Text Detection in Arabic news Video Based on SWT Operator and Convolutional Auto-encoders

Oussama Zayene[1, 2], Mathias Seuret[1], Sameh M.Touj[2], Jean Hennebert[1, 3], Rolf Ingold[1] and Najoua E. Ben Amara[2]

[1] DIVA group, Department of Informatics, University of Fribourg, Fribourg, Switzerland
{firstname.lastname}@unifr.ch
[2] SAGE group, National Engineering School of Sousse, Sousse, Tunisia
najoua.benamara@eniso.rnu.tn
samehmasmouditouj@yahoo.fr
[3] Institute of Complex Systems HES-SO, University of Applied Science Western Switzerland

***Abstract*—Text detection in videos is a challenging problem due to variety of text specificities, presence of complex background and anti-aliasing/compression artifacts. In this paper, we present an approach for horizontally aligned artificial text detection in Arabic news video. The novelty of this method revolves around the combination of two techniques: an adapted version of the Stroke Width Transform (SWT) algorithm and a convolutional auto-encoder (CAE). First, the SWT extracts text candidates' components. They are then filtered and grouped using geometric constraints and Stroke Width information. Second, the CAE is used as an unsupervised feature learning method to discriminate the obtained textline candidates as text or non-text. We assess the proposed approach on the public Arabic-Text-in-Video database (AcTiV-DB) using different evaluation protocols including data from several TV channels. Experiments indicate that the use of learned features significantly improves the text detection results.**

Keywords- Arabic text detection; SWT operator; CAE; AcTiV-DB

I. Introduction

TV news are important sources of information for most people. They allow to better understanding of the social, cultural and political events punctuating our daily lives.

Today, thanks to the rapid progress in mass storage technology, we can archive big amounts of digital news videos. As the archive size grows considerably, the manual annotation of all video sequences becomes impractical. This entails an urgent need to fast as well as effective information retrieval systems to ensure easy access to the relevant information contained in large news video archives.

Texts embedded in videos, especially captions, are one of the most important high-level information of the video content. They can be used as powerful semantic cues in multimedia content retrieval. Mainly, there are two kinds of text in videos, namely scene and artificial text. Compared with scene text, the artificial text provides brief and direct description of video content such as subtitles, speaker's name, place, event information, etc. Thus, we mainly focus on artificial Arabic text detection in videos.

Recognizing text in video sequences, often called Video Optical Character Recognition (Video-OCR), is an essential task in many applications like content-based multimedia retrieval, automatic broadcast annotation, large archive managing, etc. Therefore, the field has gained increasing attention of the researchers in the last decades [10, 12, 13]. A preliminary step to Video-OCR processing is to detect the text area in video frames. However, text detection is a challenging problem due to variety of text specificities (positions, fonts, sizes, etc.), presence of complex background with various objects resembling text characters and anti-aliasing/compression artifacts.

Major contributions have already been made in the field of Arabic OCR [1]. However, few attempts have yet been made on the development of detection/ recognition systems for embedded text in Arabic videos. Special characteristics of Arabic script include non-uniform intra/inter word distances, diacritics, cursive nature of the script, etc.

In this paper, we propose a novel text detection approach combining two main techniques: an adapted version of the original SWT algorithm [2] that enables efficient text candidate's localization and a deep learning-based textline verification method. We aim to stand out from the dominant methodology, based on so-called hand-crafted features [3, 4 , 6, 8, 10, 12]. This is done by automating the feature extraction process, i.e. an unsupervised feature learning method based on a convolutional auto-encoder (CAE) scheme. Then, a SVM classifier uses the CAE features to discriminate textline candidates as text or non-text. To the best of our knowledge, our approach is the first which combines CAE and SWT in a system specifically designed for detecting embedded Arabic texts in video.

To evaluate our text detector, we have conducted extensive experiments on the public AcTiV-DB [11] dataset under different evaluation protocols. The rest of the paper is organized as follows. Section II reviews related works on text detection. Section III presents our text detection method. Section IV illustrates our experiments. Finally, Section V draws conclusions and future work.

