To appear in the 2018 11th Int'l Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI 2018), Oct 11-13, 2018, Beijing, China.

Challenges and Preprocessing Recommendations for MADCAT Dataset of Handwritten Arabic Documents

Gheith A. Abandah
Computer Engineering Department
The University of Jordan
Amman 11942, Jordan

Ahmad S. Al-Hourani
Computer Engineering Department
The University of Jordan
Amman 11942, Jordan

***Abstract*—In this paper, we analyze the dataset often used in training and testing Arabic handwritten document recognition systems, the Multilingual Automatic Document Classification Analysis and Translation dataset (MADCAT). We report here the main challenges present in MADCAT that the preprocessing stage of any recognition algorithm faces and affect the performance of the systems that use it for training and testing. MADCAT is a representative dataset of Arabic handwritten documents and investigating its challenges helps to identify the requirements of the preprocessing stage. After presenting these challenges, we review the literature and recommend preprocessing algorithms suitable to preprocess this dataset for handwritten Arabic word recognition systems such as JU-OCR2.**

***Keywords*—MADCAT dataset; preprocessing; recognition; handwritten documents; Arabic**

I. Introduction

There is a need for investigating the problems that one needs to solve in the preprocessing stage of handwritten Arabic document recognition systems. We investigate here the Multilingual Automatic Document Classification Analysis and Translation (MADCAT) dataset [1]. We report the challenges that face the preprocessing stage to clean, prepare, and segment these documents for a handwritten Arabic word recognition system such as JU-OCR2 [2]. We also select from available research efficient preprocessing algorithms and recommend a preprocessing stage suitable for this dataset and such recognition systems. The details of this study is in [3].

The rest of this paper is structured as follows: Section II provides a brief description of MADCAT. Section III discusses the problems found in MADCAT. Section IV discusses additional problems that preprocessing stages may need to handle but do not exist in MADCAT. Section V presents the recommendations from available literature for a preprocessing stage to build high-performance recognition system. Section VI provides the conclusions of this paper.

II. MADCAT Dataset

The MADCAT is a five-year research program created by the Defense Advanced Research Projects Agency (DARPA) that aims to convert non-English document images to English transcripts [1]. The National Institute of Standards and Technology (NIST) conducted the Open Handwriting Recognition and Translation Evaluation (OpenHaRT) competition using MADCAT in order to advance state-of-the-art handwritten Arabic document analysis [4].

The MADCAT dataset was created by the Linguistic Data Consortium (LDC) in order to standardize the training, testing, and evaluation set of corpora used by systems and algorithms used for preprocessing, recognition, and translation of Arabic documents. The MADCAT dataset was created in a controlled environment. The documents were written by native Arabs who are proficient in Arabic [4]. Table I shows the writing conditions which were used when creating MADCAT [5].

TABLE I. MADCAT Writing Conditions

Writing Condition		**Ratio**
Utensil	Pen	90%
	Pencil	10%
Paper	Unlined	75%
	Lined	25%
Speed	Careful	5%
	Normal	90%
	Fast	5%

The original scripts of the dataset were selected from authentic Arabic texts from newspapers and news websites. The dataset specifications are specified in Table II [5].

TABLE II. MADCAT Specifications

Specification	**Training**			**Evaluation**
	Phase 1	**Phase 2**	**Phase 3**	**Phase 3**
Unique pages	2,334	5,773	635	211
Scribes per page	up to 5	up to 5	up to 7	3
Tokens per page	NC	NC	≤ 125	≤ 125
Total pages	9,693	27,814	4,540	633
Total tokens	758,936	2,964,266	507,689	74,553
Unique scribes	72	230	53	24
Image size	5100×6600 pixels at 600 dpi			

III. Challenges Present in MADCAT

In this section, we describe the challenges present in the MADCAT dataset.

A. Pepper Noise

During the image acquisition and binarization phases, noise dots may be produced in the scanned documents. The noise dots concentration varies from one document to another. In some documents, the noise dots are few or not present as in Fig. 1(a). In other documents, noise dots may be densely scattered all over the document where they even could cluster and form larger bodies as in Fig. 1(b).

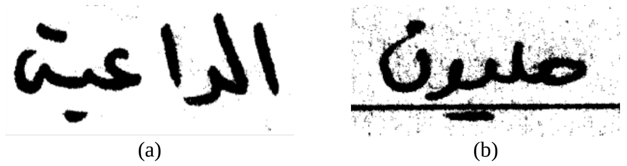


Fig. 1. (a) Sparse noise distribution. (b) Dense noise distributions.

