A New Technique for Multi-Oriented Scene Text Line Detection and Tracking in Video

Liang Wu, Palaiahnake Shivakumara, Tong Lu, Member IEEE and Chew Lim Tan, Senior Member, IEEE

Abstract—Text detection and tracking in video is challenging due to contrast, resolution and background variations, different orientations and text movements, etc. Besides, the presence of both caption and scene texts in video aggravates the problem because these two text types differ in characteristics significantly. This paper proposes a new technique for detecting and tracking video texts of any orientation by using spatial and temporal information, respectively. The technique explores gradient directional symmetry at the component level for smoothing edge components before text detection. The spatial information is preserved by forming Delaunay triangulation in a novel way at this level, which results in text candidates. Text characteristics are then proposed in a different way for eliminating false text candidates, which results in potential text candidates. Then grouping is proposed for combining potential text candidates regardless of orientation based on the nearest neighbor criterion. To tackle the problems of multi-font and multi-sized fonts, we propose multi-scale integration by a pyramid structure, which helps in extracting full text lines. Then the detected text lines are tracked in video by matching the sub-graphs of triangulation. Experimental results for text detection and tracking on our video dataset, the benchmark video datasets and the natural scene image benchmark datasets show that the proposed method is superior to the state of the art methods in terms of recall, precision and F-measure.

Index Terms—Text detection, Text tracking, Delaunay triangulation, Multi-oriented video text detection, Multi-sized text detection.

I. INTRODUCTION

Rapid proliferation of multimedia contents that are available for broadcast and on the Internet leads to an increasing need for their ubiquitous access at anytime and anywhere over a variety of receiving devices, especially due to the escalating popularity of high performance and low price digital camera devices and smart phones [1-4]. Taking video in daily life is becoming very common leading to an explosive growth of video and hence a dramatic increase in size of heterogeneous video databases [5]. For example, the total number of YouTube video clips has amounted to over 120 million [5]. It is clear that when accessing lengthy and voluminous video programs, the ability to access highlights and to skip the less interesting parts of video will save not only a viewer’s time but also data downloading or airtime costs, especially when the viewer receives video wirelessly from remote servers. Moreover, it would be very attractive if users can access and view the content based on their preferences [1]. For instance, users prefer to retrieve exciting events or navigate events from personal collections for a particular person or vehicle. To realize the above needs, each source video has to be tagged with proper semantic labels. There are plenty of methods in the literature for annotating video based on their contents [6]. However, these methods may fail to annotate video with semantics due to the gap between low-level and high-level features [7-10]. This motivates researchers to use text information present in video to label the semantics of events because video texts provide a kind of high-level information closely related to the video content with the help of an Optical Character Recognizer (OCR). Therefore, text detection and tracking can serve as a basis for numerous multimedia applications, such as landmark recognition, navigation guidance for the visually impaired, enhancing safe driving, traffic monitoring, license plate detection, tourists and transportation information systems. Furthermore, in augmented reality, signs and texts of foreign languages contained in video or natural scene images can be automatically translated for visitors by their smart phones. In summary, text detection helps in annotating video data, which in turn helps in video summarization, personalized video retrieval, etc. All these applications require robust and effective text detection and tracking in video [1-5].

Generally, video contains two types of texts: (1) caption text which is manually edited, and (2) scene text which exists in video naturally. Since caption text is edited, we can expect texts to be of good clarity or contrast usually aligned in the horizontal or vertical direction. On the other hand, scene text is part of a video frame/image and its nature is unpredictable. Thus it suffers from non-uniform illumination, perspective distortion, low resolution, low contrast, varying font types and font sizes, multiple colors and arbitrary orientations, etc [11-19]. The presence of such types of text in video adds more complexity to the text detection and tracking problem. Therefore, accurate text detection and tracking in unconstrained environments is still an elusive goal for researchers.

Text detection and recognition is actually not a new problem for the multimedia content analysis field as we can see several methods for text detection in natural scene images in the literature [8-10, 20-23]. Usually, text in natural scene images has a complex background. To solve this problem, many methods have been developed [20-23]. Since scene images are often captured by a high resolution camera, these methods take the advantage of the high contrast property of scene texts to propose features based on the
shapes of characters. Therefore, these methods rely on connected component analysis or shape analysis of characters to achieve a good accuracy. However, these methods may not be used directly on video for text detection and tracking because video generally suffers from both complex background as well as low resolution. This results in a lot of disconnections and distorted shapes, and hence it is hard to preserve the shape of each character.

There are methods that have been proposed to detect texts in video by addressing the problems of low resolution and complex background in the literature [11-14]. These methods can be categorized as connected component based, texture based, and edge or gradient based methods. Since connected component based methods [24, 25] require the full shape of each character, these methods may not be suitable for text detection in video. To address the problem of complex background, texture based features are developed [17, 18, 26, 27]. These methods are computationally expensive and their performance lies in the training of a proper classifier and a large number of training samples. To achieve efficiency, the methods that are based on edge or gradient information have also been developed [6, 15, 16, 19, 28, 29]. These methods work well with less computation, but are sensitive to background giving rise to more false positives. Overall, most of the above methods focus on caption text detection in video but not for both caption and scene texts. Several methods have been developed to find a solution for the presence of both types of texts with different orientations in video [19, 26, 27, 29]. These methods basically explore the contrast information of text rather than the use of the characteristics of text. Despite solving the problem of orientation, these methods do not utilize the temporal information of video, but rather rely on individual frames. Therefore, these methods cannot be used for text tracking in video.

II. RELATED WORK
This section provides a literature review of the existing methods [18, 30-40], which use individual frames and temporal frames for text detection and tracking in video, respectively.

Li et al. [18] proposed a method for video text tracking based on wavelet and moments features. This method uses the advantage of wavelet decomposition and spatial information provided by the moments with a neural network classifier for identifying text candidates. Text block shape features are used to track texts in temporal frames. Huang et al. [30] proposed a method for scrolling text detection in video using temporal frames. This method uses motion vector estimation for detecting texts. However, this method is limited to only scrolling texts but not arbitrarily oriented texts. Zhou et al. [31] exploited edge information and geometrical constraints to form a coarse-to-fine methodology to define text regions. Then candidate regions are labelled as connected components by morphological operations. Based on temporal redundancy, text authentication and enhancement are done. Mi et al. [32] proposed a text extraction approach based on multiple frames. The edge features are explored with similarity measures for identifying text candidates. Wang and Chen’s method [33] used a spatio-temporal wavelet transform to extract text objects in video. Statistical features are used with a Bayesian classifier for final text region classification. Huang [34] detected video scene texts based on video temporal redundancy. The method performs motion detection in 30 consecutive frames to synthesize a motion image. Further video scene text detection is implemented in single frames to retrieve candidate text regions. Finally, the synthesized motion image is used to filter out candidate text regions and only keep the candidate text regions that have motion occurrence as the final scene texts. Huang et al. [35] proposed a method for video text detection using temporal frames based on motion features by integrating multiple frames, which give text regions. Corner points are used to find candidate text pixels and then a growing process is proposed for connecting candidate text pixels. However, this method focuses on horizontal graphics texts but not arbitrary scene texts. Zhao et al. [36] proposed an approach for text detection using text corners in video. This method proposes to use dense corners for identifying text candidates. From the corners, the method forms text regions using morphological operations. Then the method extracts features such as aspect ratio and orientation from the regions to eliminate false ones. Finally, optical flow is used for the verification of text blocks. Liu et al. [6] proposed a method for video caption text detection using stroke like edges and spatio-temporal information. A color histogram is used for segmenting texts. Li et al. [37] proposed a method for video text detection using multiple frames integration. This method uses edge information to extract text candidates. Morphological operations and heuristic rules are proposed to extract the final texts from video. Moseleh et al. [38] proposed an automatic inpainting scheme for video text detection and removal based on stroke width transform to identify text objects. Then motion patterns of text objects of each frame are analyzed to localize video texts. The detected text regions are removed, and finally video is restored by an inpainting scheme. The objective of the method is to restore missing information due to overlapping caption texts. The method is proposed to detect horizontal texts but not for arbitrary text detection in video.