II. Related Works

Text detection methods in the literature can be grouped into texture-based, connected component-based and hybrid methods. Texture-based algorithms scan the image using generally multi-scale sliding windows to extract different texture proprieties and classify image areas as text or non-text based on texture-like features. Some widely used texture features include Histograms of Gradients (HOGs) [10], Wavelets [4] and Local Binary Patterns (LBP) [6, 8]. The technique introduced by Yang et *al.* in [3] applies an edge-based multi-scale text detector to identify text candidates that are then refined using an image entropy-based filter. Support Vector Machine (SVM) is applied as verification procedure to eliminate false alarms. Texture-based approaches are good for detecting text from complex background. But, they are very time consuming as all scales are exhaustively scanned.

Connected component based methods work in a bottom-up fashion by grouping neighboring pixels into connected components (CC) based on region properties between text and background, such as edge, size, color, stroke width and gradient information. The CCs are then filtered and grouped into words and textlines. Shivakumara et *al.* [4] extract CCs based on K-means clustering in the Fourier-Laplacian domain, and eliminate false alarms using edge density, text straightness and proximity. Zhuge et *al.* [5] present a CC-based algorithm which employs Maximally Stable Extremal Regions (MSER) as basic character candidates. Text CCs are then grouped into text lines using geometric information, and non-text CCs are excluded based on corner detection, multi-frame verification and some heuristic rules.

Literature can also be divided into heuristic-based approaches *versus* machine learning-based approaches [8], or spatial methods *versus* temporal methods [6, 14].

The hybrid methods can be a mixture of texture-based and CC-based methods, or a combination of heuristic-based and machine learning-based methods. In [6] Anthimopoulos et *al.* present a hybrid method combining a machine learning texture-based technique with a heuristic region-based refinement. Text blocks are firstly detected based on the edge map analysis. After that, dilation, opening, projection profiles and SVMs are introduced for refinements.

All the mentioned methods so far are dedicated to Latin or Chinese text detection. Only few researches are dedicated to Arabic texts. Ben Halima et *al.* [7] firstly use Multi Frame Integration (MFI) method to decrease background variations. Raws and columns, that contain text candidates, are then extracted using projection profiles. Finally, a three-layer perceptron is applied to refine the previous obtained textblocks. Youssfi et *al.* [8] propose three machine learning texture-based approaches to detect texts. The first one collects features from Multi-Block LBP representation and classifies text candidates using the Gentleboost Algorithm. The two other methods are based on a multiexit asymmetric boosting cascade using Haar-like features. Jamil et *al.* [9] present an edge-based method to detect Urdu (similar to Arabic script) text in video frames. The average gradient in the neighborhood of each pixel is firstly computed and the horizontally aligned gradients are merged together. An edge density filter is then applied to eliminate non-text regions followed by the application of some geometrical constraints. Unfortunately, all these methods are tested on private datasets with non-uniform evaluation protocols that make direct comparison and scientific benchmarking rather impractical. Comprehensive surveys can be found in [10, 12, 13].

III. Proposed Text Detection Method

Our system consists of two main stages i.e., CC-based heuristic algorithm and machine learning classification, as shown in Figure 1. The first stage extracts, filters and groups CC text candidates using SWT operator, geometrical constraints as well as textline formation method. The second stage uses convolutional autoencoders, to produce automatically features that have been learned from previously obtained unlabeled textline candidates. A SVM classifier receives as input these features for discriminating textlines from non-text ones.

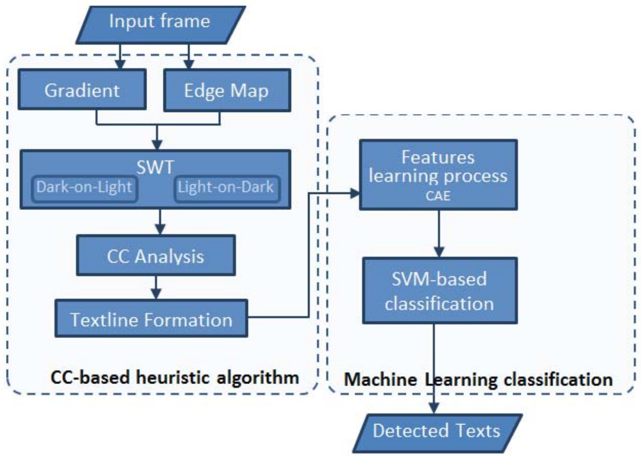


Figure 1. The flowchart of the proposed text detection method

A. Component Extraction by SWT

The SWT algorithm [2] is used to extract CCs from an input frame.

