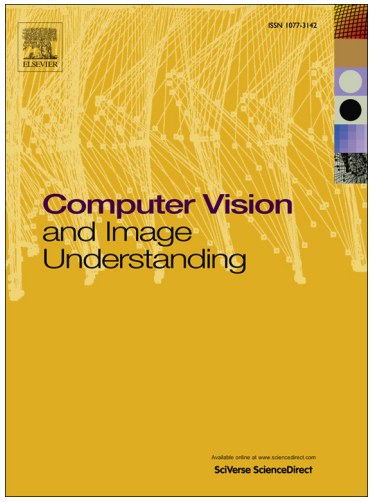


New Gradient-Spatial-Structural Features for Video Script Identification

Palaiahnakote Shivakumara, Zehuan Yuan, Danni Zhao, Tong Lu, Chew Lim Tan

PII: S1077-3142(14)00185-4
DOI: http://dx.doi.org/10.1016/j.cviu.2014.09.003
Reference: YCVIU 2175

To appear in: *Computer Vision and Image Understanding*

Received Date: 11 December 2013
Accepted Date: 8 September 2014

Please cite this article as: P. Shivakumara, Z. Yuan, D. Zhao, T. Lu, C.L. Tan, New Gradient-Spatial-Structural Features for Video Script Identification, *Computer Vision and Image Understanding* (2014), doi: http://dx.doi.org/10.1016/j.cviu.2014.09.003

New Gradient-Spatial-Structural Features for Video Script Identification

[a]Palaiahnakote Shivakumara, [b]Zehuan Yuan, [c]Danni Zhao, [b]Tong Lu and [c]Chew Lim Tan

[a]Faculty of Computer Science, University of Malaya, Kuala Lumpur, Malaysia

[b]National Key Lab for Novel Software Technology, Nanjing University, Nanjing, China

[c]Department of Computer Science, School of Computing, National University of Singapore

[a]hudempsk@yahoo.com, [b]zhyuan001@gmail.com, [b]lutong@nju.edu.cn, [c]tancl@comp.nus.edu.sg

Abstract

Multi-script identification helps in automatically selecting an appropriate OCR when video has several scripts; however, script identification in video frames is challenging because low resolution and complex background of video often cause disconnections or the loss of text information. This paper presents a novel idea that integrates the Gradient-Spatial-Features (GSpF) and the Gradient-Structural-Features (GStF) at block level based on an error factor and the weights of the features to identify six video scripts, namely, Arabic, Chinese, English, Japanese, Korean and Tamil. Horizontal and vertical gradient values are first computed for each text block to increase the contrast of text pixels. Then the method divides the horizontal and the vertical gradient blocks into two equal parts at the centroid in the horizontal direction. Histogram operation on each part is performed to select dominant text pixels from respective subparts of the horizontal and the vertical gradient blocks, which results in text components. After extracting GSpF and GStF from the text components, we finally propose to integrate the spatial and the structural features based on end points, intersection points, junction points and straightness of the skeleton of text components in a novel way to identify the scripts. The method is evaluated on 970 video frames of six scripts which involves font, font size or contrast variations, and is compared with an existing method in terms of classification rate. Experimental results show that the proposed method achieves 83.0% average classification rate for video script identification. The method is also evaluated by testing on noisy images and scanned low resolution documents, illustrating the robustness and the extensibility of the proposed gradient-spatial-structural features.

Keywords- *Video text blocks, Gradient blocks, Dominant video text pixels, Gradient-Spatial-structural-features, Video script identification*

1. Introduction

Due to the advancement of IT technology and the increase in the usage of video that is delivered via TV broadcasting, Internet and wireless networks, the size of video database is increasing drastically nowadays [1-4]. In order to enable users to locate their interested content in such enormous quantity of video data quickly, there must be powerful indexing, retrieval and summarization techniques. In this regard, the current methods proposed in the field of content based image retrieval are not much

effective for a large video database especially for retrieving and labeling video events, which requires high level semantics but not only low level features [5]. Since content based image retrieval methods are not effective in bridging the gap between high level semantics and low level features, text detection, script identification and recognition have come into play with the objective of filling the gap with the help of Optical Character Recognizer (OCR). In this way, text detection and recognition helps in generating meaning that is close to the content of video. Thus, text information can be used effectively for video labeling or events retrieval [1-5].

Text detection and recognition is a familiar problem for the document analysis community where researchers detect and extract text information from scanned document images with a high resolution. Since a document image contains plane or homogenous background with a high resolution, text detection based on connected component analysis is a successful technique in this field [6, 7]. However, the same method cannot be used for detecting and extracting text from natural scene images because of complex background, font, font size variations and color bleeding [4, 7].

To solve this problem, several methods [8-13] have been proposed in the literature based on bottom up analysis or region segmentation. However, these methods still enjoy the high resolution of text which gives high clarity and visibility compared to background. These methods expect directly or indirectly the complete shape of the character or consider the character as one component. Hence the methods depend much on connected component analysis. Thus, these methods may not be used directly on video for text detection and recognition because video suffers from both low resolution and complex background, which often cause disconnections, the loss of information, distortion, etc. It is evident that these methods [14, 15] have so far achieved the character recognition rates only between 60% to 72%. This poor accuracy is due to the complex background, for example, usually sports video contains scene texts embedded in complex background with greenery, buildings, advertisements, etc. In addition, video usually contains two types of texts: graphics text which is superimposed and scene text which is a part of the image. Since graphics text is the edited one, it is easy to process and we can expect good clarity and visibility, whereas scene text is unpredictable due to the variations in its characteristics. Further, graphic text is useful for news video analysis while scene text is useful for navigation applications, such as assisting blind people, vehicle tracking, assisting tourists, sports events extraction, and exciting events extraction. [1-4]. The presence of both such texts in video makes the problem more complex and challenging compared to text detection and recognition from either natural scene images or scanned images. In the same way, experimental results shows that when we apply conventional character recognition methods directly on video, the character recognition rate is achieved typically from 0% to 45% [16, 17], which is much lower than the recognition rate of scene text in natural scene images. The poor accuracy is because of the presence of both graphics and scene texts, low resolution and complex background. Note that although we can see lots of cameras with a high resolution for capturing images and video, low resolution cameras are still required in many real life applications such as mobile services and surveillance systems because of the shortage of memory,

battery power, computation power or even operating cost that limits to capture relatively low resolution [18, 19]. Therefore, text extraction and recognition from video is still challenging and thus has become an emerging area for researchers.

To address the problems of video, several methods have been proposed in the past years. These methods can be classified widely as follows: connected component-based [6, 20], texture-based [21-24], and gradient or edge based methods [25-28]. Although these methods achieve a good accuracy for text detection or text block detection irrespective of fonts, font sizes, types of text and orientations, they do not have the ability to differentiate multi-script frames in video because the goal of these methods was to detect text block or text region regardless of scripts.

Similarly, we can see scene text recognition from natural scene images [29-31] and from video [16, 17, 32, 33], which takes the output of text detection, such as text region, blocks, lines, words and characters as the input for recognition. Generally, these methods take care of segmentation and binarization of text areas/blocks with the help of enhancement criteria to increase the contrast of text lines before feeding them to OCR [34]. Most of the methods either use publicly available OCR for binarization or their own classifier to recognize text in video. However, these methods are limited to a single script frame in video/image but not video containing multi-scripts because the extracted features and OCR are usually designed for a specific language. In addition, there is no universal OCR for recognizing different scripts in video since it is a hard task. Therefore, without script identification that helps automatically choosing appropriate OCR, there will be a big gap between text detection and recognition. Thus script identification is essential when the environment is of multi-script and multi-lingual like in Singapore, Malaysia and India, where we can see more than two official languages.

Script identification for a given document image with plain background at block level, text line level and word level in the field of document analysis and recognition has attained high recognition rates [35-38]. In contrast to document images, script identification in video frame is a new topic and is challenging as it suffers from low resolution, complex background and font variation as mentioned above. Besides, the presence of multi-scripts in a video frame makes script identification and recognition more complex and challenging for researchers. As noted from the methods on script identification in document images [39, 40], Japanese, Korean, Chinese and Tamil scripts worsen the performances of the methods in terms of classification rate because the scripts share common features. Therefore, there is a great demand for developing a new method for automatically identifying scripts in video frames irrespective of text orientation, font variation, font size variation and contrast. In this work, we consider six scripts, namely, Arabic, Chinese, English, Japanese, Korean and Tamil since these languages represent a wide range of users in the world and are very popular.

2. Related Work

Identification of different scripts in a document with plain background and a high resolution is a familiar problem in document analysis. In this regard, we can see a lot of methods in the literature. An

overview of script identification methodologies based on structure and visual appearance is presented in [35]. It is noted from this review that the surveyed methods work well for camera-based images but not for video frames since the latter has a low contrast and complex background, and the surveyed methods are developed specifically for camera based documents. The rotation invariant features for automatic script identification are proposed by Tan [36] based on Gabor filter. This work considers six scripts for identification. Since texture feature is sensitive to font or font size variations, the method may not work well for video text script identification. Busch et al. [37] have explored the combination of wavelet and Gabor features to identify scripts. However, these approaches expect a large number of training samples to achieve a good classification rate. In addition, the number of samples for training limits the ability to work on more scripts. Lu and Tan [38] have proposed a method for script identification in noisy and degraded document images based on document vectorization. This method considers the features based on zonalization and connected component analysis because its target is camera based documents but not video images. Moreover, the use of coding in the method is sensitive to disconnections and complex background. Although the method is tolerant to various types of document degradations, it does not perform well for Tamil because of the complexity of the script. Texture features based on Gabor filter and Discrete Cosine Transform are used for script identification at word level in many methods [41-43], which expect high contrast documents for the segmentation of words and rely on connected component analysis to achieve a good accuracy. However, connected component analysis requires the knowledge of the shape of a character. Extracting the exact shape of a character is hard in case of video scripts. Similarly, the studies on character shape for identifying scripts are proposed in [44, 45]. These methods perform well as long as segmentation works well and character shape is preserved. Since scanned documents and camera based images have a high resolution, the shape of a character can be preserved with binarization methods. However, the same is not true for video because of low resolution, complex background and the existence of scene text. Online script identification is addressed in [46], where spatial and temporal information is used to recognize words and text lines. Online recognition is easier than offline recognition because online features can easily extract strokes without much noise and distortion as in offline documents. Recently, composite script identification and orientation detection for Indian text images have been proposed by Ghosh and Chaudhuri [47]. This method considers eleven scripts for identification purpose. The features presented in this work are derived from the connected component analysis. These features are good only if the connected components preserve their shapes; however, it is not clear how it works for video script lines. Bashir and Quadri [48] proposed a method for identification of Kashmir script in a bilingual document image using profile based features. As we know that horizontal and vertical projection profile based features work well for document images with plane background, and are sensitive to noise and background. Therefore, this method may not be suitable for video script identification. Rani et al. [49] proposed a method for script identification of pre-segmented multi-font characters and digits using texture feature and gradient features with the SVM classifier. This method

first segments characters from a document and then extracts features on character level for identifying scripts. Texture feature is good when we give more characters, say at least words. This method aims at script identification from a scanned document with simple background. Ferrer et al. [50] explored Local Binary Pattern (LBP) for script identification at line level. The LBP features are extracted based on zonalization of text line. Zonalization or dividing the whole text line into blocks is good for scanned document images with plane background but not for the text line with complex background where it is hard to find zones. The SVM classifier has been used for identification, but the use of a classifier probably limits the flexibility of the method.

Based on the literature review on script identification in both scanned document images and camera based images, it is observed that most of the methods used Roman and Devanagiri as the common scripts and a few methods consider other scripts for identification purpose [43]. In addition, the main focus of these methods is that the identification of the scripts in documents with plain background and a high contrast but not the scripts in video. To the best of our knowledge, the work on video script identification is not well reported in the literature. For instance, a method which is relevant to our work, takes the detected text lines as the input and uses statistical and texture features with the k-nearest neighbor classifier to identify Latin and Ideographic texts in images and videos [51]. Since the appearance patterns of English and Chinese scripts differ, the method proposes statistical features at line level, and extracts the features based on zonalization. This zonalization is good for captions and high contrast text lines, but not for the video text lines that are embedded in complex background or have a low contrast. Therefore, this method works well for English and Chinese but not other scripts. In addition, its performance depends on samples training by the classifier and zonalization. Similarly, recently, another method for video script identification has been proposed at word level using Zernike moments, Gabor and gradient features [52]. This method extracts features based on the orientation of Gabor filter and gradients, hence it is sensitive to rotation, different resolutions and scaling. The method considers only three scripts, namely, Bengali, English and Hindi. In summary, the method extracts a huge number of features and passes those features to the SVM classifier to identify the scripts and to improve the classification rate. However, the method cannot be used for identifying other scripts directly and its performance depends on the segmentation of words. It is evident from the method proposed in [53] for script identification at word level where the method explores gradient angular features. Experimental results show that segmenting words from Chinese, Japanese and Korean scripts is hard because of the uncertainty in defining space between the words and the characters compared to Arabic, English and Tamil. Therefore, they manually segment the words for script identification. Hence, we can conclude that the segmentation is language dependent.