The smallest text elements are usually much larger than the largest continuous noise bodies. This can be exploited to overcome this issue. But care should be taken to account for various text sizes, writing tip widths, and low resolution.

B. Vertical Lines

Vertical lines can be categorized into two categories depending on their source. The first, and the more influential, is the ruled vertical lines; that is, the vertical lines which are imposed by the nature of the paper used. The number of adjacent ruled vertical lines range from one to three lines. Furthermore, ruled vertical lines may intersect with the handwritten text and distort it as in Fig. 2(a).

Fig. 2. (a) Ruled vertical lines affecting text. (b) Noise-generated lines.

The second type includes vertical lines generated during the image acquisition phase. These lines happen in some scanning machines that tend to create faint lines in the acquired images. Another source of the faint lines is the shadow of the edges of the scanned pages. By nature, these lines are faint and mostly discontinuous as in Fig. 2(b).

C. External Graphical Elements

The various paper types available in MADCAT introduce various none textual shapes that need to be filtered prior the recognition phase. Depending on the paper source and image acquisition conditions, sample shapes are shown in Fig. 3.

In case these elements are fed to recognition, they may be erroneously recognized as punctuation marks or separate letters. Also, these shapes can in some cases distort the vertical and horizontal histograms of the documents, as in the first and last examples in Fig. 3. In some extreme cases, these elements may contain higher black dots density than the main text body. This needs to be considered when detecting the main text body.

These shapes are usually on the edge of the page where people do not tend to write text. This creates a significant distance between the main text body and those elements that can be utilized to mitigate the effects of those elements on the segmentation and recognition phases.

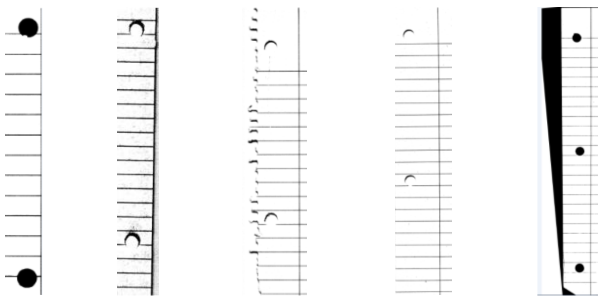


Fig. 3. Various undesired graphical elements.

D. Writing Errors

The ground truth in MADCAT identify the errors made by the scribes when copying the text. These errors should be identified and removed. Some errors can be easily identified as clusters of ink or as shapes with too many strikes while others are harder to find as they only strike out the wrong word with a single line. Writing errors that are not struck out or scribbled will only cause wrong recognition of the word and may only be detected by dictionary word matching or by the context. Fig. 4 shows examples the above mentioned writing errors.

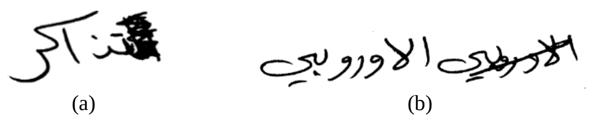


Fig. 4. Eriting errors marked by (a) dark area and (b) one strike

E. Ruled Paper

Ruled paper is common for notebooks. Different ruling types are available like narrow-ruled, college-ruled, law-ruled, and journal. Many ruled paper types have also vertical lines in different positions of the page. Ruled pages usually comply with the following conditions:

- A ruled line, horizontal or vertical, usually extends from one end of the page all the way to the other.
- Horizontal lines are parallel to each other and perpendicular to vertical lines. This is always true unless some distortion occurred to the page in the image acquisition stage.
- If the page is ruled, handwritten text is aligned to the lines and suffers no writer skew, even though text could be on unlined spaces of the page as in Fig. 5.

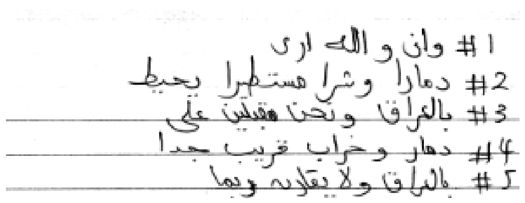


Fig. 5. Text in the unruled area of a page.