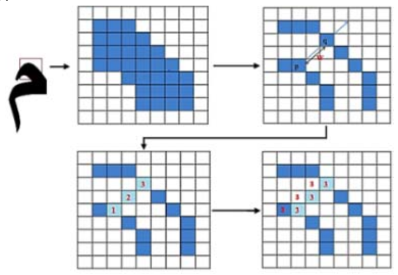
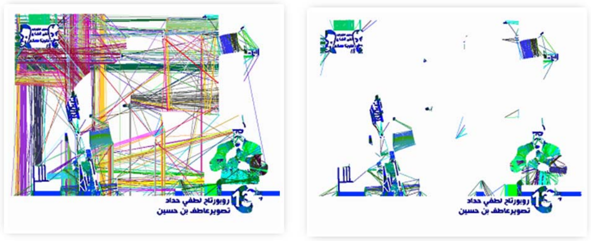


Figure 2. Example of SWT processing.

This operator detects stroke pixels by shooting a search ray from an edge pixel *p* to its opposite edge pixel *q* along the gradient direction *dg*. If these two edge pixels have nearly opposite gradient orientations, the ray is considered valid. All pixels inside this ray are labeled by the distance between p and q (as shown in Figure 2). In order to reduce the noises of incorrect connections produced by the SWT (like those in Figure 3(a)), we propose to discard the false rays whose length are higher than a predefined empirical threshold T_r.



(a) (b)

Figure 3. Results of the original SWT (a) and our modified version (b).

The neighboring pixels in the resulting SWT image are then grouped into CCs. In order to allow smoothly varying SWs in a letter, we keep the same SW ratio, which is 3, as in [2]. In Arabic script a single character may consist of several strokes and, subsequently, several labels. Considering this, we modified the original CC-labelling operation of [2] using a two-pass algorithm.

B. *Component Analysis and textline formation*

Coarse filtering: At this step, we apply a set of heuristic rules based on statistical and geometric proprieties of the components, to filter out CCs that are unlikely parts of texts. First, we remove components with very large and very small aspect ratio under a conservative threshold so that characters like Alif "أ" are not discarded. Then we discard objects with unusual sizes by limiting the length and width of the component. In addition, objects located at the border of the image are also discarded from further processes.

Vertical merging: Different from Latin script, an Arabic character may consist of several diacritic marks such as Hamza above/below Alif "أ،إ" or dots. Among the previously obtained candidate CCs, some of them are parts of a character, which need to be merged into a single bounding box. We design a small set of rules to group these CCs:

- The CCs should have similar SW (ratio between the median SW values has to be less than 2.0).
- The vertical distance between two CCs should not exceed an empirical predefined threshold T_{vd}.

Textline formation: In order to form the larger context of textual information, given the obtained character/subword candidates, we develop a textline grouping method. Specifically, we define an upper triangular probability matrix M given by (1), where $m_{i,j}$ is the matching probability corresponding to a pair of text candidates (C_i, C_j).

$$M=\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ 0 & m_{2,2} & \cdots & m_{2,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_{n,n} \end{pmatrix} \tag{1}$$

In order to compute $m_{i,j}$ for a given pair of components, we firstly calculate the following probability functions:

- *Ov(C_i, C_j)*: probability based on spatial overlap between their corresponding rectangles i.e., R_i, R_j, respectively.
- *Ds(C_i, C_j)*: probability based on the proximity of R_i and R_j. the closer R_i and R_j are, the more important *Ds(C_i, C_j)* is.
- *Al(C_i, C_j)* increases depending on components' alignment, since text always appears in the form of horizontally aligned lines.
- *Sw(C_i, C_j)*: probability based on SW similarity.

The probability matrix M is then calculated as follows:

$$m_{ij}=\begin{cases} 1 & \text{if } Ov(C_i,C_j) > T_{ov} \\ s & \text{if } Ds(C_i,C_j) > T_{ds} \text{ and } \\ & \quad Al(C_i,C_j) > T_{al} \text{ and } \\ & \quad Sw(C_i,C_j) > T_{sw} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Where

$$s=\frac{Ds(C_i\,C_j)+Al(C_i,C_j)+Sw\,(C_i\,C_j)}{3} \tag{3}$$

and T_{ov}, T_{ds}, T_{al} and T_{sw} are thresholds for, respectively, the overlap ratio, distance, alignment and stroke width scores. Text lines formation process consists finally in pairing C_i and C_j when $m_{i,j}$= max(M) with respect to a minimum matching probability threshold T_m. The process ends when no components can be grouped.

