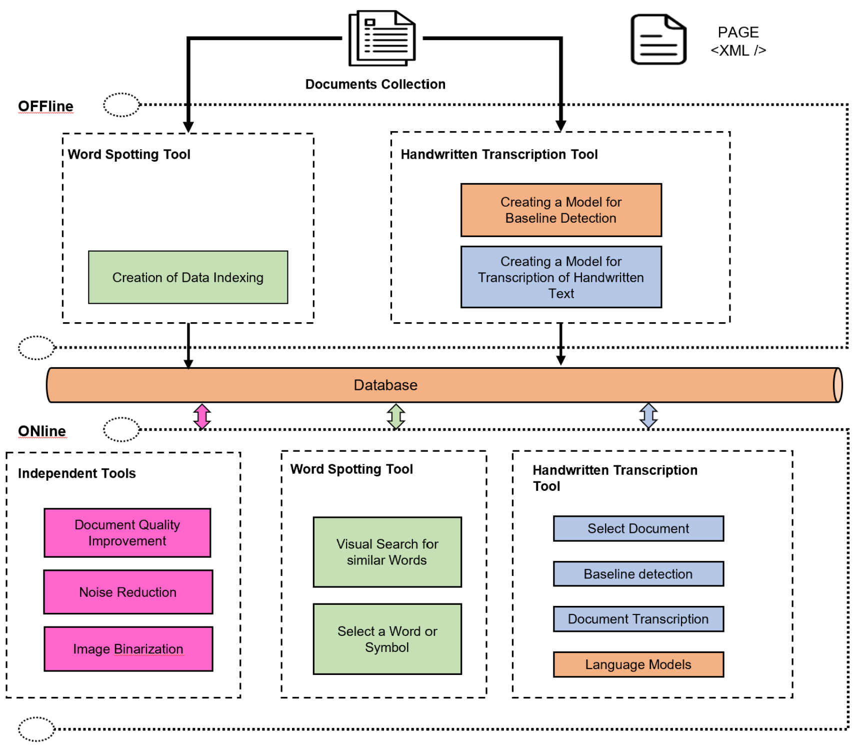
Documents Collection
PAGE
<XML />
OFFline
Word Spotting Tool
Creation of Data Indexing
Handwritten Transcription Tool
Creating a Model for Baseline Detection
Creating a Model for Transcription of Handwritten Text
Database
ONline
Independent Tools
Document Quality Improvement
Noise Reduction
Image Binarization
Word Spotting Tool
Visual Search for similar Words
Select a Word or Symbol
Handwritten Transcription Tool
Select Document
Baseline detection
Document Transcription
Language Models

Fig. 1. The offline and online component of a toolbox dedicated to Handwritten Document Analysis

2.1 Preprocessing

The preprocessing of the input images utilized in the proposed pipeline aims to standardize images from different sources and writers. Towards this end, Illumination Compensation was used to remove shadows and balance brightness/contrast along a text line image. As a next step, deslanting is applied, to soften the cursive style that may occur during handwriting, affecting the slope of the line and the slant of the letters.

2.2 Octave-CNN architecture

The octave-convolution operation (OctConv), introduced in (Chen et al., 2019), is a drop-in replacement for the convolution operation in a CNN architecture, which involves processing the input in two different scales, aiming to capture both high- and low-frequency patterns. Towards this end, the input feature map X is factorized into two portions along the channel axis, resulting in two feature maps that capture fine- and low-detailed information.

The proposed Octave-CNN architecture, as shown in Figure 2, is aimed at the extraction of features from the input image in a feed-forward manner. It consists of five convolutional blocks, each one containing an OctConv layer with kernel size 3 × 3 pixels, stride equal to 1 and batch normalization. The leaky rectified linear (LeakyReLU) function is used for neuron activation, which provides a small gradient value when the unit is not active. A maximum pooling layer with kernel size equal to 2 × 2 is used after the first three blocks, to reduce the spatial dimensions of the features. Additionally, a dropout layer, with probability equal to 0.2 (experimentally defined) is included in the last three blocks, to assist for better generalization ability and robustness of the features. Finally, the average of each column of the feature maps of the last layer is calculated, to acquire a feature vector with 80 features for each time step along the width of the image, as shown in Figure 3.