A few methods have been developed for text detection and tracking text in video [5, 18]. Yusufu et al. [5] proposed a video text detection and tracking system based on SURF features and a classifier. This method proposes many heuristics based on edge and morphological information for identifying text candidates. Moving text blocks are identified using motion estimation. The text blocks in a previous frame and the current frame are tracked when the features are found to be matching. We can observe from these methods that they are developed to track caption texts in the horizontal direction because detecting and tracking caption texts is much easier than detecting scene texts. Therefore, none of the methods addressed the problem of scene text detection and tracking in video regardless of orientation, non-linear movement with multilingual ability.

Similar to this work, Wu et al. [19] recently proposed a method for detecting both caption and scene texts in video. The scope of this method is limited to text detection but not tracking. In addition, the method does not work well for texts of multiple fonts or sizes in video. Optical flow based properties have also been proposed by Shivakumara et al. [39] for dynamic curved text detection in video. The method is good when text is moving while the background remains static. It thus fails when the background moves. Moreover, the scope of the method is limited to text detection but not tracking. Arbitrary text detection is also proposed by Shivakumara et al. [29] by proposing gradient vector flows and grouping. Though the method finds a solution to the complex curved text detection problem, it does not explore temporal information for text detection in video. Therefore, the methods cannot be used for text tracking.
In summary, the above discussion shows that most of the current methods focus on caption texts in the horizontal direction for detection. The methods make use of temporal frames for enhancing text detection performance but not for tracking them in video. The use of spatial information for text detection and temporal information for tracking is ignored. Therefore, there is an immense scope for developing new methods that are capable of detecting and tracking texts in video accurately and efficiently irrespective of text types and orientations.

In this paper, we propose a new technique for text detection and tracking based on spatial and temporal information. Inspired by the work presented in [6] for caption text detection using temporal frames, we propose to use spatial information in a different way, namely inter or intra character symmetry and dense corners by Delaunay Triangulation for identifying text candidates. The proposed technique tracks every text by identifying its motion status based on transformations and templates. Moreover, the technique proposes multi-scale integration in a pyramid structure to tackle multi-font and multi-size problems. The main contributions are three folds: (1) Exploring invariant features, namely, symmetrical features that exist in inter and intra characters with the help of Delaunay Triangulation for identifying text candidates. These features work for any orientation and any type of texts and scripts; (2) The use of multi-scale integration for multi-sized and multi-oriented text lines detection in a new way; and (3) Tracking arbitrarily oriented texts through motion status identification in video by matching the sub-graphs given by Delaunay Triangulation with respect to different motion status.

III. PROPOSED APPROACH

The flow diagram of the proposed technique is shown in Fig. 1, where we can see that the proposed technique consists of two parts. The first part focuses on text detection, while the second part focuses on text tracking in video. As we are inspired by the work presented in [18, 36, 38] for text detection and tracking, where the methods used the first frame for text detection and then use temporal frames for text tracking and restoration, we follow the similar way to use the first frame in video for text detection and then temporal frames for text tracking. It is true that character components have double strokes (parallel edges) with nearly constant stroke width [9]. This cue indicates the presence of texts. With this cue, we derive symmetry between strokes based on the gradient direction to smooth the edge map of an input frame, resulting in the elimination of unwanted components. In the same way, it is noted that character components in the same text line symmetry between intra and inter character components due to regular spacing between character components and constant stroke width [10]. This observation motivates us to propose Delaunay Triangulation for the corners of the smoothed edge map, which extracts inter and intra symmetries by studying the spatial relationship between the corners. This results in text candidates for the input frame. The proposed method explores geometrical properties of sub-graphs formed by triangulation, such as edge strengths and density for text candidates to eliminate false ones, which results in potential text candidates. It is true that multi-level decomposition in a pyramid structure is an interesting idea for tackling the multi-sized or multi-font texts [18]. Therefore, we propose multi-scale integration in a pyramid structure on different scales to extract the full text line. We can assert that the proposed features are independent of orientation, font, font size, orientation, text type and script. This is the main advantage of the proposed method, which is a departure from the existing methods. In the second part, the detected texts are tracked in video with respect to motion status such as zoom in, zoom out and rotation along with arbitrary movements (linear or non-linear) with 2D transformations and the KLT tracker. While tracking, the proposed technique tests the same invariant features mentioned above to find matches. The complete flow of the proposed technique can be seen in Fig. 1.

A. Gradient Directions Pair for Smoothing Edge Map

For each video frame, the proposed technique obtains its Canny edge image using Canny edge operator because Canny operation gives fine details for both low and high resolution texts in video compared to the other edge operators such as Sobel and Prewitt operators [10]. However, due to the complex background of video, Canny operation often gives erratic edges for the components in the background. To eliminate such erratic edges, we propose the idea of symmetry in gradient directions because of the fact that character components have parallel edges and exhibit symmetry in shape. We divide the directions of each pixel on an edge component in the edge map into eight bins as shown in Fig. 2(a) to study the distribution of directions. The technique performs grouping by merging the gradient magnitude values that have opposite directions. Essentially, this leads to four paired groups as shown in Fig. 2(b), where we can see the parts that have different colors of both inner and outer contours share the same patterns. This is called symmetry. It is illustrated in Fig. 3, where (a) and (c) show sample text and non-text components with their respective Canny and gradient maps, respectively, while (b) and (d) show the histograms of the four directional pairs for the images in (a) and (c), respectively. It is observed from Fig. 3(b) and (d) that the bars in the histograms of the four paired groups look almost the same, while the bars of non-text components do not. This is valid because these pairs are formed based on the fact that character components have regular shapes and symmetric in nature, while non-text components do not exhibit regular shapes and hence they may not satisfy the directional symmetry condition. If any component in the edge map satisfies such symmetry, it is considered as a text component, or else it is considered as a non-text component. More specifically, the algorithmic steps are as follows.

![Flow chart of our text detection, tracking and motion analysis](image-url)
Fig. 2: Parallel double edges of character components

First, we calculate the gradient map for the input gray frame. We segment the gradient direction from 0 to 360° into eight parts as shown in Fig. 2(a). Then we consider each part and its corresponding part whose direction is opposite to it as one pair, resulting in four pairs of parts as shown in Fig. 2(b). For example, (337.5°, 360°) U (0, 22.5°) and (157.5°, 202.5°) are said to be one pair as these have opposite directions to each other with the same gradient magnitude values. The effect of symmetry to eliminate non-text components which appear like straight lines on the full frame can be seen in Fig. 4, where (a) is the gray input frame, (b) is the gradient map with colors of different directions, and (c) is the smoothed edge map after filtering. However, we can notice from Fig. 4(c) that there are non-text components that satisfy the symmetry. Therefore, this step is considered as the preprocessing step as smoothing the edge map, which helps subsequent steps for obtaining better text candidates.