We have proposed new features, namely, smoothness and cursiveness at text lines without classifiers for video script identification [54]. The smoothness and cursiveness are studied based on the angle information determined by Principal Component Analysis (PCA) through giving coordinates of thinned text line at different zones. Finally, the K-NN with K=1 classifier is used for script

identification. This method considers only English, Chinese and Tamil scripts and it is noted from the experimental results that the features are not good enough to handle more scripts present in video frames. To overcome the problem reported in [54], we have also proposed a method based on Spatial-Gradient-Features for the identification of six scripts at frame level in [55], in which it was shown that the gradient-spatial features have the ability to discriminate six scripts. It was a preliminary work to test the idea for identifying scripts using spatial domain based features. However, it was shown through the experimental results that spatial based features alone were not sufficient to achieve a good accuracy for a large database. To overcome this limitation, in this work, we extend the method to integrate Gradient-Spatial-Features with Gradient-Structural-Features in a novel way by calculating weights based templates for identifying the six scripts to achieve a better accuracy than the work presented in [55]. In addition to this, it is shown that this soft integration method can be extended to other images like natural scenes and scanned documents in addition to robust and noise to some extent. Thus, the current work is more comprehensive and general comparing to the existing method [55].

In summary, this work aims at solving script identification in video by proposing a new soft integration of gradient-spatial-structural-features. The novelty of the proposed method lies in the following unique aspects in comparison with the existing approaches: (i) The method does not rely any segmentation algorithm to segment text lines, words and characters (local information) to identify the script. Instead it uses blocks (global information) that contain a few words of different fonts, font size, contrast and orientation. As it is noted from the method [53], segmentation is not a smooth operation for the scripts like Chinese, Japanese and Korean due to the uncertainty in defining the space between words. (ii) The method uses a new way of using horizontal and vertical gradient information to extract candidate text components, which will help in enhancing the low contrast text information that is challenging as shown in [55]. (iii) The method uses the features based on end points, intersection points and junction points which are invariant to rotation, scaling and translation, and hence work well for low contrast video images. (iv) The method integrates spatial and structural features with the flexibility of changing weights determination, allowing the proposed method to be extended to more scripts or other datasets with few modifications. In other words, despite the fact that the proposed features are low-level image representations, the ways we extract the features, such as the spatial features with dominant points, the structural features with flexible rules, and the soft integration with flexible weights determination as mentioned above, behave like high-level image representations. This is the insight and the strength of the proposed work.

3. The Proposed Methodology

As discussed in the introduction section, we can find several sophisticated methods for text block detection and text region detection in video frames in the literature. These methods work well for texts from arbitrary orientations, low contrast texts, complex background and more important multi-scripts frames in video. For this work, we propose to use our earlier text detection method [56] to detect text

blocks from video frames. This method explores wavelet-moments features with mutual nearest neighbor clustering to identify the blocks. According to the results reported in [56], the method gives a good precision for text block classification. We are inspired by the work presented in [57-59] for text recognition from natural scene images, where the authors propose a multiple hypotheses framework for text detection and recognition without segmenting text lines, words and characters. Instead, they try to use multiple hypotheses at component level for recognizing text in video. The main reason to recognize text without segmentation is that recognition accuracy should not rely on a segmentation method, namely, independent of segmentation errors because segmenting words or text lines exactly is hard from natural scene images. The same thing is true for video, which is much more difficult compared to natural scene text images. Therefore, in this work, we use a method that detects text at block level but not at text line or word level. The method detects text blocks in video frames irrespective of font variations, font size, background and orientation variations, and most importantly, it detects texts without discriminating different scripts in video frames. In other words, the text detection method detects almost all the texts blocks in the input frame regardless of scripts and hence if we combine all the text blocks, the result is as good as text detection results of the whole input frame. If any text block is missed then it can be restored with the help of boundary growing and the edge image of the input frame as proposed in [60], which detects text blocks and then integrates all the text blocks to get exact text lines. The proposed angle projection growing boundary restores text information at line level if any text is missed. Therefore, the current work aims to identify the scripts for the given input text blocks of different scripts. This method splits the whole 256×256 video frame into 16 blocks of size 64×64, and studies wavelet-moments and the mutual nearest neighbor concept as mentioned in [56] to identify text blocks among the 16 blocks. We expect that the divided block of size 64×64 dimensions should contain at least a few words, which may ease implementation difficulties and speed up the computation process. This is valid because generally text appears as scatted clusters rather than filling the entire video frame. Another advantage of the proposed text block detection and script identification at block level is that when a block contains multiple scripts, the proposed method is capable of identifying scripts accurately. The divided block is big enough to capture a good number of characters and at the same time small enough to segregate multiple scripts. Even if two scripts may exist in the same block, there is a likelihood that one script may dominate in the block. Since our method chooses the minimum of Euclidean distance such that the dominant script will get identified. The less dominant script may appear more dominant in another block and gets identified later. This will help us in two ways. The first is to allow parallel processing, such that whichever pipe that detects text will go to the next stage in parallel with other pipes. Those pipes that do not find text will stop. Second, the parallel processing helps us to detect different scripts according to the dominant script in the respective blocks. The respective blocks will then be sent to the respective OCR engines according to the script identified automatically.

The proposed method is structured into five sub-sections. Section 3.1 introduces a gradient histogram method for obtaining dominant text components. Finding text candidates based on skeleton and filtering is proposed in section 3.2. Section 3.3 presents new gradient-spatial features which use the distances between end points, junction points, intersection points and pixels for the classification of scripts. Section 3.4 proposes new gradient-structural features based on the skeletonized text components. Section 3.5 presents a novel way of soft integration of spatial and structural features in the gradient domain for script identification. In Section 3.6, we describe how to create a template for identifying scripts for each method, namely, spatial, structural, combined and soft integration.

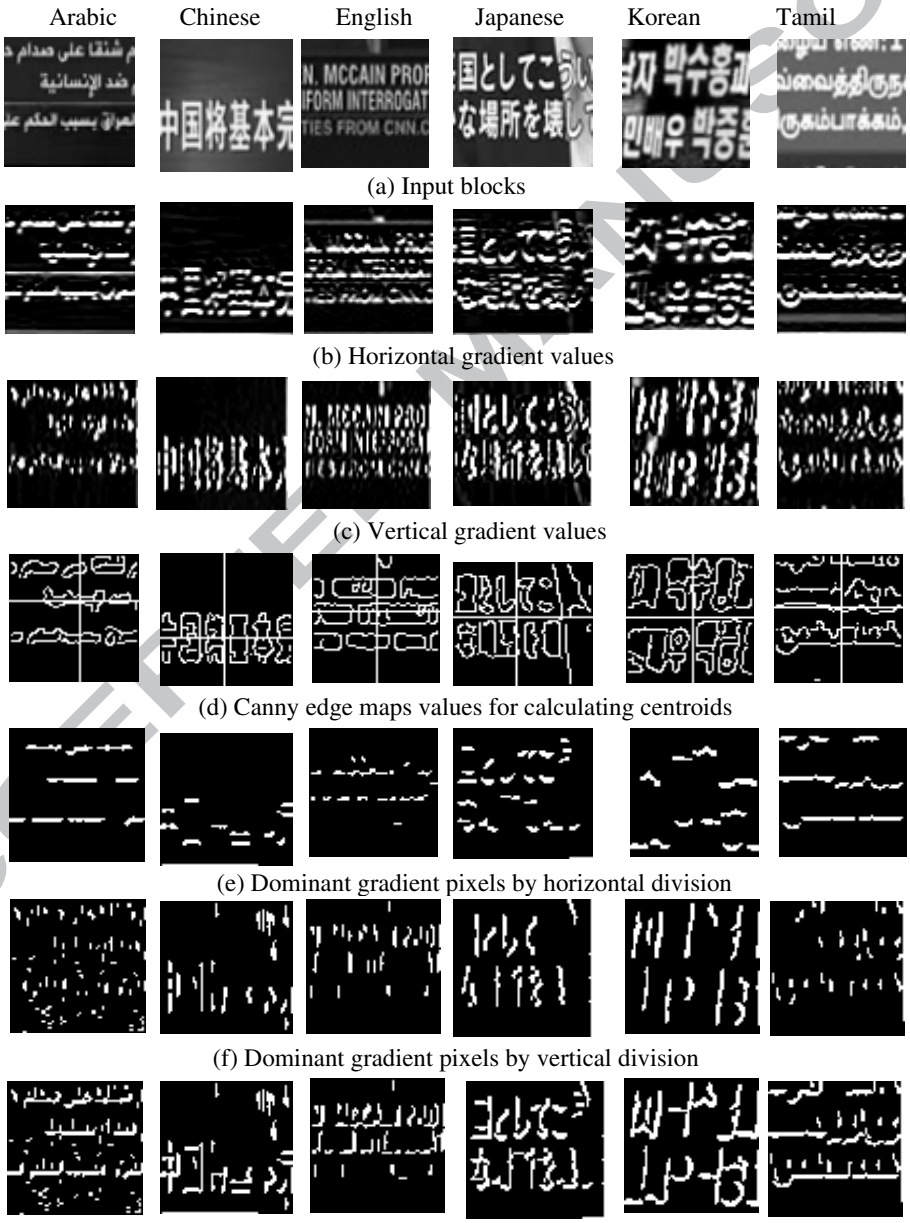
3.1. Gradient Histogram for Text Components

According to the literature, gradient provides a high value for a text pixel and a low value for a non-text pixel in video because it gives high response at edges or near edges [23]. It is true that edges are prominent features for classifying text and non-text. This clue leads us to perform convolution operation with horizontal and vertical masks on the input text block of any scripts. Equation (1) shows the formula for convolving horizontal and vertical masks over an input text block to obtain the gradient image. This is illustrated in Figure 1 where (a) shows the input text blocks of different scripts, namely, Arabic, Chinese, English, Japanese, Korean and Tamil, (b) and (c) show the outputs of respective horizontal convolution and vertical convolution operations. One can notice from (b) and (c) that text pixels at near edges and on edges are sharpened compared to the input blocks. Since video suffers from low resolution and complex background, there are chances of erroneously considering gradients which represent non-text as dominant gradient values when we study at block level. Therefore, we propose a new way of studying dominant gradient information by dividing each horizontal gradient block at the centroid into two equal parts, namely, the upper and the lower horizontally. In the same way, we divide each vertical gradient block at the centroid into two equal parts, namely, the left and the right vertically. This helps us to obtain the dominant gradient values which represent actual texts even for low contrast text pixels.

To find the centroid for a gradient block, we use the Canny edge image of the input block as shown in Figure 1(d). We perform histogram operation on each sub-part of gradient blocks and consider each gradient value which gives the highest peak as the dominant gradient value as shown in Figure 1(e) and (f), respectively for horizontal and vertical gradient blocks. Figure 1(e) and (f) show that prominent gradient values that represent actual edge pixels of text compared to horizontal and vertical gradient blocks shown in Figure 1(b) and (c). This is the advantage of gradient block division and histogram operation. Finally, we combine the dominant gradient values given by horizontal block and vertical block to get text components as shown in Figure 1(g), where the structure of the text components can be seen corresponding to different scripts. The steps and the logical flow can be seen in Figure 2 for identifying text components by gradient division and histogram operation. Equation (1) for gradient operation is as follows:

$$G_x = I'_x = \lim_{\Delta x} \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x} \approx \frac{I(x+1,y) - I(x-1,y)}{2},$$
$$G_y = I'_y = \lim_{\Delta y} \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y} \approx \frac{I(x,y+1) - I(x,y-1)}{2}$$
$$G_{x,y} = \sqrt{G_x^2 + G_y^2} \quad (1)$$

where I'_x and I'_y correspond to the derivatives along x and y directions at coordinate (x, y), respectively. $G_{x,y}$ is its corresponding gradient magnitude. Finally, G_x and G_y represent the gradients in horizontal and vertical directions, respectively, and I is the image.