- A close-up of ruled lines show that they are not perfect rectangles as shown in Fig. 6, it is a rectangle-like shape with variable widths and many imperfections including complete line breaks at some points. These disfigurements are created by the inconsistencies encountered along the processes of ruled line printing, capturing, and binarization.

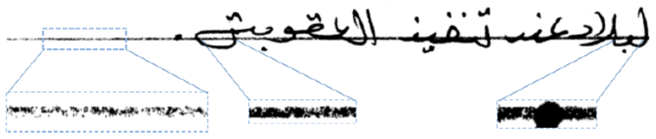


Fig. 6. Close up of a ruled line [6].

The ruled lines are often of same color, thickness and orientation of text elements and thus can significantly overlap with text. This, combined with the characteristic horizontal baseline of Arabic handwriting [7], results that inattentive line removal generates severe deterioration to words, especially for algorithms that depend on text thickness as in Ref. [8]. Such cases are illustrated in the sample of Fig. 7. The effect of ruled line removal is expected to be minimal when ruled line width is considerably thinner than the text line width.

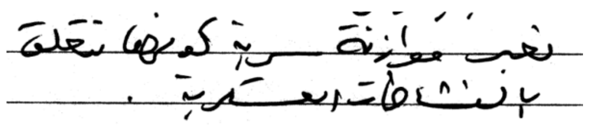


Fig. 7. Sample where the ruled line width is close to the text width and the word baseline is overlaping the ruled lines.

F. Text Lines Overlapping

Irregularity and inconsistency are common traits of handwriting. Such traits create problems that we do not face in printed text [9]. Such problem is text lines overlapping. That is a straight line to separate consecutive text lines cannot be found. This problem would have a major impact on the performance of the recognition unless an efficient and deterministic approach is used to solve it. The effects of lines overlapping include the following:

- Some word elements like strokes and secondary components may be mistaken to be belonging to another text line as in Fig. 8(a).
- Sometimes overlapping produces connected elements between two consecutive lines as in Fig. 8(b).

(a) (b)

Fig. 8. Effects of overlapping text lines.

In terms of inter-line spacing, pages can be divided into three categories:

Distant lines, where all the lines in the pages are visibly separated and easily discriminated, as in Fig. 9(a).

Crammed lines, where all lines of the page are mingled and no clear line can separate successive lines, as in Fig. 9(b). This would require more processing to separate the lines.

Mixed line spacing, where some lines in the page are easily separable and others are crammed, as in Fig. 9(c).

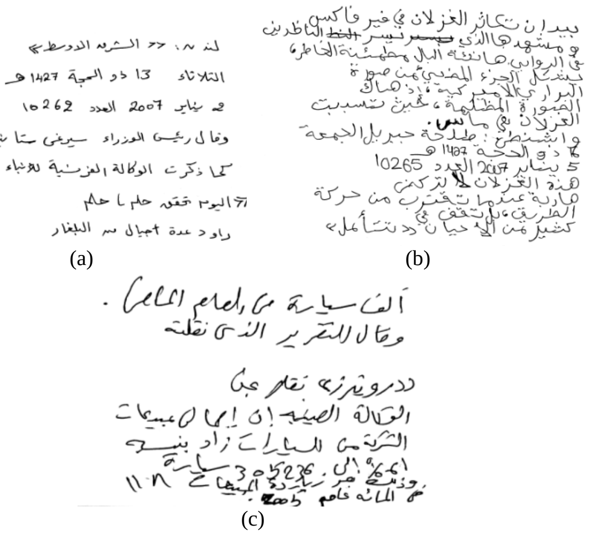


Fig. 9. Different types of line spacing.

G. Intra-word Spaces

Arabic is cursive in nature, however, some letters in Arabic are never connected from the left, causing many words to break at single or multiple positions creating the intra-word spaces [10]. As a standard in writing, inter-word space should be significantly larger than the intra-word space. This is usually the case but that is not always followed in handwriting. In some cases, intra-word space is significant and in some cases, it even surpasses its inter-word counterpart as in Fig. 10(a).

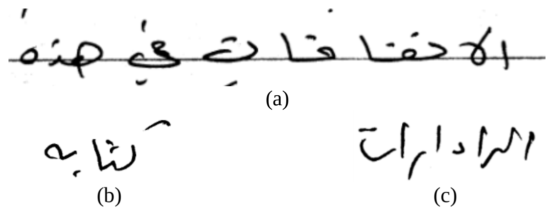


Fig. 10. Intra-word space examples.