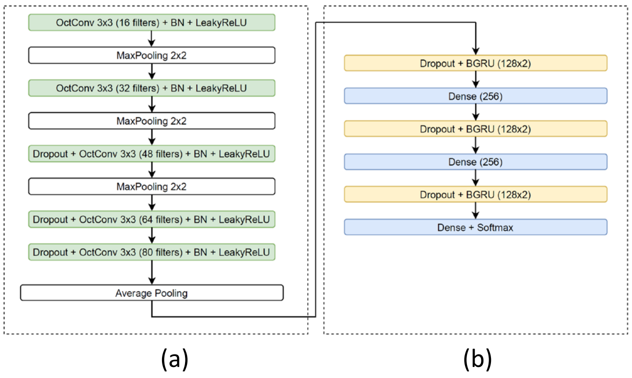
OctConv 3x3 (16 filters) + BN + LeakyReLU
MaxPooling 2x2
OctConv 3x3 (32 filters) + BN + LeakyReLU
MaxPooling 2x2
Dropout + OctConv 3x3 (48 filters) + BN + LeakyReLU
MaxPooling 2x2
Dropout + OctConv 3x3 (64 filters) + BN + LeakyReLU
Dropout + OctConv 3x3 (80 filters) + BN + LeakyReLU
Average Pooling
Dropout + BGRU (128x2)
Dense (256)
Dropout + BGRU (128x2)
Dense (256)
Dropout + BGRU (128x2)
Dense + Softmax
(a)
(b)

Fig. 2. Schematic diagram of the proposed HTR architecture, consisting of (a) the CNN stage, and (b) the recurrent stage.

3 Keyword Spotting

Figure 4 shows the proposed methodology. First, the DoLF local features are calculated and detected on a preprocessing document image. Afterwards, an indexing procedure uses the calculated DoLF and creates a set of data structures which allow efficient and effective word spotting. Next, the user selects the query word image on which the DoLF local features are calculated. Finally, a novel matching procedure called Quantitative Near Neighborhood Search (QNNS) detects on documents visually similar regions to this query word image and presents them to the user. For a comprehensive presentation the interested readers can consult the work in (Zagoris et al., 2021).

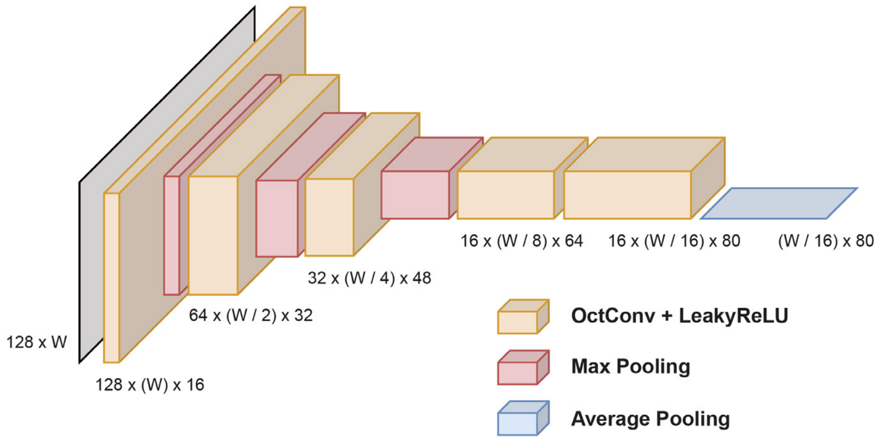

Fig. 3. Feature maps produced by each layer of the Octave-CNN model.