(a) Input blocks

(b) Horizontal gradient values

(c) Vertical gradient values

(d) Canny edge maps values for calculating centroids

(e) Dominant gradient pixels by horizontal division

(f) Dominant gradient pixels by vertical division

(g) Text components (horizontal + vertical dominant gradients)

Fig. 1. Intermediate results of text components identification

3.2. Candidate Text Components

The previous Section 3.1 obtains text components from the input blocks. We believe that the structure of the text components differs from one script to another. Therefore, we set our objective to study the structure of the text components through identifying end points, junction points and intersection points. We propose to use the skeleton criterion to thin text components as shown in Figure 3(a), where we can see single pixel width text components compared to the results in Figure 1(g). This helps in studying the exact structure of the text components in addition to saving the computations in processing thick contours as well as to making the implementation simpler. Note that we use the modified skeleton algorithm as in [23] for thinning in this work, which shows that the modified skeleton preserves the structure of the connected components. The example for single pixel width can be seen in the magnified position on the skeleton of the text components of Chinese script in Figure 3(b), where the thickness of the stroke is actually a single pixel. To make our skeletons more robust to noise caused by the background, we calculate the area for each skeleton component and eliminate the skeleton components that have small areas. For the purpose of elimination, we use k-means with k=2 clustering algorithm on the areas of all the skeleton components. This results in two clusters. The method considers the cluster that gives a lower mean as the unwanted skeleton component clusters and hence we eliminate them using equation (2). The effect of the elimination can be seen in Figure 3(c), where we can notice that only stable skeleton components are retained, which we call candidate text components. This step helps us obtain stable candidate text components which are robust to noise.

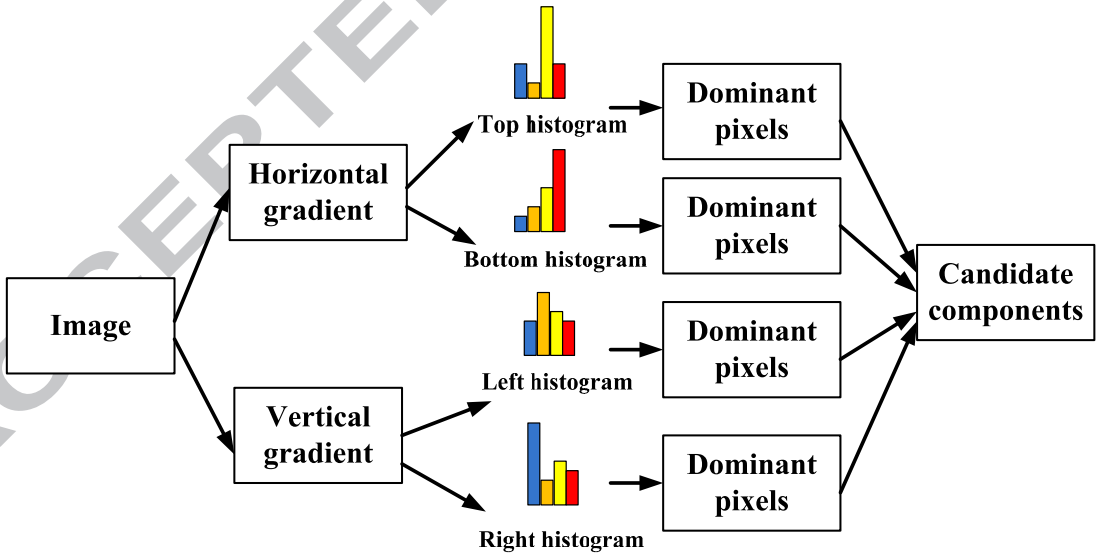
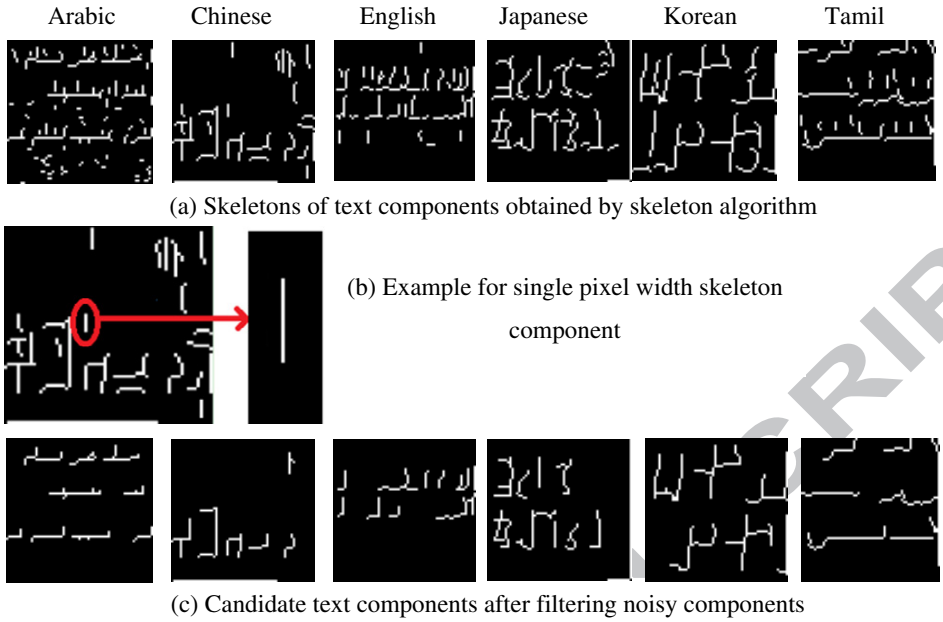


Fig. 2. Flow diagram for text components selection using gradient division



(a) Skeletons of text components obtained by skeleton algorithm

(b) Example for single pixel width skeleton component

(c) Candidate text components after filtering noisy components

Fig. 3. Candidate text components selection after removing false text candidates

$$\mu_{NC} = \min(\mu_{C1}, \mu_{C2}) \qquad (2)$$

where $\{\mu_{C1}, \mu_{C2}\}$ are the means of the two clusters and μ_{NC} is the mean of the noise clusters.

3.3. Gradient-Spatial-Features (GSpF)

When we look at the candidate text components of different scripts in Figure 3, one can notice that the spatial distributions of end points, junction points, intersection points, and the pixels of the components differ from each other. Equation (3) defines the end points, the junction points and the intersection points of the candidate text components in the block. This clue leads us to propose features for estimating the distances between the end points, the junction points, the intersection points, and the pixels using the Euclidean distance as defined in equation (4), which results in proximity matrices for each block of different scripts. The proximity matrix of endpoints is defined as the distance between the first end point to the rest of the end points, the second end point to the rest of the end points and so on until it visits all the end points. In this way, the method estimates proximity matrices for the junction points, the intersection points and the pixels. For example, it is observed from Figure 3(c) that the distributions of the above points of Arabic and Tamil appear sparse compared to the spatial distributions of the points in Chinese, Japanese and Korean. This is due to more sub-components with more cursiveness which results in more end points, junction points and intersection points in the cases of Chinese, Japanese and Korean compared to Arabic and Tamil, where we may not see components containing several sub-components. However, since the components in English share both cursiveness and straightness, the spatial distribution may be the same as the distributions in Chinese, Japanese and Korean. To validate these observations, we calculate the variances for the proximity matrices of respective points for the input scripts shown in Figure 1(a). The results are reported in Table 1. One

can notice from Table 1 that the variance of almost all the points of Arabic and Tamil is higher than other scripts. This shows that the spatial distribution of those feature points is sparse compared to other scripts. On the other hand, the variances of Chinese, English and Korean have low values. This indicates that these points are distributed as close as possible. In summary, we can conclude that these observations contribute more for classifying English, Chinese, Japanese and Korean from Arabic and Tamil scripts. We extract these observations by computing the variances for the proximity matrices of the end points, the junction points, the intersection points and the pixels, respectively. This results in a single vector containing four variance features.

The definitions for the end points, the junction and the intersection points are as follows.

For any pixel *P* with surrounding pixels $P_{surround}$ in a component $P_{component}$:

$$P_{surround} \wedge P_{component} = k \qquad (3)$$

P is an end point when k = 1 in equation (3), a junction point when k = 3, and an intersection point when k = 4.

The proximity matrices for the end points, the junction points, the intersection points and all the pixels are defined as follows. The proximity matrix of end points is represented by $ED_{i,j}$, where P_{End} is the set of all end points.

$$ED_{i,j} = ||P_{End_i} - P_{End_j}||_2 \qquad (4)$$

The proximity matrices of the junction points, the intersection points and all the pixels respectively represented by $JD_{i,j}$, $ID_{i,j}$ and $PD_{i,j}$ are computed as in equation (4). In the same way, the variances for proximity matrices are computed as follows. The variance of the proximity of end point distances is represented by $Var(ED)$, where ED is the set of all the end point distances and μ_{ED} is the mean of the end point distances. n is the size of the corresponding proximity matrix.

$$Var(ED) = \frac{1}{n^2}\sum_{i}^{n}\sum_{j}^{n}(ED_{i,j} - \mu_{ED})^2 \qquad (5)$$

For example, we randomly sample five pixels in one candidate text component image of Korean script. For those five actual pixels, we estimate the proximity matrix using equation (4), resulting in the following matrix. The method finds the variance for this proximity matrix and then it follows normalization to make the feature more stable as shown in Table 1.

$$\begin{bmatrix} 0.00 & 9.22 & 8.25 & 16.49 & 10.20 \\ 9.22 & 0.00 & 9.22 & 10.44 & 8.54 \\ 8.25 & 9.22 & 0.00 & 10.00 & 15.62 \\ 16.49 & 10.44 & 10.00 & 0.00 & 18.97 \\ 10.20 & 8.54 & 15.62 & 18.97 & 0.00 \end{bmatrix}$$

The variance of the junction point distances, the intersection point distances and all the pixel distances respectively represented by $Var(JD)$, $Var(ID)$ and $Var(PD)$ are computed as in equation (5).

Table 1. Variance features for proximity matrices of end points, junction points, intersection points and pixels for the input scripts shown in Figure 1(a)

Scripts/Features	Variance (ED)	Variance(JD)	Variance(ID)	Variance(PD)
Arabic	0.13	0.02	0.01	0.15
Chinese	0.04	0.01	0.00	0.05
English	0.04	0.00	0.00	0.03
Japanese	0.04	0.04	0.02	0.09
Korean	0.05	0.01	0.01	0.07
Tamil	0.15	0.09	0.07	0.18

3.4. Gradient-Structural-Features (GStF)

It is observed that spatial features alone may not be sufficient to achieve a good accuracy due to low contrast and complex background of video. To achieve a good accuracy for identifying the six video scripts, we propose a few more new features using gradient-structural information of the Candidate Text Components (CTC) because as it is noted from Figure 3 that English, Arabic and Tamil may share common spatial information but not structural information. Therefore, gradient-structural features at component level and branch level are proposed.

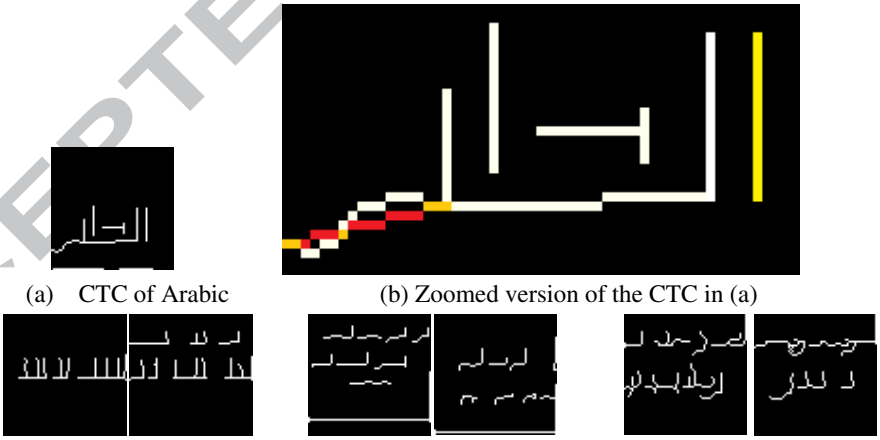
3.4.1. Features at component level

We believe that the numbers of end points, intersection points and junction points exhibit differences among the six scripts especially for the group consists of Arabic, English, Tamil and the group consists of Chinese, Japanese, Korean because the former group does not have components with sub-components and shows low cursiveness compared to the latter group. With this observation, we propose to extract the features by counting the numbers of end, intersection and junction points. The method finds these points as defined in the previous section and counts their numbers as follows. The total number of end points in a block is represented by $Sum(P_{End})$ where P_{End} is the set of all the end points in the block. We divide the total sum by N where N is the number of components in the block to make the feature invariant to the number of components in the block. This is applied to all the features introduced in this section at the component level except the features F7-F10.

$$Sum(EP) = \frac{1}{N} Sum(P_{End}) \qquad (6)$$

In the same way, the total numbers of the junction points and the intersection points respectively represented by $Sum(JP)$ and $Sum(IP)$ are computed as in equation (6). With this procedure, we get three features (F1-F3) based on the end, the junction and the intersection points.