![Fig. 4: Effect of smoothing](image)

**B. Spatial Study of Corners for Text Candidates Selection**

It is noticed that in the previous step the proposed symmetry misclassifies non-text components as text components due to the complex background of video. To filter out such non-text components, we propose corner and spatial study. Corner based features for text detection is a topic of interest due to its resilience to font, font size, orientation, text type, and to some extent distortion [36]. Initially, the proposed technique obtains corners using Harris method as shown in one sample in Fig. 5(a), where one can notice that small curvatures are considered as corner points. Therefore, we propose modifications to the conventional Harris corner detection algorithm based on edge angle and its quantization as follows.

![Fig. 5: Edge angle at corners for corner verification](image)

Let \( P \) be a corner point detected by the Harris method as shown in Fig. 5(a), we locate it at the center of a window of size \( N \times N \). We determine empirically the size of \( N \) as 9 in this work. Along the edge points in the window, we find the farthest edge point from the center. Then the proposed technique finds an angle between the farthest point and the centroid as shown in Fig. 5(b). If this angle is too small then we ignore the corner which is shown in Fig. 5(b), where the corner points do not contribute much deviation. On the other hand, the corners points shown in Fig. 5(c) contribute enough deviation and hence these corners are considered as correct corner points. In this way, the proposed technique finds correct corners as shown in Fig. 5(d). The effect can be seen in Fig. 5(f), where the corner points that represent non-text component as shown in Fig. 5(e) are removed.

As stated in [40], Delaunay triangulation is good for preserving spatial relationship between corners to study the characteristics of objects irrespective of orientation and to some extent distortion.
Thus we propose to explore Delaunay triangulation for the detected stable corners to study their spatial features such as the proximity between the corners, and inter/intra character components through connected graph and sub-graph formation to identify text candidates. These properties motivated us to propose Delaunay triangulation for text detection and tracking, which requires robust features mentioned above to achieve good results. Further, these features help us to distinguish text and non-text components as shown in Fig. 6(a) and Fig. 6(b), where the triangular mesh pattern reflects text characteristics such as regular spacing between characters and constant stroke width distances, while the pattern of non-text does not. This is the advantage of Delaunay triangulation.

The same observation can be seen in Fig. 7(a), where the triangulated meshes for the whole image Fig. 5(f) using corners. The proposed technique extracts the above features, which represent video texts through sub-graph properties as follows. For each triangle, the technique calculates the distances between nodes to eliminate the edges that connect from one text line to another as shown in Fig. 7(a), where those edges are marked by white color because they do not contribute to text components detection. Since the spacing between character components, words or successive lines is lesser than the spacing between two random lines, the distance between the two nodes of the character components that belong to two successive text lines would be small compared to the distance between the nodes that represent two farthest text lines, as shown by the white color edges in Fig. 7(a). This results in the removal of unwanted edges as shown in Fig. 8(a), where the white color edges in Fig. 7(a) are removed. Mathematically, it can be defined as

\[
Sp_{pq} = len_{pq}
\]

(1)

where \(Sp_{pq}\) is the value of spatial proximity between nodes \(p\) and \(q\), \(len_{pq}\) is the length of the edge between nodes \(p\) and \(q\). The threshold is determined automatically by calculating the average distance between characters, words and lines in the smoothed edge map. This is valid because the directional symmetry eliminates most of the non-text components from the Canny edge map. As a result, the image retains more text components.

Second, for each component in the smoothed map, we estimate stroke width using the gradient direction as described in [21]. Generally, stroke width distance is constant throughout a character [21, 40]. Therefore, we estimate stroke width consistency to identify text candidates. According to [21], the stroke width for a pixel \(p\) is defined as a ray \(r = p + n \cdot d_p\), which moves in a direction perpendicular to the stroke direction from \(p\) until it reaches another edge pixel, say \(q\). Here \(d_p\) denotes stroke orientation, and \(d_p\) is estimated for pixel \(q\). The ray from \(p\) moves approximately opposite to \(d_p\) \((d_p = -d_q \pm \pi / 6)\). The distance between \(p\) and \(q\) is considered as the stroke width. We also find the stroke width for the opposite gradient direction when the gradient directions point to the outside of a part, which means the background is darker than the character part as shown in the SWT images in Fig. 7, where (b) shows the stroke width of the gradient direction and (c) shows the stroke width of the opposite direction. With these stroke widths distances, we calculate stroke width consistency by estimating the standard deviation of stroke widths along the graph edges. The formulation is as follows.

\[
CS_1^{p,q} = \sqrt{\frac{1}{len_{pq}} \sum_{i=1}^{len_{pq}} \left( \frac{SWT_k(p_{i} + \frac{s_p - p}{g_p} \cdot i)}{SWT_k^{p,q}} - 1 \right)^2}
\]

(2)

\[
CS_2^{p,q} = \frac{1}{len_{pq}} \sum_{i=1}^{len_{pq}} \left( \frac{SWT_k(p_{i} + \frac{s_p - p}{g_p} \cdot i)}{SWT_k^{p,q}} - 1 \right)^2
\]

(3)

\[
CS^{p,q} = \min(CS_1^{p,q}, CS_2^{p,q})
\]

(4)

Where \(SWT_k^1\) and \(SWT_k^2\) are two stroke width maps respectively generated from two opposite directions. \(CS^{p,q}\) is the value of the stroke width consistency of the edge between nodes \(p\) and \(q\) calculated from \(SWT_k\), and \(CS^{p,q}\) is the smaller one between \(CS_1^{p,q}\) and \(CS_2^{p,q}\). \(SWT_k(\sigma)\) is the stroke width at pixel \(p\). \(SWT_k^{p,q}\) gives the average \(SWT_k\) value of the edge between nodes \(p\) and \(q\), which is represented as

\[
SWT_k^{p,q} = \frac{1}{len_{pq}} \sum_{i=1}^{len_{pq}} SWT_k(p_{i} + \frac{s_p - p}{g_p} \cdot i)
\]

If the stroke width consistency of one edge \(CS^{p,q}\) is too large, then we prune this edge. As a result, the proposed technique retains the edges that satisfy stroke width consistency and eliminates the edges that do not satisfy as shown by the white color edges in Fig. 8(b).
results can be seen in Fig. 8(c), where the white color edges that represent the spacing between text lines are successfully removed because these strokes do not meet stroke width consistency.

Fig. 8(c) still contains some unwanted edges that connect the components of one line to the components of another line. We propose one more feature called Canny rate which considers dense corners of text components. For each component in the smoothed edge map, we propose to use Canny rate through triangulated mesh as shown in Fig. 8(c) to identify candidate edges. Before testing the third property, we perform dilation over the smoothed edge map to connect small gaps between the edges as shown in Fig. 9(a) for the smoothed edge map in Fig. 4(c). As a result, we eliminate those edges in non-text regions, which are shown in Fig. 9(b) with white color. Formally, we define this feature as canny rate below.

\[
Cr_{\text{pq}} = \frac{|E_{\text{pq}}(t) \cap C_{\text{pq}}(t)|}{|E_{\text{pq}}(t)|}
\]

where \(Cr_{\text{pq}}\) is the canny rate of the edge between nodes \(p\) and \(q\), \(E_{\text{pq}}(t)\) and \(C_{\text{pq}}(t)\) respectively represent the set of pixels belonging to the edge between nodes \(p\) and \(q\), and the set of pixels belonging to Canny edge components. We expect equation (5) to give high values (it is set to 1) for video texts and low values for non-texts in video (other than 1). The final result is shown in Fig. 9(c). In this way, the above three features help us to identify the candidate edges that represent text candidates.