C. *Features Learning Process*

As the previous step produces text lines and areas erroneously recognized as text lines, we then use a machine learning method for classifying them as text and non-text. Machine learning methods take as input features giving information about the data and return as output a class label. There are various ways to extract features from data, for example by hard-coding mathematical or morphological operators. In this work, we aimed at automatizing this task; therefore we used an autoencoder as feature extractor.

Auto-encoders (AE) are artificial neural networks which are trained to encode and decode data. They usually learn either to encode the inputs with fewer dimensions (compression) or with a higher dimensionality (sparse representation). Convolutional auto-encoders are stacked autoencoders in which layers, excepted for the top, are convolved. This allows covering a larger area while keeping the number of weights of the neural network small enough to have an acceptable training time. The output of a CAE, which is used as features during the classification, is the encoded values, not the result of the reconstruction, as the latest is used only during the training phase.

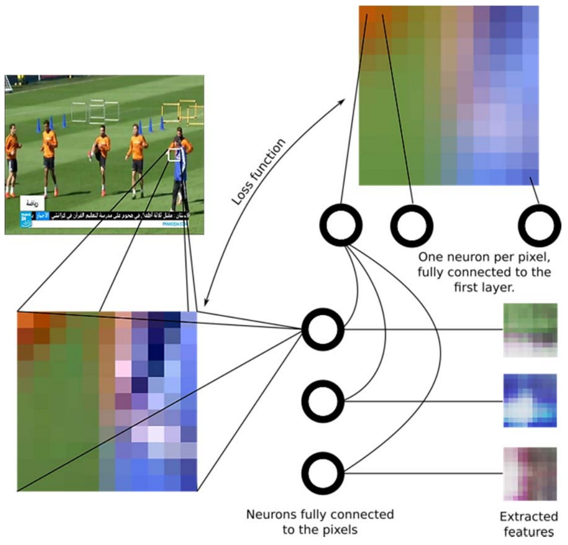

Loss function
One neuron per pixel, fully connected to the first layer.
Neurons fully connected to the pixels
Extracted features

Figure 4. Illustration of a single-layer autoencoder.

We used the CAE introduced in [15]. Each of its layers is composed of a convolved artificial neural network that has two neuron layers, one for encoding and one for decoding. A single layer autoencoder is illustrated in Figure 4. The first neural layer encodes the inputs, and the second neural layer, which is used only during the training phase, reconstructs the inputs from the encoded values. In case of stacked autoencoders, the first layer of the CAE takes raw pixel data as input; the other layers take as input the output of the previous layer.

Our AE encodes an input x of dimension n to an output y of dimension m as follows:

$$y_i = f\left(b_i^e + \sum_{j=1}^{n}\left(w_{i_j}^e . x_j\right)\right) \quad (4)$$

Where b^e_i is a bias and w_{ij} are the weights used for encoding. Decoding an output y to reconstruct the input is done in a similar way:

$$\chi_j = f\left(b_j^d + \sum_{i=1}^{m} w_{j_i}^d . y_i\right) \quad (5)$$

Where χ is an approximation of the x vector encoded by y, b_j a bias, and w_d weights used for decoding. This means that the AE has to learn m x (n + 1) + n x (m + 1) weights during its training. For this reason, it is more time-efficient to use a convolution of small AEs rather than training a single one covering a large patch.

The convolutions are created as follows. First, an AE covering W_1 x H_1 pixels and having m_1 outputs is trained. Then, we create W_2 x H_2 copies of it, and put them in a grid, with an offset of O_{1x} x O_{1y} pixels. This grid covers then $((O_{1x} . (W_2 - 1) + W_1)$ x $(O_{1y} . (H_2 - 1) + H_1))$ pixels. The output of the AEs in this grid can be seen as an array composed of W_2 x H_2 x m_1 values, which can then be given to a second level AE. When creating a convolution of the second level AE, in order to add a third level, the convolution of the first level AE must be redimensioned accordingly.