Next, we can also observe from the candidate text components that the structure of the components of different scripts differs from one script to another. For instance, more straightness like straight lines can be seen in the cases of English and Arabic compared to other scripts. Based on this observation, we introduce new features to study the structure of the components based on straightness and cursiveness of the branches in the components. The method finds a pixel that satisfies the condition *k* >=2 in equation (3) (intersection or junction), which joins two branches of the components in the block. This segment consists of two branch endpoints that come from EP, IP and JP. Let a pixel be the branch point which satisfies condition k >= 2 in equation (3). Traversing any branch point via its two adjacent pixels will give us the two branch end points. Let $P_{Straight}$ be the calculated straight line pixels formed from the two branch end points and P_{Branch} be the pixels in the branch. A branch is defined as straight if $P_{Branch} \in P_{Straight}$ else it is defined as cursive or oscillating. The total numbers of the branches that satisfy either straightness or cursiveness property, the straight branches that satisfy the above straightness condition, and the curve (oscillating) branches that unsatisfy the straightness property are respectively represented by $Sum(Branches)$, $Sum(Branchs_{Straight})$ and $Sum(Branches_{Cursive})$, which are computed as in equation (6). With this procedure, we get three more features (F4-F6). These features are shown in Figure 4 for better understanding.



(a) CTC of Arabic (b) Zoomed version of the CTC in (a)

(c) CTC of English (d) CTC of Arabic (e) CTC of Tamil

Fig. 4. Illustration of straightness and cursiveness of the edges in the components

Figure 4(a) shows a candidate text component of Arabic, while (b) is a zoomed-in version of (a) where we can see both straight and curve branches. For example, the line on the right side of Figure 4(b) marked by yellow color is a straight branch, and the cursive branch on the left bottom corner side in Figure 4(b) has junction points and end points that are marked by orange color. The straight line is formed by joining the end points and the junction points as shown in red color, which intersects with

the cursive branch at orange color pixels. As a result, we get two oscillations (like holes) with one cursive branch as the branch does not satisfy the straightness property. In other words, if the branch does not have any oscillation then it is considered as a straight branch. This feature is useful for classifying the scripts such as Arabic, English and Tamil. For instance, it can be seen from Figure 4(c) of CTC-English, Figure 4(d) of CTC-Arabic and Figure 4(e) of CTC-Tamil that more straight and less curve branches in the case of CTC-English, less straight and more cursive branches in the case of CTC-Arabic, while in the case of CTC-Tamil, cursiveness increases and straightness decreases compared to CTC-English and Arabic. In this way, this feature helps in achieving a good classification rate. Besides, this feature is new to define straightness and cursiveness in this work as per our knowledge.

We define structural features based on the proximity matrices defined as in Section 3.3. The proximity matrices are obtained for the end points, the junction points, the intersection points and all the pixels as defined in equation (4). Then the means of the proximity matrices are computed as follows and considered as features for script identification. The mean of the proximity of end point distances is represented by μ_{ED}, where ED is the distances matrix of all end points:

$$\mu_{ED} = \frac{1}{n^2}\sum_{i}^{n}\sum_{j}^{n} ED_{i,j} \qquad (7)$$

The mean of the junction point distances, the intersection point distances and all the pixel distances respectively represented by μ_{JD}, μ_{ID} and μ_{PD} are computed as in equation (7). As a result, we get four more features (F7-F10) based on the proximity distances.

The features introduced earlier (F5 and F6) are rigid to check the straightness property of the branch in the components. Therefore, we propose a weak feature to define straight components because F5 fails if a branch has a small curve. This condition allows the branches that have small curves along with exact straight branches compared to the previous feature F5. Let P be all the pixels in the component and $P_{Centroid}$ be the centroid of P. If $P_{Centroid}$ falls on the same component (if $P_{Centroid} \in P$) then we consider that the component satisfies the straightness property. We count the number of the components that satisfy the above straightness property in the block and the feature (F11) is computed as in equation (6).

When we look at the candidate text components in Chinese, Japanese, Korean and other scripts, it is observed that the numbers of intersection and junction points are more for Chinese, Japanese and Korean compared to the rests, due to the cursive nature of the scripts in the sense that the characters have more strokes in different directions whereas Arabic, English and Tamil scripts have one continuous stroke. Thus there are more chances of intersections. Therefore, we extract the components that have more than two end points and junction points to classify the scripts. This feature is a stricter one compared to F1-F3, where the feature considers all the end points and the junction points. This

feature helps in classifying Chinese, Japanese and Korean from other scripts. Let Comp be the set of all the components in a block. The number of the components that have more than two end points is represented by $Sum(Comp_{EP>2})$, where $Comp_{EP>2}$ is the set of all the components with more than two endpoints:

$$Sum(Comp_{EP>2}) = Sum(Comp_{Sum(CompEP)>2}) \quad (8)$$

The number of the components that have more than two junction points represented by $Sum(Comp_{JP>2})$ is computed as in equation (8), while the features (F12 and F13) are computed as in equation (6).

3.4.2. Features at branch level

We also extract features at branch level to study the local structure of the components in the block. Similar to the procedures at the component level, we divide features in this section by E, which is the number of branches, to make them invariant to the number of branches.

Let P be all the pixels in a branch and $P_{Centroid}$ be the centroid of P. We count the number of the branches that satisfy the straightness property ($P_{Centroid}$ falls on the branch if $P_{Centroid} \in P$) as it is done in Feature 11. This feature is extracted based on branch information, while F11 considers the whole component to check the straightness property. This feature helps in studying the straightness of each branch in the component and it is considered as F14 after dividing the sum by the number of the branches.

To analyze the cursiveness of each branch of the component, we extract features based on intersection points hit on curve branch as defined in F6. Therefore, here the number of intersections is defined as the number of times points on the curve branch intersect with the straight line formed with the two endpoints of the branch. Let $P_{Straight}$ be the calculated straight line pixels formed from the two branch end points and P_{Branch} be the pixels in the branch. The number of intersections in the branch is represented by $Sum(\text{Intersections})$ where

$$\text{Intersections} = P_{Straight} \cap P_{Branch} \quad (9)$$

This feature is considered as feature (F15).

To analyze the amount of cursiveness in each branch of the components, we extract features based on areas of oscillations defined for each branch as discussed in F5 and F6. Area of oscillation, namely, $Area(Oscillations)$, sums up the deviation of the curve branch points from the straight line formed with the two end points along the x-coordinates of $P_{Straight}$.

$$Area(Oscillations) = \sum |P_{Straight} - P_{Branch}| \quad (10)$$

This area based feature is considered as F16.

To study the exact straightness of each branch in the components, we propose a new condition based on midpoint and centroid of the branch. Let P be all the pixels in the branch and n be the number of pixels in the branch, the middle point P_{Middle} of the branch is P_i where $i = n/2$. The centroid and the middle points fall on the same point if $P_{Centroid} \equiv P_{Middle}$. The total number of the branches that satisfy this property is considered as a feature (F17). In this way, we extract in total 17 structural features at both component and branch levels to improve the accuracy of video script identification. The above features at both levels are derived based on the observation but not with a fixed threshold. Therefore, we can believe that the features work like objective features rather than heuristics. In order to find the contributions of the features both at component level and branch level, we conduct experiments on our test data to calculate classification rates independently. This will be discussed in the Experimental section.

3.5. Soft Integration of Gradient-Spatial-Structural Features (GSSF)

It is noted from the above feature extraction methods that the Gradient-Spatial-Features (GSpF) are good for Chinese, Japanese and Korean since the spatial distribution of the end point distances, the intersection point distances, the junction point distances and the pixel distances differ from one script to the other. It can be seen in Figure 3 where due to different cursive natures of the scripts, we can see a different spatial distribution for each script. Similarly, the Gradient-Structural-Features (GStF) are good for Arabic and Tamil scripts because these scripts have distinct structures of the components and branches. It can be confirmed from the results shown in Figure 3, where we can notice different structures at component level and branch level in each script. As a result, it can be concluded that both the feature extraction methods complement each other. Therefore, we combine and integrate the above two feature extraction methods. Here combination means considering 4 from GSpF and 17 from GStF together as a single feature vector (21 features) for script identification, while integration means calculating the error factor based on the weight of the confidence score given by a boosting classifier with the help of respective templates of the two methods. In our method, inspired by the boosting framework [61], a soft integration is proposed. Rather than passing the errors of the first method to the second one, for every block, we calculate its respective Euclidean distances to the templates of the two methods and then integrate these distances together to identify the block. More importantly, we do not generate templates of the two methods independently when training and the templates of the second method need to cater to more errors of the first method. Thus, compared to combining in our method, the terminology “soft integration” is used to manifest that the template formation of the two methods are dependent to make use of their complementation with each other. However, we consider a 21-feature vector as a single feature extraction method for combining and generate one set of templates using few sample data. Template formation for combining and soft integration will be discussed in Section 3.6. The flow diagram for the proposed soft integration can be seen in Figure 5. The top rectangle in Figure 5 shows the template formation of GSpF and GStF for the six scripts for integration. We first get templates of GSpF and then use GSpF method to obtain classification results

of the training data. Based on these results, we reweigh the training data in a way that increases the weight of misclassified results and lowers those for correct ones. Then the templates of GStF are generated by the reweight data. Intuitively, the templates of GSpF and GStF can be complementary to make identification. The classification pipeline is given in the bottom rectangle in Figure 5. More details for predicting error based on weights will be discussed in Section 3.6 for the soft integration method.

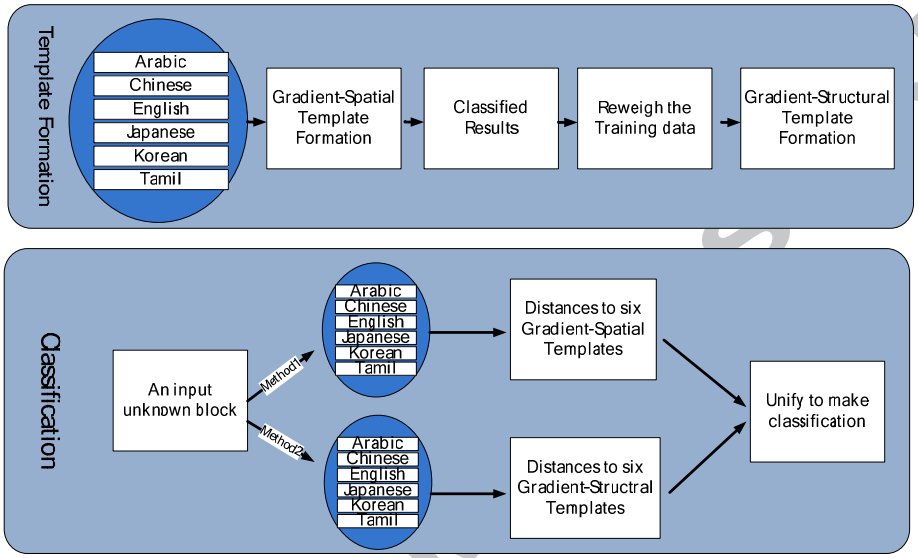


Fig. 5. Template formation and classification of soft integrated method

3.6. Template Formation and Classification

As our intention is to use the number of samples as few as possible, we choose 50 blocks from each class of scripts randomly for constructing templates. We have tested this idea of creating templates and identification of scripts in [55] with only GSpF features on a small dataset. It is noted that this template creation works well for achieving a good accuracy. In addition, it requires less time for matching and classification. Hence, we are motivated to use the same idea for script identification by a new soft integration method in this work for a large database. The four spatial, seventeen structural and combined features (21 feature vectors) are computed for the blocks corresponding to 50 blocks of each script class. Then the average of the variance (4), structural (17) and combined features (21) of the blocks for each script is computed as defined in equation (11) below:

$$Avg(Feature) = \frac{1}{50}\sum_{i=1}^{50} Feature_i \qquad (11)$$

This gives six templates (vectors) for the six scripts containing four average variances, seventeen structural and twenty one combined features, respectively for the GSpF, the GStF and the combined. For the given block, the method extracts its GSpF, GStF and combined features and compares them with the respective six templates to find the minimum Euclidean distance to classify the frame into a

particular class. The Euclidean distance for classification here is defined as follows. For a text block with extracted feature vector FV, the set of Euclidean distance ED = {ED_1, ED_2 … ED_6} between FV and the set of template vector T = {T_1, T_2 … T_6} is given by

$$ED_i = \sqrt[2]{\sum_{r=1}^{d}(FV_r - T_{i_r})^2} \qquad (12)$$

where d is the dimension, d = 4 if we are considering the 4 gradient-spatial features, d = 17 if we are considering the gradient-structural features, and d = 21 if we are considering their combined features.