![Illustration of Canny rate](image1)

**C. Directional Region Grouping for Merging Text Candidates**

Each text candidate given by the previous step is considered as a cluster. In order to extract the full text line, we need to group all the clusters that represent text candidates. For this purpose, we propose Directional Region Grouping (DRG) which first finds the direction of text clusters and then groups the nearest neighbor clusters in the text direction. Initially, the technique chooses a seed cluster as shown in Fig. 10(a) and then finds a direction using Principal Component Analysis (PCA) if the cluster has more than two character components. Here we consider the largest Eigen vector to find the angle for the text cluster. The proposed technique searches for the nearest neighbor cluster by finding the distance between the seed cluster and its neighbor clusters as shown in Fig. 10(b) along the text direction given by PCA as shown in Fig. 10(c). The final result of grouping for one line can be seen in Fig. 10(d). For the whole frame, DRG gives results as shown in Fig. 10(e), where it fixes a bounding box for the grouped regions. This grouping will terminate when the cluster set becomes empty. This idea works well because of the fact that the spacing between character components is usually less than the spacing between text lines. The advantage of DRG is that it works well for any orientation of text lines. The DRG is illustrated in Fig. 10, where bounding boxes are fixed for text regions.

![Illustration of DRG](image2)

**D. False Positives Removal**

Due to the problem of low resolution and complex background, it is hard to remove false positives completely with the above features. Therefore, we further propose two rules of cluster groups, which represent text lines to reduce false positives.

Rule 1 is based on dense corners. It is true that due to more cursive nature of text, one can expect dense corners for text blocks and low corner density for non-text blocks. With this notion, we define rule-1 as the number of corners of a text block should not be a too small value. Let \(Ngc\) be the number of corners in a text block. If it is smaller than a threshold \(NT\), we consider it as a false positive.

Rule 2 is based on edge strength. For an ideal text block, the number of edges of the text block should be larger than the number of corners in the text block. With this property, we define rule-2 as the ratio of the number of edges in a text block divided by the number of corner points in the same block. This results in edge strength of a text block as defined in equation (6).

\[
Seg = \frac{Nge}{Ngc}
\]

where \(Seg\) denotes edge strength and \(Ngc\) as earlier defined is the number of corners points in the text block. Thus, if \(Seg\) is smaller than a threshold \(ST\), we discard it or else it is considered as a text block. The algorithmic steps of the false positive verification are shown in Fig. 11. The false positive elimination is illustrated in Fig. 12, which shows two types of false positives that are denoted by green color and blue color. The output of Section C is considered as text blocks for text block verification. The effect of the two rules can be seen in Fig. 12, where false positives (green and blue color components) shown in Fig. 12(a) are removed as shown in Fig. 12(b). One more example is shown in Fig. 12(c) and (d) where the non-text components created by the background complexity are eliminated in Fig. 12(d) using equation (6).
The two threshold values for $NT$ and $ST$ are determined empirically by conducting experiments on the recent benchmark ICDAR 2013 training dataset, which for the first time include video data for detecting texts. The details of the experiments can be found in the Experimental section.

In this work, we prefer to propose objective rules for removing false positives rather than using a classifier which is popular nowadays for the purpose of false positive elimination [41]. The main reason is that the objective of the proposed work is to propose a generalized algorithm for text detection and tracking, which should work well irrespective of scripts and orientations. This is also because text detection and tracking is actually a preprocessing step to text recognition in video. The proposed two rules work based on dense corners and edge strength of the text components. We believe these rules behave like objective heuristics but not hard and fast rules which may be sensitive to different situations. In the light of this, if we use a classifier for removing false positives, it may restrict the ability to detect and track the text of different script for different datasets. In addition, finding training samples for non-text components is not trivial because the boundary of a non-text is unknown and sometimes difficult to define for a classifier.

E. Multi-Scale Integration for Multi-Sized and Multi-Oriented Text Detection

Since the nature of scene text in video is unpredictable, it can have any orientation and font size. We propose multi-scale integration to handle this ill posed problem. We use a three-level pyramid structure approach to detect texts with a wide range of font sizes. For the input frame, we apply the steps from subsection A to subsection D on three levels to find text blocks. Then the bounding boxes at different levels are mapped to find actual text blocks. If a bounding box at one level does not overlap with the boxes at other levels, we consider it as a separate box that has a single text line. If the bounding box of a text line overlaps with the bounding boxes of the same text line at multiple levels, then we merge them by forming a new bounding box that can include all with union operation. The whole process is illustrated in Fig. 13, where (a) shows the results of the original input frame, (b) gives the results of scale 1 where we can see some of the characters are missed without bounding boxes, (c) illustrates the results of scale 2 where some of the characters are detected with bounding boxes and few characters are missed, and (d) shows the final results after integrating all the text blocks at scale 0, scale 1 and scale 2, where we can see all the texts are covered by proper bounding boxes. This is the advantage of multi-scale integration.
We determine different scales automatically with the original input frame of scale 0 as follows. Each scale of image is $0.8 \times 0.8$ of its next upper scale. This is required to keep the height of the top image in pyramid to be smaller than 100. Therefore, the input images of different sizes produce different numbers of layers or scales in the pyramid structure as defined in equation (7)

$$SN = \log_{0.8} \frac{100}{H}$$

where $SN$ gives the number of scales and $H$ is the height of the input image.

$$[x, y] = [S_2 x, S_2 y]$$

$$\alpha \rightarrow p_0 [x, y]$$

$$\alpha \rightarrow M_\alpha$$

$$\alpha \rightarrow M_\alpha$$

### IV. TEXT TRACKING THROUGH MOTION STATUS ANALYSIS

Text detection step gives texts with bounding boxes for the text lines in video. In this section, we track text lines of any orientation, font size at different levels to track text lines. We propose to use KLT tracker as it is simple and good for tracking texts in video. The tracking step involves motion status analysis and tracks text lines based on identifying motion status. In this work, we consider linear motion as well as non-linear motion. We propose to use 2D transform matrix for identifying motion status. Since the proposed technique identifies motion status, it works well for both linear and non-linear motions and hence improves text tracking results. This is the advantage of the technique compared to the existing tracking methods [18, 36, 38]. As detected text regions are represented by sub-graphs of corner points at different levels, the KLT tracker tracks them using corners of text regions in the same level where texts are detected and generates a set of trajectories within a specific time window of video sequence. The length of time window is named $\tau$, which means the number of frames in the video sequence.

#### A. Motion Status Analysis

Text motion in digital video consists of three types: static, simple linear and complex nonlinear such as zooming in and out or rotation. The goal of this section is to respectively derive the transformation matrices for zooming, rotation and translation, namely, $M_z$, $M_r$ and $M_t$, which are all $3 \times 3$ matrices. Once we find these transformation matrices then it will be easy to identify the status of text motions. The theoretical analysis for deriving these transformation matrices is as follows.

The position of point $p_0 [x, y]$ after zooming, rotation or translation can be obtained by

$$[x', y'] = [x, y] \cdot M_t$$

where $M_t$ represents one of 2D transformation matrices, namely,

$M_z, M_r$ and $M_t$. So $[x, y] \cdot M_z \cdot M_r \cdot M_t$ are the transforms required for point $p_0$ to identify its motion status, such as zooming, rotation and translation, respectively, which and it is illustrated in Fig. 14. Let a 2D transform matrix $M = M_z \cdot M_r \cdot M_t$, then

$$[x, y] \cdot M_z \cdot M_r \cdot M_t = [x, y] \cdot M,$$

where

$$M = \begin{bmatrix} s_x \cos \alpha & s_x \sin \alpha & 0 \\ -s_y \sin \alpha & s_y \cos \alpha & 0 \\ dx & dy & 1 \end{bmatrix}$$

in which $\alpha$ is the rotation angle, $[s_x, s_y]$ is the zooming scale and $[dx, dy]$ is the translation vector. According to the forms of $M_z$, $M_r$ and $M_t$, we can easily obtain the exact values of them if we know $M$.