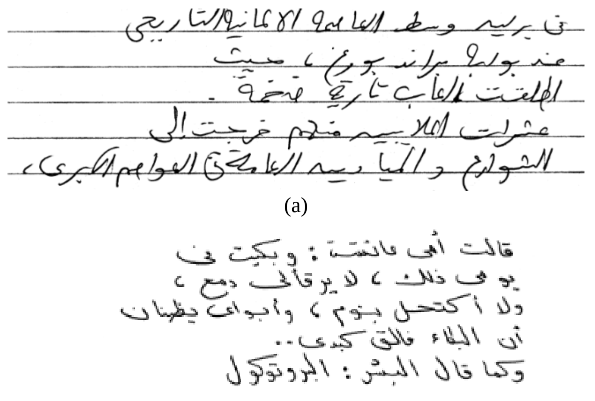
If a letter is composed of multiple strokes, which are written separately, the scribe may generate discontinuities in the single letter creating intra-word space at the letter level as in Fig. 10(b). This also needs to be identified and segmented properly.

Some words are composed of several parts that the number of separate parts cannot be determined sharply. The word in Fig. 10(c) is composed of eight horizontally aligned parts.

The ability to distinguish complete words without breaking them down improves the segmentation and helps in building a more meaningful document later after recognition.

H. *Slant*

Slant, as shown in Fig. 11, is the inclination of the downward strokes in handwriting. It originates from the scribes writing style. The two samples shown are of opposite slants. This has an impact on the validity of the extracted features, especially the characteristic angles in letters, of the handwritten text and the consequent sequence processing and recognition applied to features matrices [11]. Slant can be preprocessed before feeding word images to the recognition system or can be solved by the recognition system itself. It is noticed that slant is not frequent in MADCAT. In some documents, slant only exists in parts of the document.

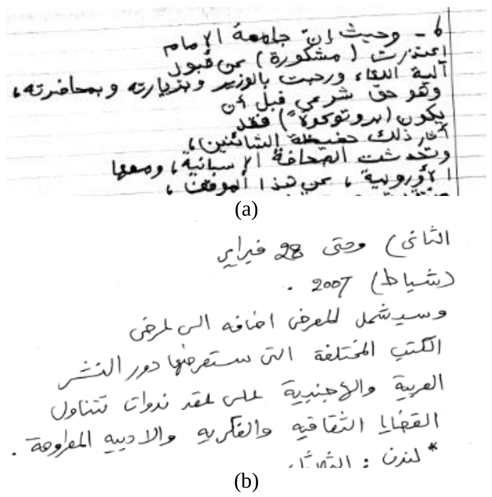

(a)

(b)

Fig. 11. Slant examples.

I. *Skew*

Skew of handwritten text is the angular deviation of the text from the horizontal line as shown in Fig. 12. Skew has significant impact especially on the Arabic recognition system because many systems use baseline estimation algorithms that are not robust against skew. Wrong estimation of the baseline leads to wrong feature detection and hence erroneous detection of letters and words.


(a)

(b)

Fig. 12. Types of skew.

After inspecting MADCAT, we noticed that skew can be categorized as follows:

1. Skew from image acquisition phase, or image skew where all lines share a relative skew angle as in Fig. 12(a).

2. Skew from writer, or writer skew where not all lines have the same skew angle as shown in Fig. 12(b).

Hence, the total skew is the vector sum of image skew and writer skew components. Image skew is simpler and can be corrected using collective algorithms that analyze the whole text body. Writer skew, on the other hand, is more complex and requires approaches that process each line or line component in the text body to detect and correct the skew of each line or each part of the line separately.

It was also noticed that if the paper is lined, then ruled lines orientation can determine the image skew and the document is free of writer skew.

J. *Bleed-through Text*

A single sheet of paper is not a completely opaque object due to its nature and thickness. In some cases, a problem arises when multiple pages are stacked one over another or when transcriptions exist on both sides of the paper sheet. Traces of the transcription on faces other than the one being scanned may appear like a bleed-through text as in Fig. 13. Fortunately, the density of bleed-through text is much less than the original text and thus may be treated as noise.