The classified script is given by $S \in \{1, 2 \ldots 6\}$ where

$$S = \arg\min_S(ED_S) \qquad (13)$$

This is the same as K nearest neighbor classification with k=1. This procedure gives confusion matrices for the six scripts of the three methods. The sample templates for GSpF, GStF and combined features (21 feature vectors) for the scripts, namely, Arabic, Chinese, English, Japanese, Korean and Tamil can be seen respectively in Figure 6(a)-(f), Figure 7(a)-(f) and Figure 8(a)-(f). The templates shown in Figure 6, Figure 7 and Figure 8 are plotted by considering the number of feature as X axis and the average feature values as Y axis. It is observed from Figure 6(a)-(f) that Tamil template may appear almost the same as English template. As a result, we may get poor results for Tamil compared to other scripts using GSpF. Figure 7(a)-(f) show that the templates of Chinese, Japanese and Korea may share some common features and hence we may get poor results for these scripts using GStF. Further, Figure 8(a)-(f) show that the templates of Tamil and Japanese may appear as English and Arabic templates and hence we may get poor results for these scripts compared to other scripts. From the above discussion, it is confirmed that GSpF alone, GStF alone and their combination are not good enough for classifying six video scripts with a good accuracy. However, the two complement each other since GSpF gives good results for English, Chinese, Japanese and Korean but it gives bad results for Arabic and Tamil scripts, while GStF gives good results for Arabic and Tamil but not the rest. Hence, the integration of GSpF and GStF is proposed in this work.

Without loss of generality, we consider the GSpF features for the first method and GStF for the second method. Thus, we first generate the templates for GSpF features and then get those for GStF based on the results obtained only by GSpF templates and the minimal Euclidean distance. We also use GSpF features first because the GSpF method gets better results than the GStF method individually according to our experiment. Therefore, we use the templates of GSpF features to modify the template formation for GStF to cater to errors caused by the GSpF predictor. Specifically, we select 50 blocks for each script. Then the four spatial and the seventeen structural features are computed for the 50 blocks of each script. The template T_i^p for GSpF and $ED_i^p i = \{1, \ldots, 6\}$ are calculated according to Equation (11) and (12), respectively. Here we normalize ED_i^p over the six scripts, namely, $ED_i^p =$

$\frac{ED_i^p}{\sum_i ED_i^p}$. For any sample (x, i) (i is the script label of x) in training data (6 script*50 blocks), the predicted class from the template of GSpF is $h^p(x) = \text{argmin}(ED_i^p)$. For each script, we calculate the error rate $\varepsilon^p(h^p(x) \neq i)$. Then for the sample (x, i), the weight w_x is defined as

$$w_x = \begin{cases} \exp(-a_p(1 - ED_i^p)), h^p(x) = i \\ \exp(a_p ED_i^p)), h^p(x) \neq i \end{cases} \quad (14)$$

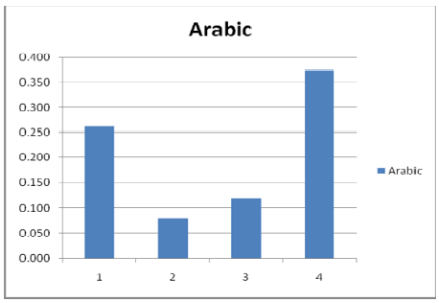
$$a_p = \log[(1 - \varepsilon^p)/\varepsilon^p] \quad (15)$$

Thus w_x is large for an incorrect sample x caused by the minimal Euclidean distance classifier (Equation 11) h^p (corresponding to $h^p(x) \neq i$) and is low for the sample with $h^p(x) = i$. Then the template for GStF features are calculated by $T_i^t = \frac{w_x}{Z}\sum_x \text{GStF}_x, i = \{1, \ldots, 6\}$, where Z is used to normalize w_x. And for the class i, the error rate $\varepsilon_i^t = \sum_x \frac{w_x}{Z}$(for x with $argmin(ED_i^t) \neq i$), where ED_i^t is also calculated by (12) and normalized over six classes. Let

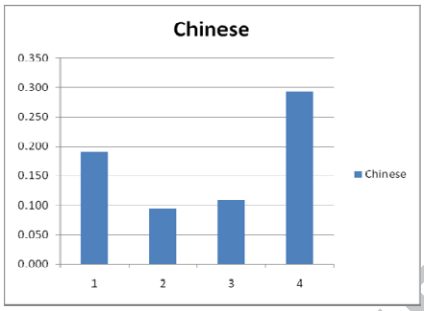
$$a_t = \log[(1 - \varepsilon^t)/\varepsilon^t] \quad (16)$$

the overall $ED_i = a_p ED_i^p + a_t ED_i^t$.

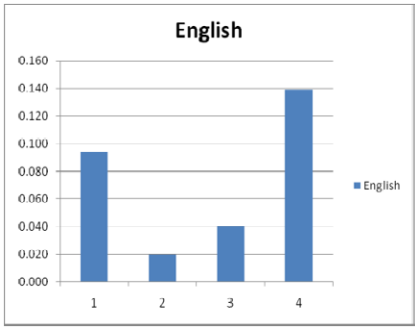
For a text block c, we just calculate the Euclidean distance according to (12) for templates T_i^t and T_i^p where $i = \{1, \ldots, 6\}$ and then obtain ED_i. The class label is given by $\text{argmin}(ED_i)$.



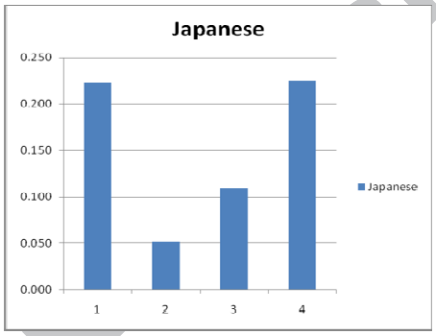
(a) Arabic



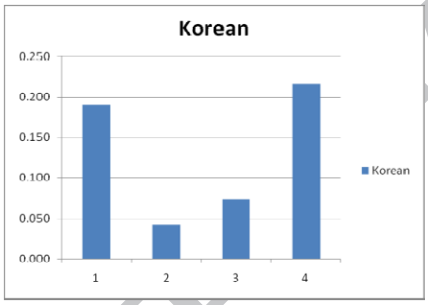
(b) Chinese



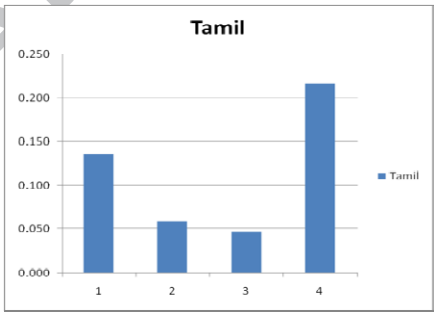
(c) English



(d) Japanese

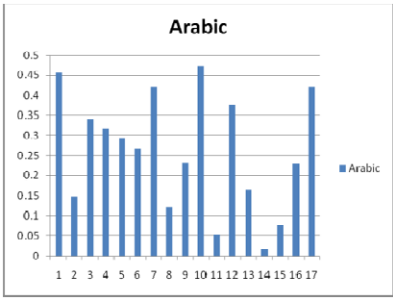


(e) Korean

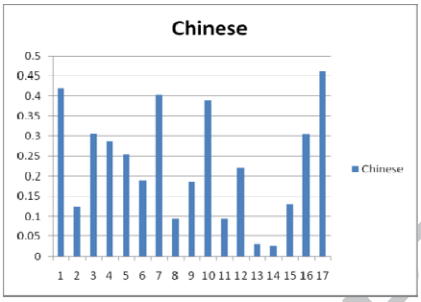


(e) Tamil

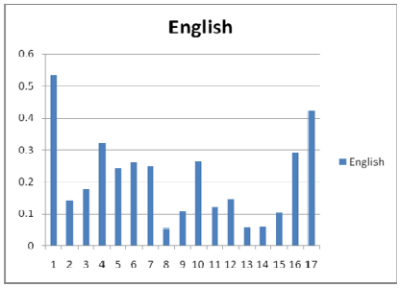
Fig. 6. Templates using Gradient-Spatial Features: X axis represents the number of features and Y axis represents average spatial (variance) feature values (4)



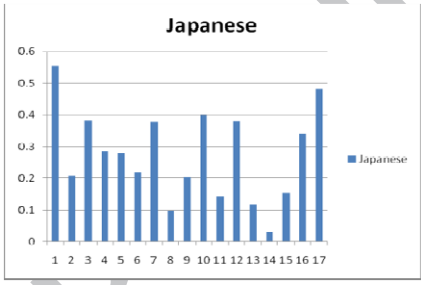
(a) Arabic



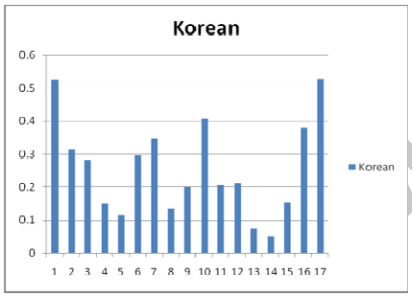
(b) Chinese



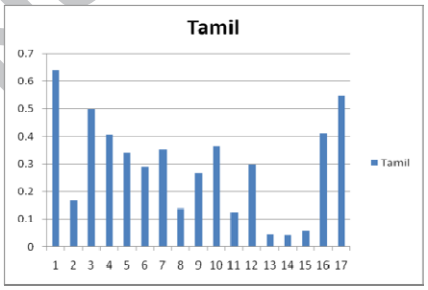
(c) English



(d) Japanese

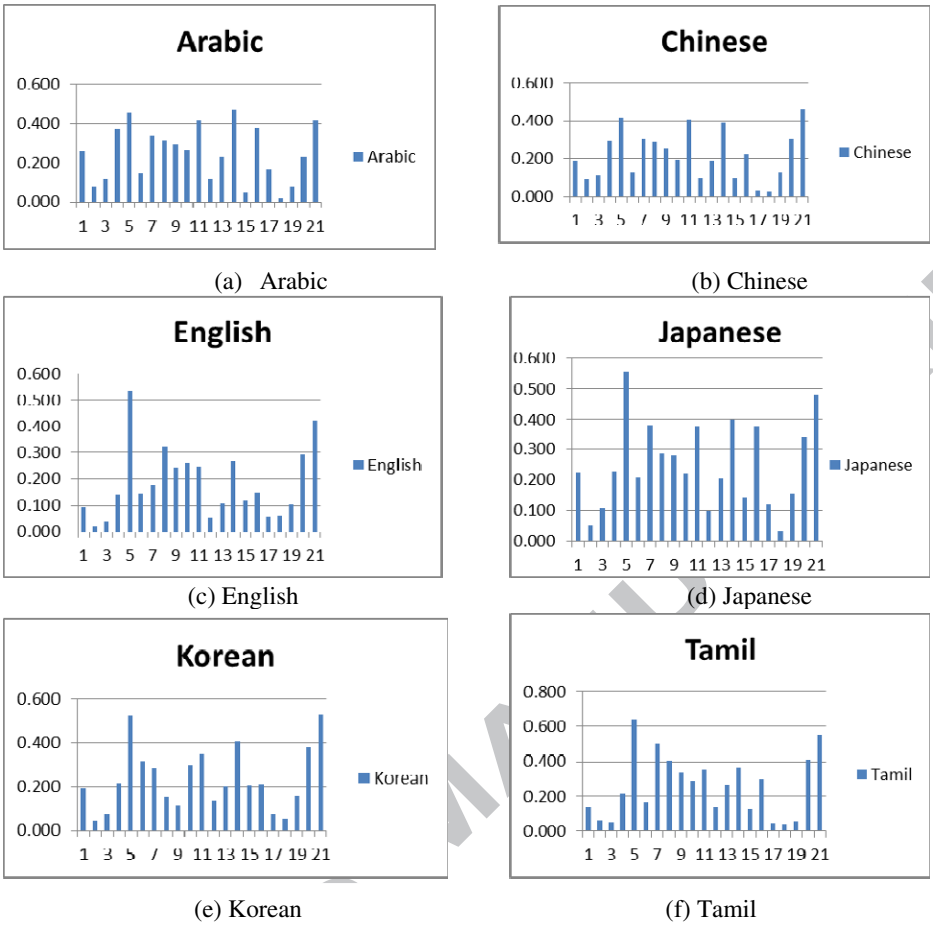


(e) Korean



(f) Tamil

Fig. 7. Templates using Gradient-Structural-Features. X axis represents the number of features and Y axis represents average structural feature values (17)



(a) Arabic (b) Chinese

(c) English (d) Japanese

(e) Korean (f) Tamil

Fig. 8. Templates for combining Gradient-Spatial-Structural-Features. X axis represents the number of features and Y axis represents average combined feature values (21)

4. Experimental Results

We create our own dataset chosen from different sources, such as sports news, weather news, and entertainment video to show that the proposed method works well for different varieties of video frames. As to template generation, we choose 50 blocks manually for each script, thus a total of 300 blocks to construct their templates. Our testing dataset includes 200 Arabic frames, 200 Chinese frames, 200 English frames, 200 Japanese frames, 200 Korean frames and 200 Tamil frames. In total, 1200 frames are used for the purpose of experimentation and evaluation. The sample blocks of the six scripts from our database are shown in Figure 9, where the first six columns show sample blocks of Arabic, Chinese, English, Japanese, Korean and Tamil scripts, respectively. One can notice from Figure 9 that the blocks of different scripts contain different fonts, font sizes, contrast and background. To evaluate the performance of the method, we consider classification rate as the measure, and present a confusion matrix containing classification rate/misclassification rate of each script. In subsequent sections, we present the analysis of the individual features of GSpF and GStF (component level and branch level features), the experiments on combined features, the experiments on integrated method to

show that the proposed method is robust to noise some extent and the experiments on scanned images using integrated method to evaluate the effectiveness and the extensibility of the proposed method.