In our work, we apply these equations on the corners in successive temporal frames to identify a text block motion status. Since the coordinates of the corners in two successive frames do not change much, we use the coordinates of the corners of every $\tau$ frame to identify motion status by deriving transform matrix $M$. For instance, let $K$ be the number of corners in a detected text and let the coordinates of the corners be $[x_k, y_k]$ ($k \in \{1, 2, ..., K\}$). After tracking them in 10 frames, their coordinates have become $[x_k', y_k']$ ($k \in \{1, 2, ..., K\}$). Importantly, we need to move the origin to the center of the text because rotation does around the origin. After coordinate conversion, the previous coordinates have became $[x_k, y_k]$ ($k \in \{1, 2, ..., K\}$) and the latter ones have become $[x_k', y_k']$ ($k \in \{1, 2, ..., K\}$). According to the theoretical analysis discussed above, we get the following equation:

$$\begin{bmatrix} x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \\ \vdots & \vdots & \vdots \\ x'_K & y'_K & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_K & y_K & 1 \end{bmatrix} \cdot M$$

in which there are 6 unknown parameters in the transform matrix $M$. Since $K$ is larger than 3, we propose to use the least square method for determining these six parameters. With the transform matrix $M$, we can derive the zooming matrix $M_z$, rotation matrix $M_r$ and translation matrix $M_t$ easily. The same transform matrices are used for identifying text motion status as follows. The ratio of $\sqrt{s_x^2 + s_y^2}$ is 1 or inverse when a text region is zoomed, or else we check another status, namely, if $\alpha$ in $M_r$ is big enough, then the text is considered as having rotated with $\alpha$ degrees, and if $\sqrt{dx^2 + dy^2}$ is large, the text is considered as having translated.

#### B. Text Tracking through Motion Status Analysis

As discussed in the previous Section, the proposed technique determines the motion status, such as linear, rotation and zoom, for the text line in the frame using KLT tracker and optical flow estimation. Here, the KLT tracker finds the correspondences in temporal frames. Then an optical flow is estimated between the correspondences. Based on the optical flow information, the proposed method derives the motion status matrix as described in Section IV.A. As a result, the search domain is reduced in tracking the text lines. For example, if the motion status is linear then the tracking algorithm searches in the linear direction rather than multiple directions in the temporal frame. In this way, motion status determination helps in reducing the search time and region for tracking the text lines, which in turn helps in finding accurate locations of the text lines. More specifically, the steps for tracking are as follows.
When we use the KLT tracker to calculate the corresponding point $v_2$ in the next frame $f_2$ of a point $v_1$ in the current frame $f_1$, we need to calculate the optical flow vector $\vec{v}$. Traditionally, $\vec{v}$ is initialized as zero: $\vec{v}^0 = [0; 0]^T$. Next an iterative scheme is applied to modify the $\vec{v}^k$ to approximate the real optical flow: $\vec{v}^k = \vec{v}^{k-1} + \vec{\eta}^k$, until the computed pixel residual $\vec{\eta}^k$ is smaller than a certain threshold, or a maximum number of iteration is reached. The detail of how to calculate $\vec{\eta}^k$ can be found in [42]. Usually, around 5 iterations are enough to reach convergence for our experiments. Let $\vec{v}$ equal to the convergence of $\vec{v}^k$. Thus, $v_2$ in frame $f_2$ is determined as $v_2 = v_1 + \vec{v}$. Once we have the motion status of the text line, the overall motion matrix $M$ for all the corners in the text line is determined. Using this, we can predict the possible location of the tracked point with $\vec{v}_1 \cdot M$, in which $\vec{v}_1$ is the extended vector of $v_1$. We use $\vec{v}_1 \cdot M$ to initialize $\vec{v}^0$ to reduce the number of iterative steps (about 2-3 steps) to reach convergence. In this way, the motion status helps in reducing time for tracking text lines in video.

V. EXPERIMENTAL RESULTS

The proposed technique consists of two major contributions, that is, text detection by spatial features and text tracking through motion status analysis on multi-oriented, multi-font, multi-sized texts in video. Further, the text detection part is divided into two sub-parts, namely, text detection in video and text detection in natural scene images. This is because when the proposed technique works for low resolution video, it should also work for high resolution scene images. We consider three video datasets for experimentation, namely, our own video dataset at the rate of 30 frames per second, which includes multi-oriented, multi-font, multi-sized graphics and scene texts, the video dataset from the recent benchmark database of ICDAR 2013 [43] which has mixed texts with less multi-orientations, and the video dataset from the standard database [7] YouTube (YVT) which is a collection of only scene texts along the horizontal direction but without graphics texts.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Total number</th>
<th>Type</th>
<th>Resolution range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our dataset</td>
<td>500</td>
<td>Video and Image</td>
<td>$320 \times 240$ to $1280 \times 720$</td>
</tr>
<tr>
<td>ICDAR 2013 Video dataset</td>
<td>15</td>
<td>Video</td>
<td>$720 \times 480$ to $1280 \times 720$</td>
</tr>
<tr>
<td>YVT</td>
<td>30</td>
<td>Video</td>
<td>Around $720 \times 1280$</td>
</tr>
<tr>
<td>ICDAR 2013 Scene dataset</td>
<td>462</td>
<td>Image</td>
<td>$307 \times 730$ to $1296 \times 960$</td>
</tr>
<tr>
<td>Microsoft data</td>
<td>307</td>
<td>Image</td>
<td>$1024 \times 1360$ to $1024 \times 768$</td>
</tr>
<tr>
<td>MSRA-TD500</td>
<td>500</td>
<td>Image</td>
<td>$1296 \times 864$ to $1920 \times 1260$</td>
</tr>
</tbody>
</table>

Similarly, we consider three benchmark scene image datasets from ICDAR family, namely, ICDAR 2013 scene text data, the Microsoft data [21] in which most of the texts are street view texts in the horizontal direction, and the MSRA-TD500 [44] data which contains horizontal and non-horizontal straight text lines. Since the resolutions vary greatly for both video and scene images, it adds more complexity to the problem. More details about the datasets are listed in Table 1. In summary, 545 videos and 1269 scene text images are considered for evaluating the performance of the proposed technique. We believe that we have considered enough diversified data to test the generic property of the proposed technique. The second part, text tracking is tested on all the three video datasets mentioned above for the purpose of evaluation.