3.1 Preprocessing and Local Points Calculation

The initial step contains a pre-processing step that aims to enhance the contrast between the foreground and background pixels of the document image and keep the valuable edge boundary information. This enhances the effectiveness of the subsequent calculation of the gradient-based keypoints and descriptors. The required pre-processing step comprises two steps. The first step consists of a contrast normalization method which deals with the illumination changes. Subsequently, the gradient vectors I_x and I_y of the document image I are calculated and filtered by a high pass filter for eliminating any remaining background noise. The filter thresholds are calculated dynamically based on the Otsu algorithm for minimizing the intra-class variance between the foreground and background pixel clusters. The two filtered gradient images, as shown in Figure 5(b) and Figure 5(c), are then encoded in a JavaScript Object Notation (JSON) format and passed to the server application for the keypoint extraction.

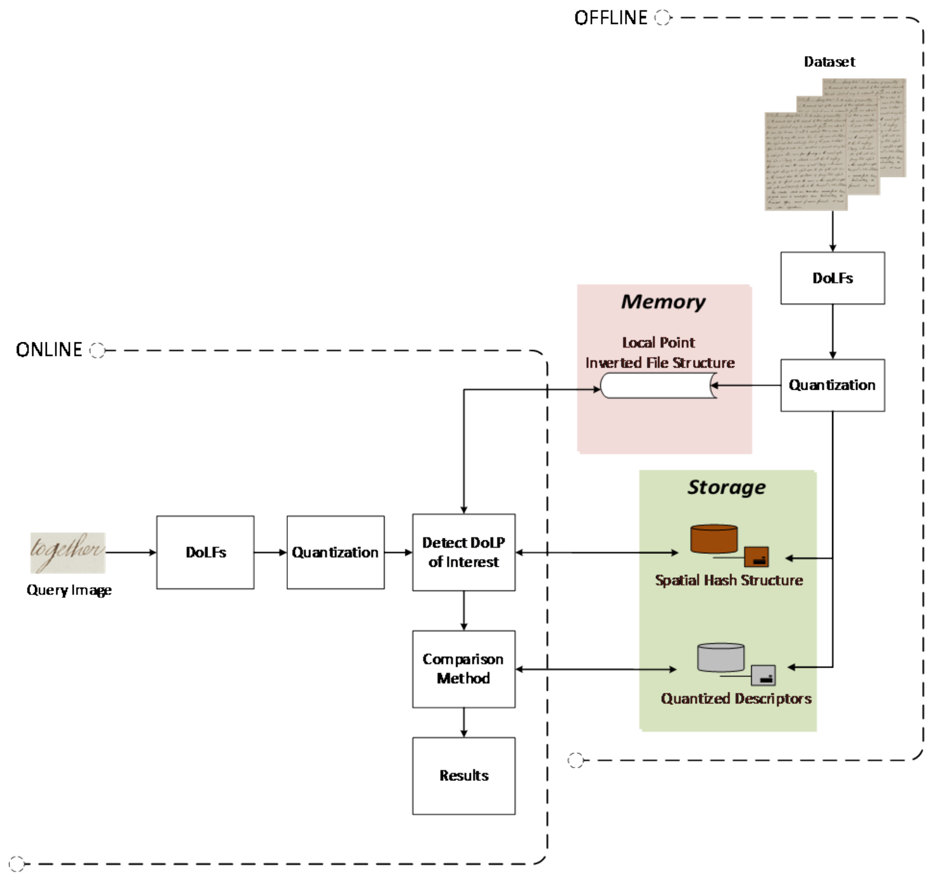

Fig. 4. An efficient keyword spotting architecture

For the keypoint and feature extraction, we adapted a version of the Document-oriented Local Features (DoLF) proposed in (Zagoris et. al., 2017). While most works that are using local features are based on the Scale Invariant Feature Transform (SIFT) (Lowe, 2004) initially motivated as being used for natural images, in comparison with the document images, have many structural differences that create problems such as (i) the erroneous local points detection between the document lines due to the image pyramid scaling (ii) the invariant properties of those descriptors which amplify noise.