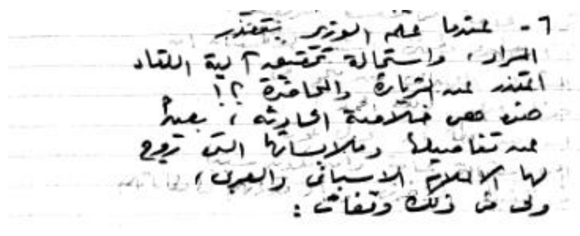


Fig. 13. Bleed-through text.

K. *Varying Text Body*

Images provided in MADACAT have almost the same resolution of around 5100×6600 at 600 dpi. However, text body highly varies in size, font height and size, line density, and reference position. In some cases, text starts immediately from the right or top edges of the image with no margins.

L. *Irrelevant Text or Numerals (Non-Arabic Characters)*

We noticed that some documents have some side notes or some irrelevant alphanumeric characters distant from the main text body as in Fig. 14. These characters have no reference in the corresponding ground truth files. Thus, they should be dropped, especially when written in foreign languages.

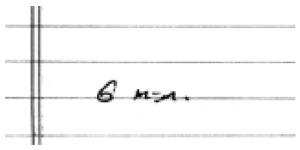


Fig. 14. Irrelevant text example.

M. Short Text Lines

The last line in a paragraph often ends before the end of the line. Some lines have only one or two words. This may affect line segmentation algorithms that assume that the line starts and ends at the edges of the page or text body (i.e., all lines are equal in length or a line is considered as such if it exceeds some length threshold).

N. Line Thickness Variations/Manipulations

The line thickness varies significantly from document writer to another and from document to another. In lined pages, the text stroke line can be thinner than the ruled line. This should be taken into consideration when designing the recognition stage or ruled line removal algorithm.

IV. Nonexisting Problems and Unrequired Preprocessing Operations

Some problems and preprocessing operations are ignored either because they were not found in MADCAT or because they are treated in the selected recognition system. These problems and preprocessing operations are:

- **Binarization**: MADCAT is already binarized.
- **Height normalization**: MADCAT resolution is sufficient for recognition and no height normalization is required.
- **Multiple text bodies**: MADCAT documents have single plain text body each.

V. Study of Available Solutions & Recommendations

Many solutions related to the problems mentioned above were thoroughly reviewed and assessed in [3]. We focus on the main problems that can cause significant performance deterioration for JU-OCR2 [2]. The following subsections summarize the main criteria of solution elimination and the recommended algorithms for the main two problems of handwritten documents, ruled lines and segmentation.

A. Ruled Line Removal

In order to evaluate an algorithm, one should first identify the requirements needed from that algorithm. In our case study, it is JU-OCR2 [2]. This posed some requirements from the line removal algorithm in addition to the requirements induced by the nature of the problem itself; ruled lines. The main requirements are:

- In case ruled lines are present, the algorithm should be able to handle them without distorting the text characters.
- The algorithm should be able to handle different grades of line degradation, refer to Fig. 6.
- The algorithm should not have effect on unlined pages; this can mainly be achieved in case a proper and non-destructive detection algorithm is utilized.
- The algorithm should handle horizontal and vertical ruled lines. Nevertheless, this is not mandatory because some simple techniques exist which expand the capability of the algorithms to process horizontal ruled lines to vertical ruled lines. For example, the image can be rotated 90 degrees and reprocessed by the same algorithm.

After review of current literature, most algorithms have reported competitive results, thus we base the selection of the line removal algorithm on the areas of coverage in the field of line removal.

The two finalists of the high-performance algorithms were the ones proposed in [12] and [13]. The algorithm proposed in [12] offers a wide spectrum of features and still maintained high performance. The features range from de-skewing for lined pages to robustness against broken ruled lines. It has no effect on unlined pages when handling of vertical and horizontal lines.

B. Segmentation

The requirements of the segmentation algorithm are:

- Perform line and word segmentation. JU-OCR2 only recognizes single words.
- Handle consecutive lines overlapping and handle touching lines and split components stretching between two consecutive lines.
- Properly handle crammed lines where lines are close to each other as illustrated in Fig. 9(c).
- Assign diacritics and secondary components to their corresponding lines/words.

For our case study, we needed an algorithm that can handle both line and word segmentation in order to cater to the needs of our selected recognition algorithm. We can either use a single algorithm that processes both or we can use two algorithms that each handles a stage.