We consider three standard measures, namely, recall, precision and F-measure along with the Average Processing Time (APT) for measuring the performance of the proposed technique. We follow the standard evaluation scheme as given in the ICDAR 2013 robust reading competition. Note that the measure proposed in [43] requires words segmentation because the ground truth is generated for words but not text lines. In our work, we detect text lines but not words because word segmentation for video is not easy for natural scene images due to low resolution and complex background. In order to use the standard measures, we combine the ground truths of words into text lines for calculating the measures. For all the experiments in this work, we calculate the measures at text lines with the same measures. Specifically, the measures are defined formally as follows. A match $m_p$ between two rectangles (detected and ground truth) is defined as the area of the intersection divided by the area of the minimum bounding box containing both rectangles. So the best match $m(r; R)$ for a rectangle $r$ in a set of rectangles $R$ is defined as

$$m(r; R) = \max\{m_p(r; r') | r' \in R\}. \tag{11}$$

This is defined to find the closest match in the set of ground truth targets for each rectangle in the set of estimates. Then Precision and Recall are defined as

$$Precision = \frac{\sum_{r \in \mathcal{T}} m(r; T_r)}{\sum_{r \in \mathcal{E}} |E_r|} \quad Recall = \frac{\sum_{r \in \mathcal{T}} m(r; E_r)}{|\mathcal{T}|} \tag{12}$$

where $\mathcal{T}$ and $\mathcal{E}$ are the sets of targets (ground truth) and estimated boxes, respectively. These two measures are combined to form a single measure $f$ with a parameter $\alpha$. We set it 0.5 to give precision and recall an equal weight:

$$f = \frac{1}{\alpha \cdot Precision + \alpha \cdot Recall} \tag{13}$$

In order to show the effectiveness of the proposed technique, we compare it with the state of the art methods, namely, Li et al.’s method [18] which uses moments and wavelet combination for both text detection and tracking in video, Zhao et al.’s method [36] which uses corner based features and optical flow for both text detection and caption text tracking in video, Mosleh et al.’s method [38] which uses stroke width transform and inpainting for both text detection and text removal, Epshtein et al.’s method [21] which proposes stroke width transform for text detection from natural scene images, Yao et al.’s method [44] which uses an improved version of stroke width transform for non-horizontal text detection from natural scene images, and finally Yin et al.’s method [45] which uses Maximally Stable Extremal Regions (MSER) and single link clustering algorithms for non-horizontal text detection from natural scene images. Since the first three methods work for both text detection and text tracking, we compare these methods for all the experiments. On the other hand, since Epshtein et al.’s method [21] is for text detection from natural scene images but not video, we compare this method for text detection in video, as well as natural scene images but not for text tracking experiments. The results of the last two methods are reported for MSRA-TD500 dataset at the text line level, we compare these two methods on MSRA data but not all the other experiments since our technique requires text lines for experimentation.

The values for the thresholds $NT$ and $ST$ are determined empirically based on the experiments on the ICDAR 2013 training dataset as follows.

1. Initialize $nt_0 = 4$.
2. Fix $NT$ as $nt_{k-1}$ and range $ST$ from 1 to 2, and then run our text detection method on the ICDAR 2013 training dataset.
for each ST. Set ST equals to st_k where the Recall equals to Precision. Then fix ST as st_k and range NT from 0 to 10. Set NT equals to nt_k where Recall equals to Precision. Fig. 15 shows the illustration of finding st_1 and nt_1.

3. Repeat step 2 until |nt_k - nt_k-1| < Δnt and |st_k - st_k-1| < Δst.

The illustration of determining nt_k and st_k is shown in Fig. 15, where the recall and precision curves intersect at some values. At last, the experimental study helps in setting the values as NT = 6 and ST = 1.45.

![Fig.15: The illustration of determining nt_k and st_k. (a) Set nt_0 equals to 4 and range ST from 1 to 3, then we get st_1 = 1.45. (b) Set ST equals to st_1 and range NT from 0 to 10, then we get nt_1 = 6.]

**A. Evaluation of Text Detection in Video**

As mentioned in the previous section, we consider three video datasets for experimentation, which are our own dataset, the ICDAR 2013 video dataset and the YVT video dataset. Since all the existing methods [18, 36, 38] detect texts in a key frame, the first frame, or individual scene images, we follow the same way for text detection in this work. For tracking, we use temporal frames after detecting texts in a key frame.

**a. Experiments on Our Video Data**

Since our dataset contains 500 videos which lasts for 1 or 2 seconds, we extract 500 key frames for text detection in this experiment. Our main objective is to achieve good results for both text detection and tracking, we consider the first frame as the keyframe for text detection and the same text line would be traced in temporal frames. This helps us to speed up the text detection process by skipping duplicate frames. Therefore, 500 frames correspond to 500 videos for experimentation in this work. This dataset includes a wide range of texts that are multi-oriented, multi-sized, multi-fonts, and multi-scripts of both graphics and scene texts. Therefore, this dataset can be considered as a complex dataset for testing the capability of generalization of the proposed technique. Sample qualitative results are shown in Fig. 16, where one can see that the proposed technique detects almost all the texts with proper bounding boxes for different text lines including scene texts and graphics texts. Fig. 16 also shows that the proposed technique is good at detecting multi-script text lines. The quantitative results of the proposed and the existing methods are reported in Table 2, where the proposed technique gives better recall and F-measure than the existing methods. Note that Li et al.’s method is good at recall and a short average processing time but is worse at precision compared to the proposed technique because their performance depends on the number of samples used for training and hence the method produces more false positives. It is faster than all the other existing methods including the proposed technique. Mosleh et al. and Zhao et al. are good for detecting caption texts but not for scene texts because their scope is caption text detection. Therefore, low precision, recall and F-measure are reported for both techniques compared to the proposed technique. On the other hand, the proposed technique reports good results in terms of precision, recall, F-measure and processing time because of its ability to handle both graphics and scene texts of different orientations.

![Fig.16: Sample qualitative results of the proposed technique for our video data]

**Table 2: Performance of text detection on our video data**

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>APT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method</td>
<td>0.70</td>
<td>0.79</td>
<td>0.74</td>
<td>1.3</td>
</tr>
<tr>
<td>Mosleh et al. [38]</td>
<td>0.56</td>
<td>0.50</td>
<td>0.53</td>
<td>1.3</td>
</tr>
<tr>
<td>Epstein et al. [21]</td>
<td>0.41</td>
<td>0.46</td>
<td>0.43</td>
<td>1.0</td>
</tr>
<tr>
<td>Li et al. [18]</td>
<td>0.10</td>
<td>0.83</td>
<td>0.18</td>
<td>0.8</td>
</tr>
<tr>
<td>Zhao et al. [36]</td>
<td>0.31</td>
<td>0.16</td>
<td>0.21</td>
<td>3.4</td>
</tr>
</tbody>
</table>

**b. Experiments on ICDAR 2013 Video Data**

Since this dataset provides 15 videos lasting from around 10 seconds to 1 minute, we extract 50 key frames from the video sequences for experimentation. This dataset represents a wide range of real life situations using different types of cameras. In addition, this dataset includes texts in various languages such as Spanish, French and English. Most of the video captures indoor and outdoor scenes to represent real life applications. Therefore, this dataset does not contain graphics texts. Since the ground truth is available at the word level for texts in frames, we use the same ground truth for calculating the measures by combining the ground truths of words into text lines. The same evaluation scheme as mentioned in [43] is used for evaluation. Sample qualitative results of the proposed technique are shown in Fig. 17, where we can see that the proposed technique detects well for complex background texts except for the last frame. This shows that the proposed technique is capable of handling texts embedded in indoor and outdoor frames. The quantitative results of the proposed and the existing methods are reported in Table 3, where it is noticed that the proposed technique gives better results than all the existing methods in terms of recall, precision and F-measure. However, Li et al.’s method is the best in the average processing time among all the methods. Table 3 shows poor precision, recall and F-measure for the existing methods compared to the proposed technique because the existing methods are developed for graphics and caption text detection in video but not scene text in video. At the same time, this dataset contains a majority of scene texts with complex background and low resolution of different script types.
Therefore, this is also a complex dataset for detecting texts. Each video has 15 seconds in duration. From this video dataset, we extract 35 key frames for text detection in this experiment. Sample qualitative results of the proposed technique are shown in Fig. 18, where the proposed technique detects well for almost all the frames except the last frame. Similarly, the quantitative results of the proposed and the existing methods are reported in Table 4, where the proposed technique gives better results than the existing methods in terms of recall, precision and F-measure but APT is the best for Li et al.’s technique. The main reason for poor precision, recall and F-measure for the existing methods is that this dataset does not contain any graphics and caption text. Though Li et al.’s and Epshtein et al. techniques work well for scene texts, they report poor precision, recall and F-measure compared to the proposed technique because Li et al.’s technique requires training samples and Epshtein et al.’s technique depends on Canny edge image of each input image. On the other hand, though the proposed technique uses Canny edge image, it selects potential text candidates successfully at different scales by eliminating unwanted edge information with the help of symmetry features. Therefore, the proposed technique outperforms the existing techniques.