On the contrary, the DoLF exhibits some desirable characteristics which makes them suitable for the proposed method, such as (i) they take into consideration the handwritten document particularities; (ii) they provide consistency between different handwritten writing variations; (iii) their descriptors contain texture information in a spatial context which is suitable in dealing with a document collection created by different writers, containing significant writing style variations.

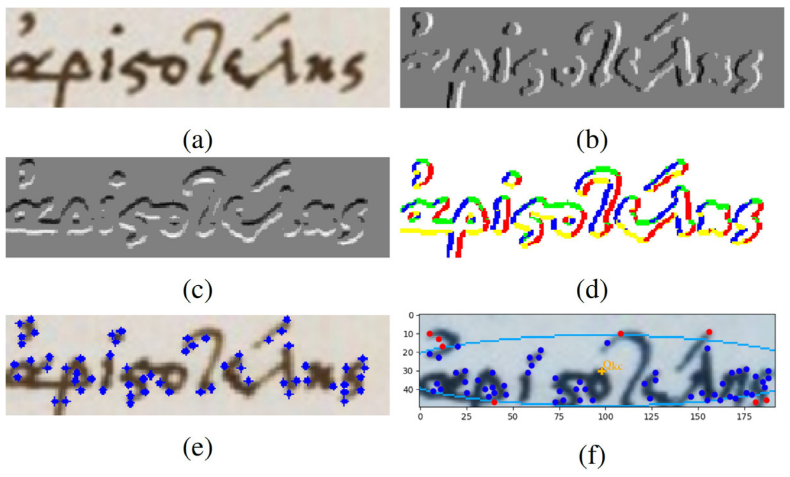

Fig. 5. The proposed word spotting indexing and matching pipeline for the Greek handwritten word of 'Aristotle'. (a) Query image; (b) Filtered gradient image Ix; (c) Filtered gradient image Iy; (d) Quantization of the gradient orientation; (e) Keypoints; (f) Query keypoint (yellow) and the included keypoints (blue) to be matched during the matching process.

The DoLF computation comprises two parts, the keypoint detection and the feature calculation around it. Subsequently, a brief description of the keypoints and their descriptors is provided.

The next step involves the linear quantization of the gradient orientation. This step aims to label the changes to writing direction as these points consist of important and descriptive information. Figure 5(d) shows the output for the quantization of the gradient orientation values. Each colour represents a different quantization level. The current work uses four different quantization levels. Next, for each quantization level, the Connected Components (CC) are detected.

These CCs represent chunks of strokes that correspond to different writing directions. The final local points are the centre of gravity of each remaining CC. An example of these keypoints is shown in Figure 5(e). The keypoint calculation method can detect meaningful points of the characters that reside in the documents independently of its scale. Moreover, it provides some consistency between different handwritten writing variations.

The next step involves the calculation of the features around the detected keypoints. The descriptor is calculated upon a scale-invariance window size around the detected local points. The window size is defined dynamically by calculating the mean brightness in different window sizes and selecting the one that has the maximum value.
Then, the selected window size is divided into 16 cells where a 4-bin histogram is calculated, representing each cell (each bin corresponds to a quantization level). Each pixel inside a cell accumulates a vote in the corresponding angle histogram bin. The strength of voting depends on the pixel's magnitude of the gradient vector.

All histograms are concatenated in a single 64-bin histogram and normalized by its norm. Finally, all values above 0.2 are set to 0.2 and are re-normalized again to minimize the illumination effect in the descriptor. The final descriptor is shown in Figure 6(c).