The layers of the CAE are trained one after another with standard back-propagation and gradient descent in their two-layers neural network, the goal being to minimize the reconstruction error $(\chi - x)^2$. The layers of the AE must learn to encode and decode their own input. If we back-propagated the reconstruction error of the top-layer to the previous layers, then the top layer would "ask" through back-propagation the previous layers to have easy-to-reconstruct values (e.g. constants). This would lead to a degeneration of the weights, making the AE useless. For this reason, we add a new layer to the network only when its current top-layer is sufficiently trained.

The i-th feature learned by a CAE can be displayed by setting manually the CAE's outputs to zero, excepted for y_i which is set to 1, and then decoding it layer after layer until the pixel-level is reached. Figure 5 shows some features which were learned automatically by the CAE on our data. We can see that the learned patterns are more complex when there are more layers.

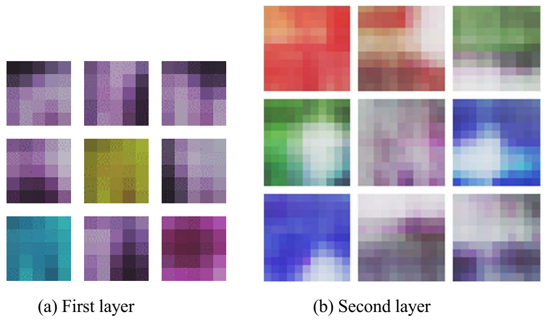


Figure 5. Illustration of features learned by two layers

While the CAE can be trained in an unsupervised way, its topology has to be manually defined: number of layers, size of the convolutions, offset and number of features.

D. SVM-based classification

We train a SVM classifier with the features of extracted patches from obtained textline candidates. We selected roughly as many patches from text candidates than from non-text ones in order to have a balanced training data.

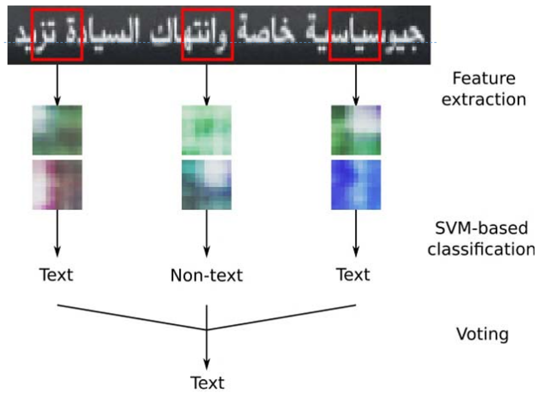


Figure 6. Principles of textline classification based on majority voting

In the prediction step, we classify patches located along the vertical center of the candidate. Other locations such as the bottom or the top of the candidate area might contain no text despite belonging to a text area. After that, a majority voting procedure is applied to classify the candidate textlines areas, as illustrated in Figure 6.

IV. Experimental Results

A. *AcTiV Dataset*

We evaluate the performance of our approach on a sub-dataset of the AcTiV-DB [11]. AcTiV-DB is the first publicly accessible annotated dataset designed to assess the performance of Arabic VIDEO OCR systems. The challenges that are addressed by AcTiV-DB are in text patterns variability (colors, fonts, sizes, position, etc.) and presence of complex background with various text-like objects.

AcTiV-*D* (*D* for Detection) represents a sub-dataset of non-redundant frames collected from the AcTiV-DB and used to measure the performance of our proposed systems to localize text regions in still HD/SD frames. AcTiV-*D* consists of 1843 frames (5133 textlines) distributed on four sets (one set per channel). Every set includes two sub-sets: *trainingFiles* and *testFiles*. Detection ground-truth is provided at the line level for each frame. We evaluate our work, specifically, in two protocols proposed by Zayene et *al.* in [11]. More details are in table I.

TABLE I. Evaluation protocols

	Resolution	Channel		Training textlines	Test textlines
Protocol 1	HD (1920x1080)	AljazeeraHD		803	226
Protocol 4	SD (720x576)	4.1	France 24	960	224
		4.2	Russia Today	1302	317
		4.3	ElWataniya 1	1068	233

B. *Parameter settings*

In all these tests, the parameters of the first stage (Section III-A & III-B) were set empirically as follows. In the components extraction module: the maximum ray length value T_r = 60 px. In the coarse filtering module: maximum character/subword height h_{max} = 40 px, character/subword width limit w_{max}= 120 px and max aspect ratio r_{max} = 5. In the vertical merging module: maximum relative vertical distance T_{vd} = 3 px. Note that these values concern SD channels. In case of HD channels, they should be doubled. The probability thresholds, in the textline formation procedure, were set at these values: T_{ov}=0.75, T_{ds}=0.35, T_{al}=0.35, T_{sw}=0.24 and T_m=0.5.