In summary, when we look at the results of the proposed technique reported in Table 2-Table 4 for our video, the ICDAR 2013 video and YVT datasets, the proposed technique gives better results for the YVT data but poorer results for ICDAR 2013 video data compared to that for our data. Since YVT does not contain mixed texts of caption and scene, and most of the texts are in the horizontal direction, the proposed technique gives better results. In case of the ICDAR 2013 video data, they are small texts with mixture of different resolutions captured by various devices such as mobile cameras, a Head mounted camera and a hand held camcorder, etc. In addition, this dataset contains texts of different languages while our data consists of multi-oriented captions and scene texts.

B. Evaluation of Text Detection in Natural Scene Images

The experimental results provided in Section V.A.4 shows that the proposed technique works well for video of low resolution and complex background. The same technique is expected to work well for the texts in natural scene images captured by a high resolution camera.

Since the proposed technique and the existing technique use single frame/image for text detection, we use the same way to detect texts in natural scene images. We use the ground truth at the text line level for calculating recall, precision and F-measures by combining the ground truths of words into text lines. There are existing techniques which report results at the word level using the ground truth. Since our method focuses on text line detection in video as well scene images, the implemented existing techniques are tested at the text line level for a fair comparative study. The techniques which report results at the word level are not considered for comparative study in this work.

a. Experiments on ICDAR 2013 Scene Text Data

Sample qualitative results of the proposed technique are shown in Fig. 19, where it is found that the proposed technique works well for scene text detection. The results of the proposed and the existing techniques are reported in Table 5, which shows that the proposed technique gives the best recall and F-measure among all the techniques under study. Since the technique proposed by Epshtein et al. is developed for scene text detection, it gives slightly lower precision, recall and F-measure compared to the proposed technique. In the same way, Zhao et al.’s technique gives the lowest precision, recall and F-measure as their technique is designed for caption texts in video but not scene texts in scene images. The same is true for Li et al.’s technique. However, Mosleh et al.’s technique gives the best at precision, recall and F-measure among the existing techniques (other than the proposed technique) because their technique considers both caption and scene texts in video.
The quantitative results of the proposed technique are low compared to those of existing techniques. However, in this experiment, we calculate precision instead of recall and F-measure. The proposed technique detects text in images better than those two existing methods because all the existing techniques except Yin et al. and Yao et al. focus on horizontal text detection in scene images but not multi-oriented text detection. Note that we compare our technique with Yin et al. and Yao et al. only for this dataset because these techniques report results at the text line level as our technique. Table 7 shows though the precision of the proposed technique is almost the same as Yin et al. and Yao et al., overall, the performance of the proposed technique is better than those two techniques especially in terms of recall. This is all because of the spatial and the symmetry features for identifying potential text candidates at multi-levels.

**Experiments on Microsoft Data**

This dataset is much harder than the ICDAR 2013 scene dataset due to the presence of vegetation and repeating patterns such as windows, which are virtually indistinguishable from texts without OCR. In addition, the text fonts in it are too small in most images. Achieving a good accuracy for this method is challenging. Sample results of the proposed technique are shown in Fig. 20, where the proposed technique detects texts correctly for a few images and incorrectly for some others. The quantitative results of the proposed and the existing techniques are reported in Table 6, which shows that the proposed technique is high at recall but low at precision compared to the existing techniques due to the presence of objects in the background which look like texts. As a result, false positives are high and hence precisions are low. However, overall, F-measure is high for the proposed technique compared to the existing techniques. Epshtein et al.’s method also reports low recall and F-measure due to the same problem. Note that the values of the measures of Epshtein et al. reported in their paper is a little different from the values of the measures reported in Table 6 because in this experiment, we calculate the measures at the text line level while the results reported in their paper is at the word level. The method proposed by Mosleh et al. is nearly close to Epshtein et al.’s technique because Mosleh et al.’s technique is an improved version of that proposed by Epshtein et al. Li et al.’s and Zhao et al.’s methods are much worse among the techniques in this study.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>APT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method</td>
<td>0.38</td>
<td>0.71</td>
<td>0.50</td>
<td>1.2</td>
</tr>
<tr>
<td>Mosleh et al.</td>
<td>0.48</td>
<td>0.51</td>
<td>0.49</td>
<td>1.1</td>
</tr>
<tr>
<td>Epshtein et al.</td>
<td>0.51</td>
<td>0.50</td>
<td>0.51</td>
<td>1.0</td>
</tr>
<tr>
<td>Li et al.</td>
<td>0.23</td>
<td>0.35</td>
<td>0.28</td>
<td>0.7</td>
</tr>
<tr>
<td>Zhao et al.</td>
<td>0.33</td>
<td>0.76</td>
<td>0.46</td>
<td>3.1</td>
</tr>
</tbody>
</table>

**Table 6: Performance of text detection on Microsoft scene text data**

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>APT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method</td>
<td>0.63</td>
<td>0.70</td>
<td>0.66</td>
<td>1.2</td>
</tr>
<tr>
<td>Mosleh et al.</td>
<td>0.56</td>
<td>0.53</td>
<td>0.55</td>
<td>1.2</td>
</tr>
<tr>
<td>Epshtein et al.</td>
<td>0.52</td>
<td>0.50</td>
<td>0.51</td>
<td>1.0</td>
</tr>
<tr>
<td>Li et al.</td>
<td>0.26</td>
<td>0.65</td>
<td>0.37</td>
<td>0.8</td>
</tr>
<tr>
<td>Zhao et al.</td>
<td>0.34</td>
<td>0.69</td>
<td>0.46</td>
<td>3.3</td>
</tr>
<tr>
<td>Yin et al.</td>
<td>0.61</td>
<td>0.71</td>
<td>0.66</td>
<td>0.8</td>
</tr>
</tbody>
</table>

**Table 7: Performance of text detection on MSRA-TD500 multi-oriented scene text data**
d. Experiments on Blurred Video Frames

It is known that while capturing video, there are chances of introducing blurring effects due to object movements or camera movements. Therefore, in this work, we test the proposed technique on 50 blurred frames chosen from ICDAR 2013 video and scene data [43] to study the effectiveness of the proposed technique. The sample results of the proposed technique are shown in Fig. 22, where the proposed technique detects text lines well for the first four images. However, for the last two images, the proposed technique does not detect text lines completely because of the blurring effects. This is because blurring affects Canny edge, gradient calculation and corner detection steps. The results of the proposed and the existing techniques are reported in Table 8, where we can notice that the proposed technique is better than the existing techniques in terms of precision, F-measure and the average processing time. However, Mosleh et al. is the best at recall compared to the other techniques including the proposed technique. The poor result of the existing techniques is mainly because the techniques were developed for good images but not blurred images. When we compare the results of the blurred data and other data, the proposed technique gives slightly lower results in terms of F-measure for the blurred data. Therefore, there is a scope for improvement to withstand blurring without sacrificing accuracy.