Fig. 9. Sample text blocks of six scripts chosen from our video database

4.1. Experiments on Individual Set of Features

The objective of this experiment is to find the contribution of each set of features, namely, GSpF and GStF in terms of average classification rate (average of diagonal elements of confusion matrices of each feature). Figure 10 shows the average classification rate for the four Gradient-Spatial-Features (GSpF). It is noticed from Figure 10 that feature (F1) and feature (F4) contribute more than feature (F2) and feature (F3) for classification because the proximity between end points (F1) and all pixels (F4) gives distinct features for the six scripts. In the same way, the proximity between junction points (F2) and intersection points (F3) contributes less compared to F1 and F4. This shows that each feature contributes for classification. Therefore, we combine these four features to get better results for classification of the six scripts.

We also analyze the 17 Gradient-Structural-Features (GStF) on classifying six video scripts in terms of average classification rate as it shown in Figure 11, where one can notice that the features F7, F9, F10, F12 and F13 are better than the other features because these features are based on the distances

between end points, the distances between junction points and the mean of the number of components that have more than two end points and two junction points. Therefore, these features contribute more than the other features for classification. However, other features also contribute significantly for classification. Hence, we combine all the 17 features together to achieve better results for classification.

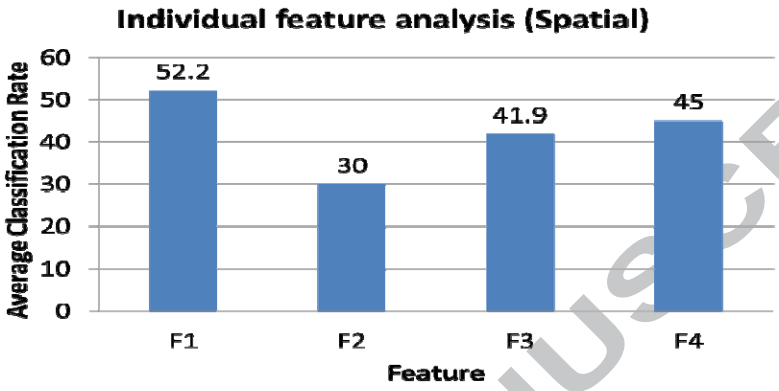


Fig. 10. Average classification rate of individual features of GSpF (spatial): F1, F2, F3 and F4 represent features of end points, junction points, intersection points and pixels, respectively.

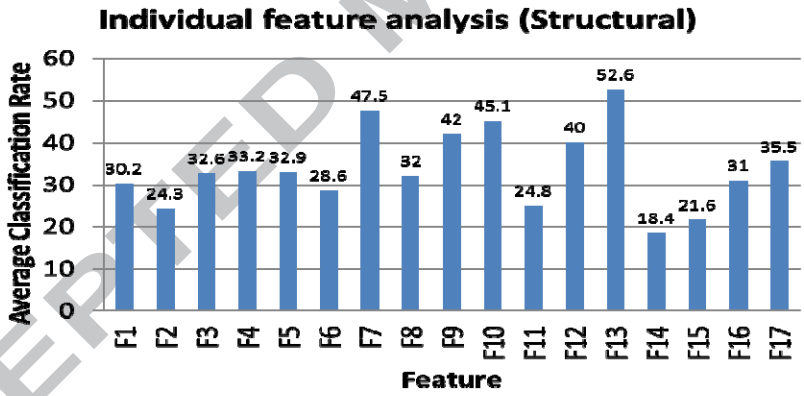


Fig. 11. Average classification rate for individual features of GStF (structural)

4.2. Experiments on GSpF

To study the contribution of Gradient-Spatial-Features (GSpF) in terms of classification rate, we conduct experiments on our dataset using four features. Based on the experiment, we generate a confusion matrix for the six video scripts as shown in Table 2. Table 2 shows that the GSpF features are good for English, Chinese, Korean and Japanese but not good for Arabic and Tamil as most of the data of Arabic and Tamil are misclassified as English. Therefore, we can infer that the GSpF features alone may not be sufficient to achieve a good classification rate for all the six scripts.

Table 2. Confusion matrix of GSpF (in %)

Scripts	Arabic	Chinese	English	Japanese	Korean	Tamil
Arabic	**47.1**	13.5	18.5	7.8	7.1	5.7
Chinese	6.1	**72.3**	3.0	3.0	8.4	6.9
English	1.4	0.7	**92.9**	0.7	2.2	1.8
Japanese	1.4	2.8	15.7	**59.2**	15.7	5.0
Korean	0.7	2.8	16.4	2.8	**70.7**	6.4
Tamil	7.3	7.3	28.6	2.0	12.0	**42.6**

4.3. Experiments on GStF

As it is realized from the previous experiments that we need a few more features that are capable of discriminating the six video scripts with a good classification rate, we conduct experiments on all the six scripts using Gradient-Structural-Features (GStF) to study the 17-feature vector contribution in terms of classification rate. According to the results reported in Table 3, GStF give good results for Arabic and Tamil compared to Chinese, English, Japanese and Korean. These scripts are misclassified as Arabic and Tamil because the extracted structural features may overlap with Arabic and Tamil scripts. Therefore, we can conclude that GStF alone may not be sufficient to achieve a better classification rate for all the six video scripts. Further, we can also notice from the experimental results of GSpF and GStF that GSpF are good for English, Chinese, Japanese and Korean but not good for Arabic and Tamil, while GStF are good for Arabic and Tamil but not good for the rest. This observation makes us combine these two feature extraction methods to get better results for classification.

Table 3. Confusion matrix of GStF (in %)

Scripts	Arabic	Chinese	English	Japanese	Korean	Tamil
Arabic	**75.0**	5.7	3.5	1.4	1.4	12.8
Chinese	13.8	**54.6**	0.0	3.0	6.9	21.5
English	30.0	1.1	**62.2**	0.7	2.5	3.3
Japanese	16.4	14.2	5.7	**50.0**	5.0	8.5
Korean	14.2	1.4	0.7	7.1	**52.8**	23.5
Tamil	12.6	2.0	4.0	0.6	2.0	**78.6**

It is noted that GStF features are the integration of the features at component level (F1-F13) and branch level (F14-F17) as discussed in Section 3.4. Therefore, we also conduct experiments on features at component level and branch level separately to understand the contribution from each type. Classification rate in the form of confusion matrix for both the types are reported in Table 4 and Table 5, respectively. Table 4 shows that the features at component level give a good classification rate for Chinese, English, Japanese and Korean, and give a low accuracy for Tamil and Arabic compared to the classification rate of the features at branch level as reported in Table 5. When we look at Figure 11, we

can easily understand that F7 and F13 are good structural features as they contribute more than other features. As discussed in Section 3.4, F7 extracts the mean of the proximity matrix of end points and F13 extracts the components that have more than two end points. These two features are good for classifying Chinese, Japanese and Korean but not good for Tamil and Arabic since F1-F13 involves F7 and F13. On the other hand, the features at branch level are good for classifying Arabic and Tamil because the branches are more cursive in case of Tamil and less cursive in case of Arabic. However, the features at branch level are not good for Chinese, Japanese, Korean and English as they may share the same amount of cursiveness and straightness. It is observed from Table 5 that most of the scripts overlap with the Arabic scripts. In summary, we can conclude that the features at component level are good for Chinese, Japanese, Korean and English, while the features at branch level are good for Arabic and Tamil. Thus, the combination of the features at both component and branch levels is needed to achieve a good accuracy for all the scripts.

Table 4. Confusion matrix of GStF features (F1-F13) at component level (in %)

scripts	Arabic	Chinese	English	Japanese	Korean	Tamil
Arabic	**87.5**	0	12.5	0	0	0
Chinese	3.4	**82.5**	13.9	0	0	0
English	4. 57	6.8	**85.7**	0	0	2.8
Japanese	18. 2	3.4	17.2	**57.1**	0	3.9
Korean	8.4	0	12.4	0.0	**73.6**	4.9
Tamil	18.3	1.9	13.8	0	0.0	**65.3**

Table 5. Confusion matrix of GStF features (F14-F17) at branch level (in %)

scripts	Arabic	Chinese	English	Japanese	Korean	Tamil
Arabic	**100.0**	0	0	0	0	0
Chinese	48.0	**47.0**	0	0	0	4.9
English	48.1	0	**48.1**	0.2	0	3.3
Japanese	58.7	0	0	**33.3**	0	7.9
Korean	53.5	0	0	0	**43.5**	3.0
Tamil	2.9	0	0	0	0.5	**96.5**

4.4. Experiments on Combined GSpF and GStF Features

The experimental results discussed in section 4.3 and 4.4 reveal that there is a necessity of combining the two feature extraction methods to get a better classification rate for all the six video scripts. With this intention, we combine the 4 features from GSpF and the 17 features from GStF as to form a 21-feature vector for classification. We conduct experiments on our dataset and the results are reported in Table 6, where we can see that the classification rates for Arabic and Tamil have dropped drastically compared to the results of GStF, and the classification rates for English, Chinese, Japanese and Korean have dropped compared to the results of GSpF. When we compare the averages of the diagonal elements in Table 2, Table 3 and Table 6, namely, 64.1% for GSpF, 62.2% for GStF and 58.6% for combined GSpF and GStF, we find the combined method gives lower results than GSpF and GStF. The main reason for this is that creating one feature vector from two different scaled feature values bring a

problem for matching where we expect the minimum Euclidean distance for classification. Due to the different natures of feature vectors, the combined method misclassifies most of the data.

Table 6. Confusion matrix of combined GSpF and GStF (in %)

Scripts	Arabic	Chinese	English	Japanese	Korean	Tamil
Arabic	**83.571**	7.143	5.0	0.714	2.143	1.429
Chinese	15.385	**64.615**	0.0	6.923	10.769	2.308
English	29.63	0.741	**64.815**	1.111	2.222	1.481
Japanese	15.0	3.571	13.571	**46.429**	15.714	15.0
Korean	15.0	2.143	15.0	2.143	**61.429**	4.286
Tamil	23.333	6.667	24.667	2.667	11.333	**31.333**

4.5. Experiments on Soft Integration of Gradient-Spatial-Structural-Features (GSSF)

The discussion in the previous Section 4.4 shows that the combined method is not good for classification of all the six scripts as it gives poor results compared to individual methods. The other way is to integrate the two methods by combining the strength of each method. Therefore, we propose soft integration of the two methods in this work to achieve a better classification rate for all the six video scripts. The results of the soft integrated method are reported in Table 7, where we can notice the classification rate for the all the scripts increases compared to GSpF, GStF and the combined method. The average of the diagonal elements in the Table 7 gives 83.0%, which is higher than 64.1% of GSpF, 62.2% of GStF and 58.6% of the combined method. Hence, it is concluded that rather than combining two different methods simply, it is better to integrate them based on error factors and weights to take the full advantage of each method to achieve a better classification rate.