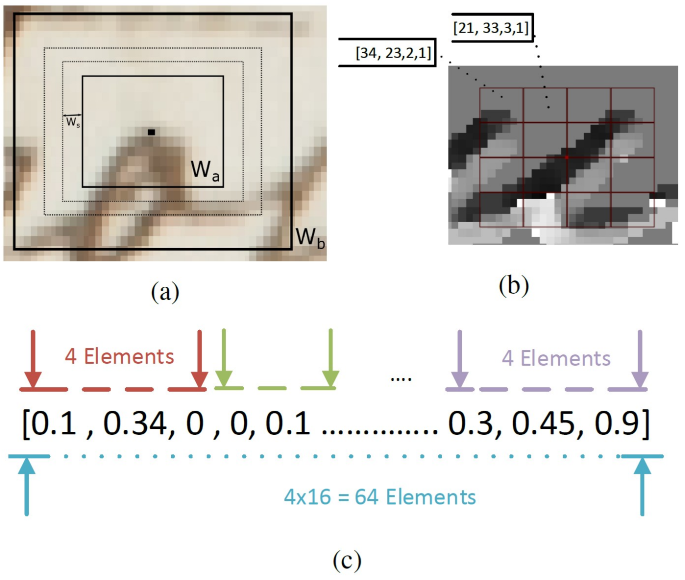

Fig. 6. Feature extraction: (a) Window size detection; (b) Features calculation; (c) Descriptor structure.

3.2 Features Indexing

The sheer magnitude of the detected local points and their descriptors are increasing the costs in terms of time, memory and storage requirements rendering them unusable to provide a service. To solve this issue, we transform the information from the DoLFs to several different structures that permit efficient word spotting without compromising on the effectiveness.

The proposed indexing method comprises two distinct steps:

A. Descriptor quantization using multiple Bag of Visual Words (BoVW) to decompose and compress its information

B. Three different memory and storage structures for very quickly segmentation-free word spotting during the client request.

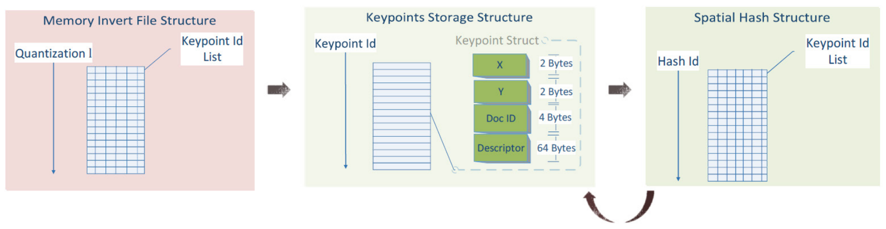

Fig. 7. Memory and storage structures along with their relationships.

At this point, the descriptor is split to four different parts. This is because each part describes a different spatial part of the descriptors. Next, four different codebooks are created for the different parts of the descriptor. The size of each codebook corresponds to the descriptor precision but with increasing processing and storage costs. For our experiments, a codebook of size 16 is used. The final quantized descriptor is a 4-bin histogram with each bin in the range of 0-15 values. Then, three data structures are created to facilitate the low retrieval times, incorporating the information from the local points location and the corresponding quantized descriptor. The Memory Invert File Structure, the Descriptors Storage Structure and the Spatial Hash Structure as shown at Figure 7.

The Keypoints Storage Structure (KSS) is the simplest and most prominent data structure. It contains all the calculated keypoints and their descriptors. It is a struct array in which the array index corresponds to the keypoints Id. Each struct (72 bytes) holds the keypoint location (X, Y), the document which resides (DocID) and its quantized descriptor.

The Memory Invert File Structure (MIFS) resides in memory. It is an array of 64-bit integer lists (8 bytes). Each descriptor corresponds to a hash id *l* which is calculated from the following hash function:

$$l = BOW_1 * K^3 + BOW_2 * K^2 + BOW_3 * K + BOW_4 \qquad (1)$$

where *BOWi* corresponds to the each quantitized value of descriptors (Figure 7(a)) and *K* equals to the codebook size. In our proposed method K = 16.

The MIFS is a dictionary-based structure that maps a list of keypoints K with the same hash value l. Thus, the MIFS primary function is to retrieve a list of descriptors D, which have the same hash value *l*.

The Spatial Hash Structure (SHS) assists the detection of every keypoint that is residing in a specific location inside a document as it provides information about the spatial distribution of the local points inside a document. Its primary function is to retrieve a list of descriptors D that their local points location is closed to each other inside the document.