Fig.22: Sample qualitative results of the proposed technique for Blur video and scene text data

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>APT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method</td>
<td>0.63</td>
<td>0.28</td>
<td>0.39</td>
<td>1.3</td>
</tr>
<tr>
<td>Mosleh et al. [38]</td>
<td>0.54</td>
<td>0.30</td>
<td>0.39</td>
<td>1.3</td>
</tr>
<tr>
<td>Epshtein et al. [21]</td>
<td>0.53</td>
<td>0.29</td>
<td>0.37</td>
<td>1.0</td>
</tr>
<tr>
<td>Li et al. [18]</td>
<td>0.10</td>
<td>0.23</td>
<td>0.14</td>
<td>0.8</td>
</tr>
<tr>
<td>Zhao et al. [36]</td>
<td>0.13</td>
<td>0.22</td>
<td>0.16</td>
<td>3.4</td>
</tr>
</tbody>
</table>

C. Evaluation of Text Tracking in Video

We consider three video datasets mentioned in Section V.A for evaluating the proposed tracking step. First, the proposed technique uses text detection step for detecting texts in the first frame and then tracks the same texts in temporal frames using the KLT tracker. Three existing techniques, namely, Mosleh et al., Li et al. and Zhao et al. are used for comparative study because these techniques involve text tracking in video. Besides, we also compare our tracking results with the tracking results reported in the ICDAR 2013 competition [43]. In this work, we consider tracking texts of different motion status, that is, video texts with linear motion, zoom and rotation (non-linear motion). This is different from the existing techniques since they generally focus on tracking texts with linear movements. Sample results for text tracking of the proposed technique are shown in Fig. 23, where we can see in (a)-(d), the proposed technique tracks video texts successfully for the different motion status chosen from the three video datasets.

Before successfully tracking texts, the proposed technique finds the motion status of each text, namely, static, linear, zoom and rotation using the derived transformation matrices as presented in Section IV. With the help of motion status identification, the proposed technique achieves good results for text tracking because motion status identification helps in modifying the parameters for the text detection step and the tracker. The quantitative results of the proposed and the existing techniques are reported in Table 9, where one can see that the proposed technique outperforms the existing techniques. Table 9 shows that the proposed and the existing techniques give good results in terms of precision, recall and F-measure for the texts of linear motion but lower precision, recall and F-measure for the texts with zoom in, out and rotation because tracking a text when it is moving with rotation, zoom in and out changes the shape of the text. This difficulty leads to poor performances for the existing techniques as those techniques are not meant for handling such motions.

The final performance of the proposed and the existing techniques for tracking in the three video datasets is reported in Table 10, where it can be seen that the proposed technique outperforms the existing methods for tracking. Zhao et al. use the KLT tracker as in our technique but their technique is more sensitive to corners due to background influence compared to our technique. The tracking performance of Li et al.’s technique depends on text detection results but does not consider text movements in complex background. Mosleh et al.’s technique uses Camshift for text tracking. This is good for tracking texts with linear motion but not texts with different rotations. In Table 10, FPS denotes frames per second used for tracking texts. In summary, since the proposed technique tracks video texts according to motion status identification while the existing techniques use some tools for tracking which are sensitive to text movements, the proposed technique is better than the existing techniques and works for different situations in the sense that the proposed technique is independent of language, transformation, font and font size variations.

Table 8: Performance of text detection on blurred data

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Our Method</th>
<th>Mosleh et al.</th>
<th>Zhao et al.</th>
<th>Li et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>98</td>
<td>96</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>Zooming</td>
<td>93</td>
<td>92</td>
<td>92</td>
<td>34</td>
</tr>
<tr>
<td>Rotation</td>
<td>90</td>
<td>91</td>
<td>90</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 9: Performance of the proposed and existing techniques for tracking text in video with respect to motion status (P: Precision, R: Recall and F: F-measure)

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Our method</th>
<th>Mosleh et al.</th>
<th>Zhao et al.</th>
<th>Li et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICDAR</td>
<td>89</td>
<td>91</td>
<td>90</td>
<td>51</td>
</tr>
<tr>
<td>YVT</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>45</td>
</tr>
<tr>
<td>Average</td>
<td>94</td>
<td>93</td>
<td>93</td>
<td>41</td>
</tr>
</tbody>
</table>
We also compare the text tracking results of the proposed technique with the tracking results given by the baseline algorithm (ABBAY OCR SDK) and TextSpotter as reported in [43] with the same measures to study the effectiveness of the proposed technique. In this experiment, we use the performance measures, such as MOTP which is the Mean Tracking Precision, MOTA which is the Mean Tracking Accuracy, and ATA which is the Average Tracking Accuracy to evaluate the tracking results given by the techniques. More details about the formulas and the definitions for these three performance measures can be found in [43]. Table 11 shows that the proposed techniques is good at MOTA compared to TextSpotter, while TextSpotter is good at MOTP. However, the proposed technique is better than the baseline algorithms in terms of MOTA and ATA. The main cause for the poor MOTP for the proposed technique compared to the existing techniques is that the proposed technique considers the first frame as the keyframe for text detection and the same text line is to be tracked in other frames. However, this experiment does not consider keyframe selection for text detection, but rather it detects the lines as text appears in the whole video. Since our goal is to achieve good results for both text detection and tracking, we prefer to use keyframe for text detection and then tracking the same text line, but TextSpotter is developed solely for text tracking. As a result, this experiment creates discrepancy between the text lines detected by the proposed technique and its ground truth. Therefore, there is a scope for testing our method on live video in real time environment without losing accuracy.

Table 11: Performance of the proposed and existing techniques on tracking data of ICDAR 2013 video

<table>
<thead>
<tr>
<th>Method</th>
<th>MOTP</th>
<th>MOTA</th>
<th>ATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method</td>
<td>0.61</td>
<td>0.46</td>
<td>0.29</td>
</tr>
<tr>
<td>Mosleh et al. [38]</td>
<td>0.45</td>
<td>0.13</td>
<td>0.03</td>
</tr>
<tr>
<td>Li et al. [18]</td>
<td>0.21</td>
<td>0.15</td>
<td>0.07</td>
</tr>
<tr>
<td>Zhao et al. [36]</td>
<td>0.24</td>
<td>0.11</td>
<td>0.05</td>
</tr>
<tr>
<td>TextSpotter [43]</td>
<td>0.67</td>
<td>0.27</td>
<td>0.12</td>
</tr>
<tr>
<td>Baseline Algorithm [43]</td>
<td>0.63</td>
<td>-0.09</td>
<td>0</td>
</tr>
</tbody>
</table>

VI. CONCLUSION AND FUTURE WORK

This paper proposes a novel technique for detecting and tracking texts of any orientation in video. The technique explores gradient directional information at the component level and spatio-temporal information for text candidate selection. The Delaunay triangulation is proposed to study the spatial relationship between corners in a different way. The characteristics of video texts are proposed to eliminate false text candidates. Then grouping is proposed for extracting text regions of any orientation based on the nearest neighbor criterion and text direction. To tackle video texts of multi-font or multi-sized fonts, we further propose multi-scale integration for full text line detection. Then the detected text is tracked in video by matching sub-graphs of triangulation. Experimental results for text detection and tracking on our video dataset, the benchmark video datasets and the natural scene image datasets show that the proposed technique is superior to the state of the art techniques in terms of recall, precision and F-measure. Since the reported accuracy is a bit low compared to the accuracy of document analysis, we are planning to further investigate the proposed technique to improve the accuracy. In addition, our next work would be text recognition by considering text detection results as inputs.

REFERENCES


Fig.23: Sample qualitative results of the proposed technique for txt tacking in video with different motion status