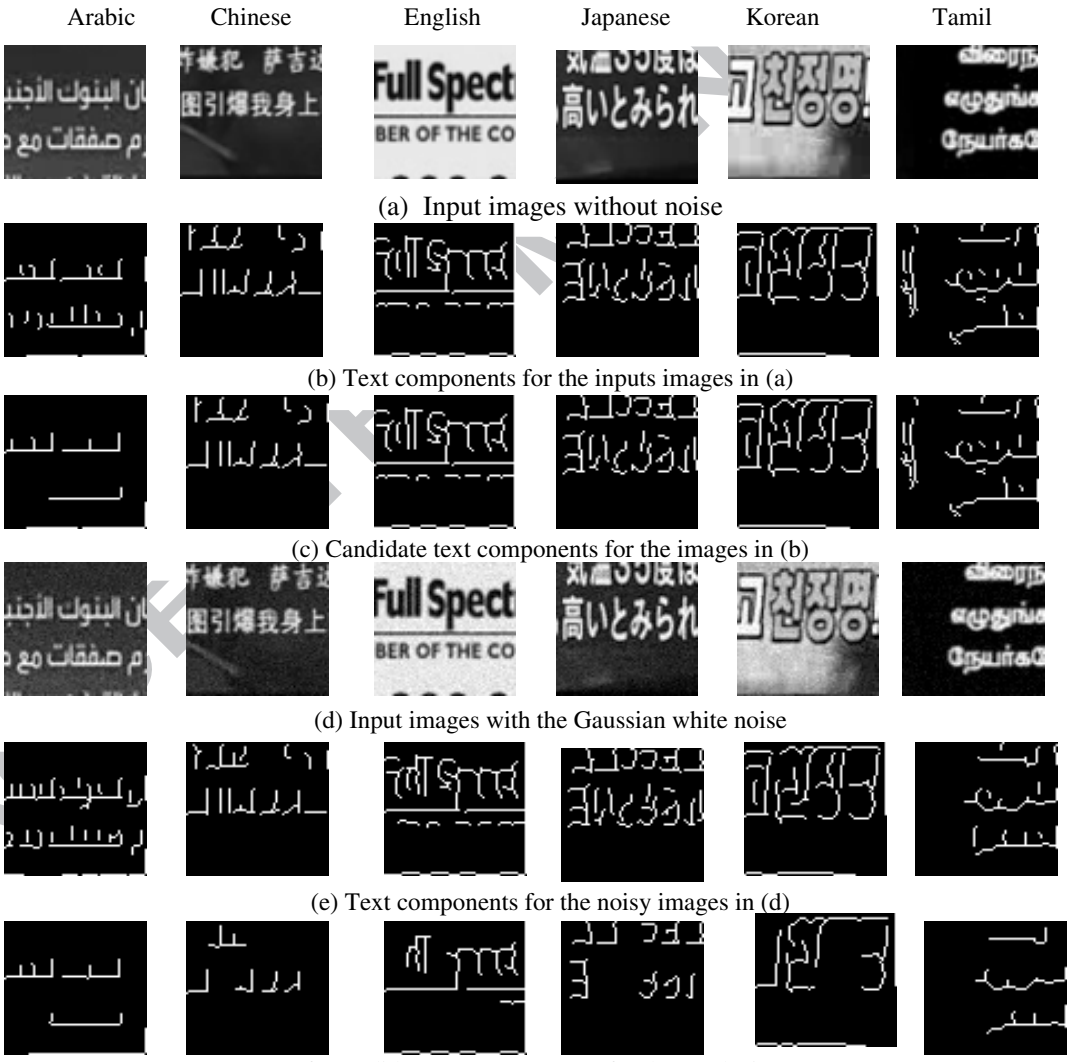
Table 7. Confusion matrix of the proposed integration GSSF method (in %)

Scripts	Arabic	Chinese	English	Japanese	Korean	Tamil
Arabic	**84.33**	07.53	04.10	00.00	03.00	02.61
Chinese	08.33	**80.70**	05.23	00.40	01.32	04.02
English	03.60	00.83	**93.31**	01.10	01.16	00.00
Japanese	02.92	01.95	12.52	**76.67**	04.88	01.06
Korean	12.51	00.82	00.00	06.43	**80.11**	00.13
Tamil	11.24	01.55	04.85	00.00	00.00	**83.36**

4.6. Experiments on Noisy Images of GSSF

Since the proposed method extracts features at component level and uses the candidate text components after eliminating false candidate text components, we believe our method should be robust to noise to some extent. In order to test the robustness of the proposed method, we introduce Gaussian white noises of mean 0 and variance 0.001 for the input text blocks of different scripts as shown in Figure 12(d). The text components and the candidate text components for the input images are shown respectively in Figure 12(b) and (c). In the same way, the text components and the candidate text

components for noisy images are shown in Figure 12(e) and (f), respectively. When we compare Figure 12(c) and Figure 12(f), we can see that there are few differences, except for the tiny difference on the classification rates. We calculate classification rate in the form of confusion matrix as reported in Table 8. It is noticed from Table 8 that the method gives 79.8% classification rate for noisy images after averaging the diagonal elements of the confusion matrix. This is lower than the accuracy 83.0% that is achieved for the images with no noises. Interestingly, we can observe from Table 8 that the method gives a good classification rate which is more than 80% for all the scripts as in Table 7 except the Japanese script, which gets even a low classification rate for the images without noises as reported in Table 7. Therefore, we can assert that the method does not lose much when we introduce noises. Thus, it is robust to noise to some extent. We can also note that introducing noises in case of video is rare and an exceptional case since it is not scanned by any device. However, noise can be introduced due to distortion and low resolution of video.



(a) Input images without noise

(b) Text components for the inputs images in (a)

(c) Candidate text components for the images in (b)

(d) Input images with the Gaussian white noise

(e) Text components for the noisy images in (d)

(f) Candidate text components for the results in (e)

Fig. 12. Sample results of the GSSF on noisy images

Table 8. Confusion matrix of GSSF on Noisy Images

scripts	Arabic	Chinese	English	Japanese	Korean	Tamil
Arabic	**82.5**	11.5	1.0	4.5	0.0	0.5
Chinese	8.3	**80.7**	3.4	2.4	0.4	4.4
English	9.0	1.1	**84.7**	0.0	1.1	3.9
Japanese	24.1	3.4	1.4	**68.0**	0.4	2.4
Korean	15.4	0.5	2.4	0.0	**80.5**	1.0
Tamil	12.5	0.0	3.0	2.0	0.0	**82.5**

4.7. Experiment on Scanned Document Images

To assess the effectiveness and the ability to extend to other scripts of the proposed soft integrated method, we conduct experiments on high resolution and low resolution scanned document images of seven scripts, namely, Arabic, Chinese, English, Japanese, Korean, Tamil and Thai. The dataset for this experimentation is chosen from the database used in [40] for script identification. For each script, we collect samples of scanned document images. Then we divide the whole document image into 64×64 sized blocks as we have done for video scripts identification in the previous experiments. We calculate classification rate at block level. The template formation for matching is the same as discussed in Section 3.6. The sample blocks for high resolution and low resolution of seven scripts are shown in Figure 13(a) and Figure 13(b), respectively. Low resolution scanned images are created by reducing the size of the original scanned document image to 1.5 time difference in size using MATLAB function. For instance, if the original size of the document image is 768×768 dimension, then for low resolution document image, the size is 512×512 dimension. It can be seen from Figure 13(a) and Figure 13(b) that the blocks representing a high resolution have less texts and bigger fonts compared to the blocks representing low resolution images. The number of the blocks used for this experimentation is 1250 for high resolution images and 834 for low resolution images. The confusion matrices for both the high resolution images and the low resolution blocks are shown in Table 9 and Table 10, respectively. The results reported in Table 9 and 10 reveal that the proposed integrated method is good for both the high and the low resolution scanned images as it gives a good classification rate for both the datasets. When we compare the classification rates by averaging the diagonal elements of the confusion matrices, the classification rate of high resolution images (94.3%) is slightly higher than the classification rate of low resolution images (93.1%). This is because of the loss of information due to the reduction in the size of the original input image. The classification rate of scanned document images is higher than video frames as scanned images have simple background and high resolution text compared to video frames. It is also noticed from the experimental results that the new script (Thai) and more scripts (7 scripts) do not affect the performance of the proposed method. Therefore, with this experimentation, we can conclude that the proposed integrated method

has the ability to handle additional scripts without sacrificing the classification rates of the existing scripts.

(a). High resolution blocks

(b) Low resolution blocks

Fig. 13. Sample blocks of high and low resolution scanned document images

Table 9. Confusion matrix of high resolution scanned documents (in %)

Scripts	Arabic	Chinese	English	Japanese	Korean	Tamil	Thai
Arabic	**97.0**	00.0	01.0	00.3	00.9	00.2	00.6
Chinese	01.1	**94.1**	01.2	02.3	00.5	00.6	00.3
English	01.2	00.7	**94.4**	01.5	01.4	00.9	00.0
Japanese	01.2	00.8	01.2	**91.2**	03.0	02.2	00.4
Korean	00.8	00.2	00.9	00.8	**95.6**	00.9	00.8
Tamil	00.5	00.6	00.9	00.0	00.6	**97.0**	00.4
Thai	00.6	00.0	00.9	00.0	00.4	00.2	**97.9**

Table 10. Confusion matrix of low resolution scanned documents (in %)

Scripts	Arabic	Chinese	English	Japanese	Korean	Tamil	Thai
Arabic	**97.4**	00.4	00.7	00.8	00.0	00.6	00.1
Chinese	00.2	**95.0**	00.8	01.3	01.0	00.9	00.8
English	00.0	01.2	**96.1**	00.9	00.9	0.00	00.8
Japanese	03.6	04.2	03.4	**84.5**	03.7	00.5	00.1
Korean	01.7	00.0	00.7	00.8	**92.2**	02.7	01.9
Tamil	02.2	02.1	01.2	01.6	01.8	**90.2**	00.9
Thai	02.2	00.0	00.0	00.1	00.6	00.8	**96.4**

5. Comparative Study

According to the literature review in the Section 2, most of the methods use SVM classifier for script classification and identification. But, in our work, we propose to use templates that are constructed based on average feature values of 50 sample blocks for each script chosen randomly for respective databases. Therefore, to show that our soft integration method is effective, we compare it with the

SVM classifier results. We use multi-class SVM classifier with RBF kernel for the same dataset which includes 50 blocks for each script for training the classifier, and 200 frames for each script for testing. The extracted gradient-spatial and gradient-structural features (21 feature vector) are passed to the SVM classifier for classification. We calculate the average precision and processing time for training and testing to evaluate the performance of the method. The results of the proposed soft integration method with template matching and with the classifier are reported in Table 11, which shows that proposed GSSF with the template matching gives a better average precision than with the SVM classifier. The main reason for getting a poor accuracy with the SVM classifier is the less number of samples for training. This is the main advantage of the proposed GSSF which does not require a greater number of samples to improve the accuracy as in the case of SVM, and it does not involve intensive computations for matching in contrast to SVM which requires more computations such as kernel processing and matching feature vector with different parameter settings. Furthermore, the proposed method with template matching can be adapted easily without much modification for different datasets and even by adding more script classes. On the other hand, SVM requires not only more samples for trainings but also requires more modifications in terms of kernel selection and parameter settings at the cost of computations. Therefore, the proposed method GSSF with template matching takes low processing times for training and testing compared to the SVM. Thus we can conclude that the proposed method with template matching is better than the proposed features with a SVM classifier.

Table 11. Comparative study of the proposed GSSF with the SVM classifier

Proposed Method	Average precision	Training time	Testing time
Proposed features with the SVM Classifier	80.4%	0.11s	38s
Proposed features with the Template Matching	83.0%	0.02s	1.41s

We also compare our GSSF method with the existing method [51] which is proposed for video script identification of English and Chinese using different texture features and the KNN classifier to show superiority of the method. We choose this existing method [51] for comparative study because this method addresses the videos script identification problem at text line level with the combination of texture and statistical features. This method considers only two scripts, namely, English and Chinese for experimentation. For a fair and comparative study, we consider the same two scripts for experimentation. Further, the existing method considers text lines as the input for script identification, while the proposed GSSF considers block as the input, therefore we run the existing method on blocks as our method does to identify the scripts at block level. The same testing data is used for experimentation. That is, 1200 frames which include 200 frames for each script. The classification rates for both the proposed method and the existing method on English and Chinese are reported in Table 12, where the proposed soft integration method gives a better classification rate compared to the existing method. The reason for poorer results for the existing method is that the extracted features are

not good enough to handle broken segments and the touching between adjacent components as these features expect some regularity of text pattern in each zone. The proposed soft integration method is capable of overcoming these problems because the candidate text component selection and the soft integration of gradient-spatial, gradient structural preserve the uniqueness of scripts in spite of broken segments and touching caused by low resolution and complex background. In addition, the existing method is sensitive to the classifier and samples. On top of this, our method works well for the six scripts under study. Thus the proposed integrated method is superior to the existing method in terms of classification rate and the number of scripts.