This is achieved by using the following hash function:

$$h(x, y, n) = n * A^2 + y * A + x \qquad (2)$$

where *A* is any custom-defined number that is greater than the maximum width and height document inside a collection.

The above hash function takes as input the location of the keypoint (x, y) and the document n resulting in the hashID computation of the corresponding local point.

The retrieval of all the DoLFs in proximity to the point (x_1, y_1) is achieved by: First, calculating the hashIds for all the local points residing inside the space (x ± dx, y ± dy) for the document n. The dx and dy denote the distance space from the local point (x_1, y_1). Next, all the descriptor Ids are retrieved from the SHS that corresponds to that specific hashIds. Finally, through the KSS, all the descriptor information is available.

These three storage structs is the minimum required information to be stored from a documents collection. Only the MIFS needs to reside in memory; the other two can be stored in disks and accessed when needed.

3.3 Feature Matching

Feature Matching is the only procedure that affects the user as it commences during the word spotting search. Initially, the query image is analyzed, and the DoLF is calculated. From them, the quantized descriptors and the corresponding hash values *l* are calculated based on the previous trained BoVW. Finally, the hash values *l* are calculated based on the Eq. 1 for each DoLF.

Through the MIFS, all the descriptor ids with the same hash value *l* with the query *l* value are retrieved. Finally, their locations and descriptors are retrieved through KSS. The feature matching goal is to identify those keypoints with similar spatial distri- bu- tion and descriptors with the query keypoints. First, the nearest keypoint Qk_c from (c_x,c_y) point is identified. The (c_x,c_y) is denoted as the mean center of the keypoints set in the query word image. The next step involves identifying the most similar local points with the Qk_c from the retrieved descriptors n.

In our implementation, the Euclidean Distance (ED) is used, and the top N matches that are kept denote those that have the distance from the query keypoint Qk_c feature equal to those that have the distance from the query keypoint Qk_c feature lower than a threshold t. This threshold is experimentally defined and controls the time expense of the search in the document space. Each keypoint that belongs to the top N matches is a document candidate coordinate origin similar to the query image.

The spatial NNS for each keypoint that resides on the query image is addressed in the next stage. The spatial NNS is realized in a search area around each point. During the search, if there are one or more keypoints in the proximity of the query keypoint under consideration, the Euclidean distance between their descriptors is calculated and the minimum distance is kept. The previous procedure is repeated for each keypoint in the query image. The final similarity measure is the average of all the minimum distances. In the case that a local point in its proximity does not exist, then the query local point is ignored.

4 Conclusion

This paper presents a toolbox that comprises effective approaches for Handwritten Text recognition (HTR) and Keyword spotting (KS) that result in effective historical handwritten document analysis.

References

Chen, Y., Fan, H., Xu, B., Yan, Z., Kalantidis, Y., Rohrbach, M., Yan, S., & Feng, J. (2019). Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), October 27–November 2, 2019* (pp. 3434–3443). https://doi.org/10.1109/ICCV.2019.00353

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

Tsochatzidis, L., Symeonidis, S., Papazoglou, A., & Pratikakis, I. (2021). HTR for Greek historical handwritten documents. *Journal of Imaging*, *7*(12), 260. https://doi.org/10.3390/jimaging7120260

Zagoris, K., Pratikakis, I., & Gatos, B. (2017). Unsupervised word spotting in historical handwritten documents using document-oriented local features. *IEEE Transactions on Image Processing*, *26*(8), 4032–4041. https://doi.org/10.1109/TIP.2017.2700721

Zagoris, K., Amanatiadis, A., & Pratikakis, I. (2021). Word spotting as a service: An unsupervised and segmentation-free framework for handwritten documents. *Journal of Imaging*, *7*(12), 278. https://www.mdpi.com/2313-433X/7/12/278

Received: April 27, 2025
Reviewed: May 15, 2025
Finally Accepted: June 01, 2025