Some algorithms were not considered due to the absence of performance evaluation data as in [14], [15], and [16]. Other algorithms were tailored for other languages like Oriya [17] and thus were excluded.

The algorithms in [18], [19] and [20] were excluded because they disregarded some of the real-life handwritten text properties as touching lines, varying text line length, non-uniform text line shape or the assignment of diacritics to their respective lines. This could greatly deteriorate the performance of the algorithm when using MADCAT.

The line segmentation algorithm proposed in [21] was selected as it satisfies all the requirements above for line segmentation, robust against skew types, and is more suited to Arabic texts.

As for word segmentation, we chose the most effective yet simple algorithm we found [21] as most of the problems were addressed in the line segmentation stage. We selected the algorithm proposed in [22] because it outperformed its peer that was proposed in [23].

We recommend the preprocessing system outlined in Fig. 15. It addresses most of the issues in Section III.

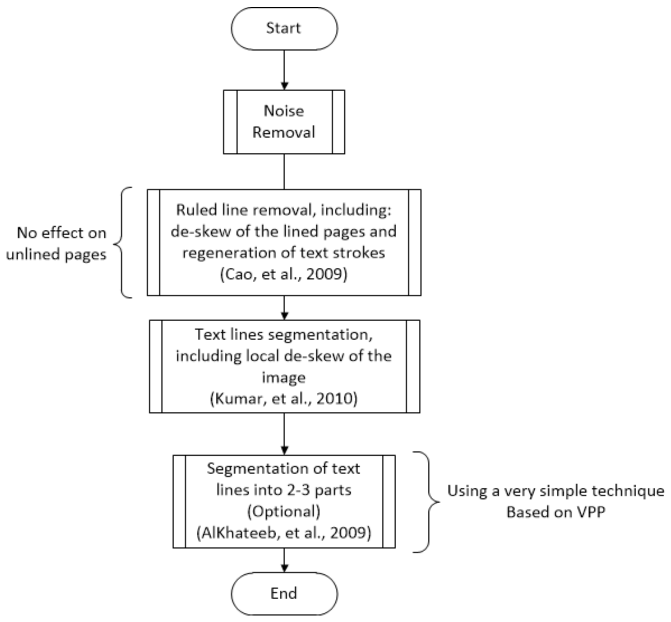

Start
Noise Removal
No effect on unlined pages
Ruled line removal, including: de-skew of the lined pages and regeneration of text strokes (Cao, et al., 2009)
Text lines segmentation, including local de-skew of the image (Kumar, et al., 2010)
Segmentation of text lines into 2-3 parts (Optional) (AlKhateeb, et al., 2009)
Using a very simple technique Based on VPP
End

Fig. 15. The proposed flow diagram of the preprocessing system.

VI. CONCLUSIONS

Prior to designing a handwritten document preprocessing or recognition system, one must examine and identify the problems they would face that may affect the performance. Also, knowing the problems helps form a comprehensive solution that handles them all.

The MADCAT dataset contains many performance-impacting challenges including pepper noise, vertical lines, bleed-through text, external graphical elements, writing errors, text lines overlapping, intra-word spaces, slant, skew, varying text body, irrelevant non-Arabic text, short text lines, and line thickness and writer style variations.

However, we have not found some other challenges that were found in literature in MADCAT. Examples include multiple text bodies and binarization. Nevertheless, one should still take these challenges into consideration when attempting to build a handwritten document preprocessing system or a comprehensive recognition system.

REFERENCES

[1] S. Strassel, “Linguistic resources for Arabic handwriting recognition,” Proc. 2nd Int’l Conf. on Arabic Language Resources and Tools, Cairo, Egypt, 2009.

[2] G. Abandah, F. Jamour, and E. Qaralleh, “Recognizing handwritten Arabic words using grapheme segmentation and recurrent neural networks,” Proc. Int’l J. Document Analysis and Recognition (IJDAR), vol. 17, no. 3, pp. 275–291, 2014.

[3] A. Al-Hourani, Preprocessing and Segmentation of Handwritten Arabic Documents for Writer-Independent Automatic Recognition, Masters Thesis, The University of Jordan, 2017.

[4] A. Tong, et al. “NIST 2013 open handwriting recognition and translation (OpenHaRT'13) evaluation,” Proc. 11th IAPR Int’l Workshop on Document Analysis Systems (DAS), 2014.