A fundamental part in our experiments consisted in optimizing the settings of the CAE, particularly its topology. We started with a single-layer autoencoder and a topology which was estimated as a good starting point: an input patch of a size slightly larger than the strokes of the text, and enough neurons for having a relatively good looking reconstruction. Then, we tried to optimize the topology by improving iteratively the number of features and the input patch size with regard to the classification accuracy. The optimal topologies found are given in Table II. It is interesting to notice first that the dimensions of the first layer input patch for the HD channel are twice larger than for the SD channels, and secondly that the optimal number of features does not change. The first is due to the difference of resolution (roughly twice higher for the HD channel), and the second can be explained by the fact that despite differences of resolution, the content of the inputs is similar and therefore requires a similar number of features.

The second layer of an autoencoder is more efficient when it receives as input useful data. Therefore, we used settings for the first layer which were optimal for the classification, as their outputs are certainly better for the classification task than when using other topologies. For this reason, when we started to create two-layer autoencoders, we used for the first layer the previously obtained settings, and optimized only the second layer.

We trained the CAEs on the obtained textline candidates. Thus, their features are trained to describe the kind of content that the autoencoders will have to deal with during the classification phase. For this training, we used between 2153 textline candidates (for France24 TV channel) and 3924 textline candidates (for ElWataniya1 TV channel), and used patches randomly placed on them for training the CAEs.

TABLE II. Optimal CAE Topology for HD/SD channels

	HD		SD	
	Layer 1	**Layer 2**	**Layer 1**	**Layer 2**
Input patch size	10*10	20*20	5*5	11*11
Convolution	3*3	-	3*3	-
Offsets	5*5	-	3*3	-
No. features	12	15	12	15

C. *Results*

To evaluate the proposed method, we compared it with two other systems. The results are given in Tables III and IV in terms of precision, recall and F-measure. The first method which we tried is Epshtein's [2]. The second, called here "System *A*", is a fully heuristic method, combining the first part of the workflow presented in Figure 1, and a refinement step using projection profiles, aspect ratio and contrast information [11]. We obtained results roughly 50% higher than Epshtein's method; however the high error rate could still be much decreased by the proposed system which replaces the refinement step by a machine learning approach; as shown in Table III and IV. For the protocol 1, the proposed system increased the F-measure by 10.5%. For the protocols 4.1, 4.2 and 4.3 (SD channels), the results are higher, with gains of respectively 15.6%, 21.7% and 14.5%.

TABLE III. EVALUATION RESULTS IN PROTOCOL 1

Protocol	Method	Recall	Precision	F-measure
1	Epshtein [2]	0.53	0.36	0.45
	System *A* [11]	0.76	0.77	0.76
	Proposed System	**0.83**	**0.85**	**0.84**

TABLE IV. EVALUATION RESULTS IN PROTOCOL 4

Protocol	Method	Recall	Precision	F-measure
4.1	Epshtein [2]	0.5	0.3	0.4
	System *A* [11]	0.6	0.69	0.64
	Proposed System	**0.73**	**0.75**	**0.74**
4.2	Epshtein [2]	0.42	0.36	0.39
	System *A* [11]	0.55	0.66	0.6
	Proposed System	**0.73**	**0.73**	**0.73**
4.3	Epshtein [2]	0.47	0.35	0.41
	System *A* [11]	0.71	0.68	0.69
	Proposed System	**0.83**	**0.76**	**0.79**

Figure 7 gives some examples of detection results using the proposed system. Figures 7–a and 7–c show the outputs of the first stage (before classification). Figures 7–b and 7–d show the final results.