Table 12. Performance of the proposed soft integration method an existing method at block level (in %)

Proposed integration method			Gllavata and Freisleben [51]		
Scripts	Confusion matrix		Confusion matrix		
	English	Chinese	Scripts	English	Chinese
English	**84.29**	15.71	English	**65.58**	34.42
Chinese	14.98	**85.02**	Chinese	45.665	**54.335**

6. Conclusion and Future Work

We have proposed a new soft integration method which uses the gradient-spatial-structural-features for identifying six video scripts. The dominant text pixel selection is done based on the histograms on horizontal gradient and vertical gradient values of the input frames. Then we propose new features based on spatial and structural information of candidate text components in the blocks to identify the scripts. The experimental results on individual feature extraction methods and the combined method show that individual and combined methods are not good enough to achieve a good classification rate for all the six video scripts. On the other hand, the integration of the two feature extraction methods by exploiting the strengths of the two methods has been shown to be superior to simply combining them as one feature vector. The proposed soft integration method is tested on both high resolution and low resolution of scanned document images to show that the proposed method works well for both video as well as scanned document images. The performance of the integrated method is compared with a SVM classifier and an existing method at the block level to show that the integrated method with template matching is superior to the SVM classifier and the existing method. Our future work will include the recognition of scripts by exploring temporal information. And also, we are planning to investigate a new idea for improving skeletonization for noisy and distorted video. Furthermore, we will try to extend our method for European scripts such as German, French and Russia.

Acknowledgement

We are grateful to anonymous reviewers for their quality comments and suggestions enhance the quality, as well as clarity of the work in greatly. The work described in this paper was partly supported by the Natural Science Foundation of China under Grant No. 61272218 and No. 61321491, the 973 Program of China under Grant No. 2010CB327903, and the Program for New Century Excellent

Talents under NCET-11-0232. This research is also supported in part under Grant No.UM.TNC2/IPPP/UPGP/261/15 (BKP010-2013).

References

[1] N. Sharma, U. Pal and M. Blumenstein, "Recent Advances in Video Based Document Processing: a review", In Proc. DAS, 2012, pp. 63-68.

[2] J. Zang and R. Kasturi, "Extraction of Text Objects in Video Documents: Recent Progress", .In Proc. DAS, 2008, pp. 5-17.

[3] K. Jung, K. Kim and A. K. Jain, "Text information extraction in images and video: a survey", Pattern Recognition, 2004, pp. 977-997.

[4] D. Doermann, J. Liang and J. Li, "Progress in Camera-Based Document Image Analysis", In proc.ICDAR, 2003, pp. 606-61.

[5] V. Y. Mariano and R. Kasturi, "Locating Uniform-Colored Text in Video Frames", In Proc. ICPR, 2000, pp. 539-542.

[6] Q. Ye, Q. Huang, W. Gao and D. Zhao, "Fast and robust text detection in images and videos frames", In Proc. Image and Vision Computing, 2005, pp. 565-576.

[7] K. Jung, "Neural network-based text location in color images", Pattern Recognition Letters, 2001, pp. 1503-1515.

[8] H. Zhang, K. Zhao, Y. Z. Song and J. Guo, "Text extraction from natural scene image: A survey", Neurocomputing, 2013, pp 310-323.

[9] C. Yi and Y. Tian, "Text extraction from scene images by character apperance and structure modeling", Computer Vision and Image Understanding, 2013, pp 182-194.

[10] C. M. Thillou and B. Gosselin, "Color text extraction with selective mtric-based clustering", Computer Vision and Image Understanding, 2007, pp 97-107.

[11] R. Minetto, N. Thome, M. Cord, N. J. Leite and J. Stolfi, "SnooperText: A text detection system for automatic indexing of urban", Computer Vision and Image Understanding, 2013 (article in press)

[12] C. M. Gracia, M. Mirmehid and J. L. G. Mora, "Fast persepective recovery of text in natural scenes", Image and Vision Computing, 2013, pp 714-724.

[13] L. Rong, W. Suyu and Z. Shi, "A two level algorithm for text detection in natural scene images", In Proc. DAS, 2014, pp 329-333.

[14] P. Zhou, L. Li and C. L. Tan, "Character recognition under severe perspective distortion", In Proc. ICDAR, 2009, pp 676-680.

[15] Y. Zhou, J. Field, E. L. Miller and R. Wang, "Scene text segmentation via Inverse rendring", In Proc. ICDAR, 2013, pp. 457-461.

[16] D. Chen, J. M. Odobez and H. Bourlard, "Text detection and recognition in images and video frames", Pattern Recognition, 2004, pp. 595-608.

[17] D. Chen and J. M. Odobez, "Video text recognition using sequential Monte Carlo and error voting methods", Pattern Recognition Letters, 2005, pp. 1386-1403.

[18] K. L. Bouman, G. Abdollahaian, M. Boutic, and E. J. Delp, "A Low Complexity Sign Detection and Text Localization Method for Mobile Applications", IEEE Trans. Multimedia 13, 2011, pp 922-934.

[19] A. Hartl and G. Reitmayr, "Rectangular Target Extraction for Mobile Augmented Reality Applications", In Proc. ICPR. 2012, pp 881-84.

[20] A. K. Jain and B. Yu, "Automatic Text Location in Images and Video Frames", Pattern Recognition, 1998, pp. 2055-2076.

[21] V. Wu, V. Manmatha and E. M. Riseman, "Text finder: An automatic system to detect and recognize text in images". IEEE Trans. on PAMI, 1999, pp. 1224-1229.

[22] K. L. Kim, K. Jung and J. H. Kim, "Texture-Based Approach for Text Detection in Images using Support Vector Machines and Continuous Adaptive Mean Shift Algorithm", IEEE Trans. on PAMI, 2003,pp. 1631-1639.

[23] P. Shivakumara, T. Q. Phan and C. L. Tan, "A Laplacian Approach to Multi-Oriented Text Detection in Video", IEEE Trans. on PAMI, 2011, pp. 412-419.

[24] P. Shivakumara, R. P. Sreedhar, T. Q. Phan, S. Lu and C. L. Tan, " Multi-Oriented Video Scene Text Detection through Bayesian Classification and Boundary Growing", IEEE Transactions on CSVT, 2012, pp. 1227-1235.

[25] N. Sharma, P. Shivakumara, U. Pal, M. Blumenstein and C. L. Tan, "A New Method for Arbitrarily-Oriented Text Detection in Video, In Proc. DAS, 2012, pp. 74-78.

[26] P. Shivakumara, T. Q. Phan, S. Lu and C. L. Tan, "Gradient Vector Flow and Grouping based Method for Arbitrarily-Oriented Scene Text Detection in Video Images", IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), Vol. 23, No. 10, pp 1729-1739, 2013.

[27] Y. Liu, Y. Song, Y. Zhang and Q. Meng, "A Novel Multi-Oriented Chinese Text Extraction Approach from Videos", In Prod. ICDAR, 2013, pp 1355-1359.

[28] L. Wu, P. Shivakumara, T. Lu and C. L. Tan, "Text Detection using Delaunay Triangulation in Video Sequences", In Proc. DAS, pp 41-45, 2014.

[29] C. Shi, C. Wang, B. Xiao, S. Gao and J. Hu, "End-to-end scene text recognition using tree-structured modesl, Pattern Recognition (to appear).

[30] T. Q. Phan, P. Shivakumara, S. Tian and C. L. Tan, "Recognizing text with perspective distortion in natural scenes", In Proc. ICCV, 2013, pp 569-576.

[31] L. Li, W. Goh, J. H. Lim and S. J. Pan, "Extended spectral regresssion for efficient scene recognition", Pattern Recognition (to appear).

[32] T. Q. Phan, P. Shivakumar, T. Lu and C. L. Tan, "Recognition of video text through temporal integration", In Proc. ICDAR, 2013, pp 589-593.

[33] S. Roy, P. P. Roy. P. Shivakumara and U. Pal, "Word recognition in natural scene and video images using Hidden Markov Model", In Proc. NCVPRIPG, 2013, pp 1-4.

[34] Tesseract. http://code.google.com/p/tesseract-ocr/.

[35] D. Ghosh, T. Dube and A. P. Shivaprasad, "Script Recognition-Rview", IEEE Ttansactios on PAMI, 2010, pp 2142-2161.

[36] T .N. Tan, "Rotation Invariant Texture Features and Their Use in Automatic Script Identification", IEEE Transactions on PAMI, 1998, pp 751-756.

[37] A. Busch, W. W. Boles and S. Sridharan, "Texture for Script Identification", IEEE Transactions on PAMI, 2005, pp 1720-1732.

[38] L. Shijian and C. L. Tan, "Script and Language Identification in Noisy and Degraded Document Images", IEEE Transaction on PAMI, 2008, pp 14-24.

[39] G. D. Joshi, S. Garg and J. Sivaswamy, "A generalized framework for script identification", IJDAR, 2007, pp 55-68.

[40] S. Lu, L. Li and C. L. Tan, "Identfication of scripts and orientation of degraded document images", Pattern Analysis and Applications, 2010, pp 469-475.

[41] S. Jaeger, H. Ma and D. Doermann, "Identifying Script on Word-Level with Informational Confedence", In Proc. ICDAR, 2005, pp 416-420.

[42] P. B. Pati and A. G. Ramakrishnan, "Word level multi-script identification", Pattern Recognition Letters, 2008, pp 1218-1229.

[43] S. Chanda, S. Pal, K. Franke and U. Pal, "Two-stage Apporach for Word-wise Script Identification", In Proc. ICDAR, 2009, pp 926-930.

[44] S. Chanda, O. R. Terrades and U. Pal, "SVM Based Scheme for Thai and English Script Identification", In Proc. ICDAR, 2007, pp 551-555

[45] L. Li and C. L. Tan, "Script Identification of Camera-based Images", In Proc. ICPR, 2008.

[46] A. M. Namboodiri and A. K. Jain, "On-line Script Recognition", In Proc. ICPR, 2002, pp 736-739.

[47] S. Ghosh and B. B. Chaudhuri, "Composite Script Identification and Orientation Detection for Indian Text Images", In Proc. ICDAR, 2011, pp 294-298.

[48] R. Bashir and S. Quadri, "Identification of Kashmir script in a bilingual document image", In Proc. ICIIP, 2013, pp 575-579.

[49] R. Rani, R. Dhir and G. S. Lehal, "Script identification of pre-segmented multi-font character and digits",In Proc. ICDAR, 2013, pp 1150-1154.

[50] M. A. Ferrer, A. Morales and U.Pal, "LBP based line-wise script identification", In Proc. ICDAR, 2013, pp 369-373.

[51] J. Gllavata and B. Freisleben, "Script Recognition in Images with Complex Backgrounds", In Proc. IEEE International Symposium on Signal Processing and Information Technology, 2005, pp 589-594.

[52] N. Sharma, S. Chanda, U. Pal and M. Blumestein, "Word-Wise Script Identification from Video Frames", In Proc. ICDAR, 2013, pp 867-871.

[53] P. Shivakumara, N. Sharma, U. Pal, M. Blumenstein and C. L. Tan, "Gradient-angular-features for word-wise video script identification", In Proc.ICPR, 2014 (to appear).

[54] T. Q. Phan, P. Shivakumara, Z. Ding, S. Lu and C. L. Tan, "Video Script Identification based on Text Lines", In Proc. ICDAR, 2011, pp 1240-1244.

[55] D. Zhao, P. Shivakumara, S. Lu and C. L. Tan, "New Spatial-Gradient-Features for Video Script Identification", In Proc. DAS, 2012, pp 38-42.

[56] P. Shivakumara, A. Dutta, Trung Quy Phan, C. L. Tan and U. Pal, "A Novel Mutual Nearest Neighbor based Symmetry for Text Frame Classification in Video", Pattern Recognition, 2011, pp 1671-1683.

[57] Y. Terada, R. Huang, Y. Feng and S. Uchida, "On the possibility of structure learning –based scene character detector", In Proc. ICDAR, 2013, pp 472-476.

[58] R. Huang, P. Shivakumara, Y. Feng and S. Uchida, "Scene character detection and recognition with cooperative multiple-hypothesis framework", Transactions on Information and Systems (IEICE), 2013, pp 2235-2244.

[59] R. Huang, P. Shivakumara and S. Uchida, "Scene character detection by an Edge-ray filter", In Proc. ICDAR, 2013, pp 462-466.

[60] P. Shivakumara, A. Dutta, C. L. Tan and U. Pal, "Multi-Oriented Scene Text Detection in Video based on Wavelet and Angle Projection Boundary Growing", Multimedia Tools and Applications (MTA), Springer-Verlag, 2013.

[61] R. E. Schapire and Y. Singer, "Improved Boosting Algorithms Using Conidense-rated Predictions", Machine Learning, 1999, pp 297-336.

Highlights

Dominant pixel selection by exploring gradient information with histogram

Proposing Gradient-Spatial-Features for text candidate selection

Proposing Gradient-Structural-Features in new way for text candidate selection

Determining weights based on templates for correct classification

Integrating Gradient-Spatial-Structural-Features in novel way for classification