[5] Linguistic Data Consortium, “LDC Data Resources for OpenHaRT-13,” https://www.nist.gov/sites/default/files/documents/itl/iad/mig/OpenHaRT2013_WorkshopPres_LDC.pdf

[6] J. Kumar and D. Doermann, “Fast rule-line removal using integral images and support vector machines,” Proc. Int’l Conf. Document Analysis and Recognition (ICDAR), pp. 584–588, 2011.

[7] G. Abandah and F. Khundakjie, “Issues concerning code system for Arabic letters,” Dirasat Eng. Scies J., vol. 31, no. 1, pp. 165–177, 2004.

[8] Z. Shi, S. Setlur, and V. Govindaraju, “Removing rule-lines from binary handwritten Arabic document images using directional local profile,” Proc. 20th Int’l Conf. Pattern Recognition (ICPR), pp. 1916–1919, 2010.

[9] G. Abandah and M. Khedher, “Printed and handwritten Arabic optical character recognition – Initial study,” A report on research supported by The Higher Council of Science and Technology, Jordan, 2004.

[10] T. Malas, S. Taifour, and G. Abandah, “Toward optimal Arabic keyboard layout using genetic algorithm.” Proc. 9th Int’l Middle Eastern Multiconf. on Simulation and Modeling (MESM), pp. 50–54, 2008.

[11] G. Abandah and F. Jamour, “Recognizing handwritten Arabic script through efficient skeleton-based grapheme segmentation algorithm,” Proc. 10th Int’l Conf. Intelligent Systems Design and Applications, pp. 977–982, 2010.

[12] H. Cao, R. Prasad, and P. Natarajan, “A stroke regeneration method for cleaning rule-lines in handwritten document images,” Proc. Int’l Workshop on Multilingual OCR, 2009.

[13] K. Arvind, J. Kumar, and A. Ramakrishnan, “Line removal and restoration of handwritten strokes,” Proc. Int’l Conf. Computational Intelligence and Multimedia Applications, pp. 208–214, 2007.

[14] A. Amin and H. Al-Sadoun, “A new segmentation technique of Arabic text,” Proc. 11th IAPR Int’l Conf. Pattern Recognition, vol. II. Conf. B: Pattern Recognition Methodology and Systems, pp. 441–445, 1992.

[15] X. Du, W. Pan, and T. Bui, “Text line segmentation in handwritten documents using Mumford–Shah model,” Pattern Recognition, vol. 42, no. 12, pp. 3136–3145, 2009.

[16] C. Boiangiu, M. Tanase, and R. Ioanitescu, “Handwritten documents text line segmentation based on information energy,” Int’l J. Computers Communications & Control, vol. 9, no. 1, pp. 8–15, 2014.

[17] N. Tripathy and U. Pal, “Handwriting segmentation of unconstrained Oriya text,” Sadhana, vol. 31, no. 6, pp. 755–769, 2006.

[18] R. Manmatha and N. Srimal, “Scale space technique for word segmentation in handwritten documents,” Proc. Int’l Conf. Scale-Space Theories in Computer Vision, pp. 22–33, 1999.

[19] M. Bulacu, R. Koert, L. Schomaker, and T. Zant, “Layout analysis of handwritten historical documents for searching the archive of the Cabinet of the Dutch Queen,” Proc. 9th Int’l Conf. Document Analysis and Recognition (ICDAR), vol. I, pp. 357–361, 2007.

[20] A. Al-Dmour and F. Fraij, “Segmenting Arabic handwritten documents into text lines and words,” Int’l J. Advancements in Computing Technology, vol. 6, no. 3, pp. 109–119, 2014.

[21] J. Kumar, W. Abd-Almageed, L. Kang, and D. Doermann, “Handwritten Arabic text line segmentation using affinity propagation,” Proc.9th IAPR Int’l Workshop on Document Analysis Systems, pp. 135–142, 2010.

[22] J. AlKhateeb, J. Jiang, J. Ren, and S. Ipson, “Interactive knowledge discovery for baseline estimation and word segmentation in handwritten Arabic text,” Recent Advances in Technology InTech, 2009.

[23] S. Srihari, H. Srinivasan, C. Huang, and S. Shetty, “Spotting words in Latin, Devanagari and Arabic scripts,” Vivek-Bombay-, vol. 16, no.3, pp. 2, 2006.