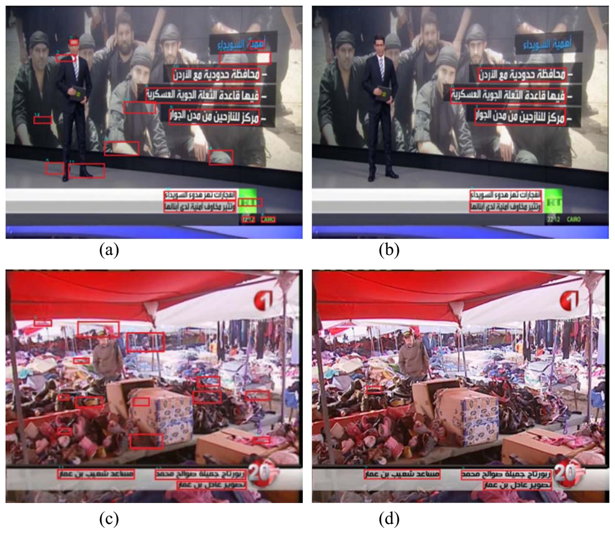


Figure 7. Some detection results from two different SD channels

V. CONCLUSION

This paper presents a new method for artificial Arabic text detection in video frames. The presented method is a combination of a CC-based heuristic approach using SWT and a machine learning approach based on CAE as features extractor and SVM as classifier.

While using machine learning for filtering the results given by the SWT increases much the accuracy, there is still much room for improvements. In the future, we would like to replace the combination of CAE and SVM by stacking a neural network on top of the autoencoder, thus having the possibility to fine-tune the features for the classification task. We also intend to investigate the possibility of directly classifying individual pixels as belonging to text or non-text, as we have shown in this work that the classifier is able to learn to distinguish these classes.

REFERENCES

[1] V. Märgner and H. El Abed, “Guide to OCR for Arabic Scripts” (book), Springer, 2012.

[2] B. Epshtein, E. Ofek, and Y.Wexler, “Detecting text in natural scenes with stroke width transform”, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2010.

[3] H. Yang, B. Quehl and H. Sack, “A Framework for Improved Video Text Detection and Recognition”, International Journal of Multimedia Tools and Applications (MTAP), pp. 217–245, October 2012.

[4] P. Shivakumara, T. Q. Phan and C. L.Tan, “New Fourier-Statistical Features in RGB Space for Video Text Detection”, IEEE transactions on Circuits and Systems for Video technology (CSV), pp.1520–1532, November 2010.

[5] YZ. Zhuge and HC. Lu, “Robust video text detection with morphological filtering enhanced MSER”, Journal of Computer science and Technology, pp. 353–363, March 2015.

[6] M. Anthimopoulos, B. Gatos, I. Pratikakis, “Detection of Artificial and Scene Text in Images and Video Frames”, Pattern Analysis and Applications, pp.1-16, 2011.

[7] M. Ben Halima, A.M. Alimi, H. Karray and A. Fernandez Vila, “Nf-savo: Neuro-fuzzy system for arabic video ocr”, Int. Journal of Advanced Computer Science and Applications, pp. 128–136, November 2012.

[8] S. Yousfi, S.Berrani, C. Garcia, “Arabic text detection in videos using neural and boosting-based approaches: Application to video indexing”, In Proceedings of the IEEE International Conference on Image Processing (ICIP), France, October 2014.

[9] A. Jamil, I.Siddiqi, F. Arif and A. Raza, “Edge-based Features for Localization of Artificial Urdu Text in Video Images”, In Proc. Of the International Conference on Document Analysis and Recognition (ICDAR), Beijing, China, September 2011.

[10] Tong L., Shivakumara P., Chew Lim T. and Wenyin Li., “Video Text Detection” (book), Advances in Computer Vision and Pattern Recognition (ACVPR), 2014.

[11] O. Zayene, J. Hennebert, S. M. Touj, R. Ingold, and N. E. BenAmara, “A dataset for arabic text detection, tracking and recognition in news videos- AcTiV”, in Proc. of (ICDAR), Nancy, France, August 2015.

[12] N. Sharma, U. Pal, M. Blumenstein, “Recent advances in video based document processing: a review”, in Proc. of the IAPR International Workshop on Document Analysis Systems (DAS), CA, USA, March 2012.

[13] Q. Ye and D. Doermann. “Text detection and recognition in imagery: A survey”. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), November 2014.

[14] L. Wu, P. Shivakumara, T. Lu and C. L. Tan, “Text Detection using Delaunay Triangulation in Video Sequence”, in Proc. of (DAS), Tours, France, April 2014.

[15] M. Seuret, A. Fischer, A. Garz, M. Liwicki and R. Ingold, “Clustering Historical Documents Based on the Reconstruction Error of Autoencoders”, in Proc. of the International Workshop on Historical Document Imaging and Processing (HIP), Nancy, France, August 2015.