Available online at www.sciencedirect.com

ScienceDirect

Pattern Recognition 00 (2022) 1–30

Procedia Computer Science

Logo Localization of Administrative Documents

Tanmoy Mondal[a], Alireza Alaei[b], Mickael Coustaty[a]

tanmoy.mondal@univ-lr.fr, mickael.coustaty@univ-lr.fr, alireza20alaei@yahoo.com

[a]*L3i, Université De La-Rochelle, La-Rochelle, France*
[b]*School of Information and Communication Technology, Griffith University, Queensland, Australia.*

Abstract

In word spotting ..

Keywords: Logo Detection, Document image retrieval, Tobacco dataset

1. Introduction

The advancement of high quality digitization technologies and the pervasiveness of digital devices, such as high quality document scanners and portable digital devices, has facilitated the transformation of various document based information into electronic formats. Consequently, the quantity of digital born documents have increased rapidly in a massive scale. Many libraries, museums and archives have shown their high interest in mass digitization and transcription of their collected books, manuscripts and resources. Digitization of these manuscripts and various documents has not only facilitated an easy and hassle free access but also has facilitated to preserve them for a longer period. Administrative, bureaucratic and organizational documents for communication and filing procedures, which were mostly paper-based in the past, are driven into a digital environment by the ubiquity of different computation facilities. In all these applications, the objectives are to preserve documents in a digital format, to encourage reduction of the use of papers and to speedup the whole administrative process by improving accessibility and accountability in bureaucracy.

To provide an easy access and retrieval service to a wider number of users, these digitized documents need to be processed automatically. There exists several state-of-the-art textual analysis systems for automatic document retrieval. These records mostly follow certain organizational document structures and are usually typewritten. These kinds of documents are generally indexed by using commercial Optical Character Recognition (OCR) products. After the application of OCR technique, ASCII information of the characters present in the document is obtained and are then used for indexation. These ASCII information then also can be compared to the query by using a string-matching algorithm. However, the commercial OCR generally fails to perform accurately on the documents with a high degradation, such as fax and scanned documents (due to scanning and transmission noise). It also fails to correctly analyze/comprehend document structure. Another stream of indexation techniques available in the literature is analyzing graphical contents present inside documents. Content Based Image Retrieval (CBIR) is one of such technologies employed not only in document images but also has a wide range of applications, such as medical, landscape, and satellite images. Content-based Document Image Retrieval (CBDIR) is a subdivision of CBIR that involves a search process, where the users mention a query in terms of a selected ROI or an image/graphics and all the document images which contains this ROI are retrieved as results. Trademark image retrieval (TIR) is also a branch of CBIR [25] and many studies have been carried out in this particular area. Review of the techniques related to trademarks is out of the scope of this article and is not included in this paper.

To improve the efficiency and effectiveness of services and operations, organizations have shown their interests in implementing digital mail-rooms. The objective is reducing the burden of manual processing of different administrative documents, including incoming mails, faxes, forms, invoices, reports, employee records, health record, etc. One of the primary functionality of these digital mail-room is to have an automatic indexation of the incoming documents that results in automatic classification, distribution, easy access and retrieval of these documents in future. As mentioned earlier, one possible solution is the use of textual information for automatic indexation of these administrative documents. However, besides the presence of textual information, administrative documents often contain different salient entities, such as logos, stamps/seals, layout-structure, signatures, and bar-codes, which corresponds to the organization, institute, product or personnel. These salient entities have rich contextual representations and provides distinctive features and characteristics for Administrative Document Image Retrieval (ADIR) and document classifica-

tion. Therefore, many research works have been carried out to automatically detect logos/seals facilitating such administrative document-based systems.

Logos detection and/or recognition inside the document images is a challenging task as logos are often composed of quite complex symbols, graphical and textual components. In this article, we will mainly focus on the detection of logos in administrative documents. There are some specific issues related to logos that make the detection/recognition of them more challenging. For instance, logos can be placed at different locations within an administrative document and the document can be rotated hence the logo would also be rotated. Missing of some parts due to improper scanning is another issue in logo detection. Generally, in logo based administrative document retrieval detection step might take place before or along recognition step [9] [10]. Therefore, in this paper we have presented a novel approach for logo detection, which have been tested on two datasets containing images of administrative documents.

The rest of the paper is organized as follows. In Section 2, some specific characteristics of logos are drawn. In Section 3, an overview of the works in relation to logo detection is provided. Benchmarks and datasets used for experimentation and analysis of the results are presented in Section 5. Discussion and remarks are provided in Section 6. Finally, conclusion and future directions are drawn in Section 7.

2. Properties of Logo

Companies, institutes, and organizations use logo as a unique sign to pinpoint their products or services to consumers and help customers to localize and remember organizations. Logos can also be used as the means of authentication/verification. Moreover, logos bear some specific symbolism to convey some message about the enterprise and their principal endeavor. There are various styles of logos available but logos are bounded by certain design constraints as they need to be easily identified by human beings. A logo may be completely graphical or textual or the combination of both graphical and textual components. Some instances of such logo types are shown in Fig. 1.

The text in a logo is often modified for its aesthetic appealing and as a result, its segmentation for the OCR processing may not be easy. Thus, the text can be viewed as a part of the logo, which needs to be handled with other graphical components by a general shape analyzer. Similarly, if a logo contains texture patterns, the texture patterns can be treated as a graphical pattern and can be processed together with other

(a) (b) (c) (d) (e)

Figure 1: Some examples of logos in administrative documents extracted from Tobacco dataset [18] [2] [1].

logo components. Sometimes, logos are designed with a set of color combinations then the color information could further be used for logo detection/recognition. Colors may sometimes be ignored as far as the unique identity of a logo (represented as an intrinsic graphic pattern) is concerned [22][6]. Furthermore, logos generally appear on a specific positions at the very top (or rarely at the foot) of administrative documents /forms of organizations in a printed or digitized format to be easily distinguishable. These specific characteristics make logos more distinctive than other graphical entities present inside a document. Considering these properties, different types of approaches, e.g. signature, radon transform, and structural based approaches, have been proposed in the literature for logo detection, recognition and retrieval [33] [37] [38] [32]. In this article, we have mainly focused on the logo detection technique and that's why we have discussed here only the related literature of logo detection in document image. A brief description of such techniques are presented in the Section 3.

3. Literature review of Logo Detection Techniques

Logo based document image retrieval basically involves two tasks: the detection of logos in a document followed by the recognition of the detected logos [14]. Considering these two related tasks the literature reviews on this topic can be divided into two parts; one is related to Logo detection and other one is pertinent to Logo recognition. Due to the advancement of image recognition techniques, the logo recognition, which can be thought as one category of image recognition problem, several researchers have attempted logo recognition problem more and earlier to the problem of logo detection on document images [15] [11] [20] [19] [30] [8].

For logo detection in document images a number of techniques have been proposed in the literature. Based on the used approaches, we have categorized these techniques into four main categories : (i) Con-

nected component based approaches, (ii) Window-sliding (block) based approaches, (iii) Recognition based detection approach, and (iv) Local descriptors based approach. In the following subsections, succinct descriptions of state-of-the-art techniques for the logo detection are provided.

3.1. Connected Component Based Approaches

In connected component based techniques for localizing logos in document images, initially, a document image is binarized and then connected component labeling process is performed to obtain a list of connected components [12] [19] [31] [23] [13] [37]. Often the individual connected components e.g. individual characters, parts of graphics etc. are merged by using Horizontal Run Length Smoothing Algorithms (H-RLSA) and/or Vertical Run Length Smoothing Algorithms (V-RLSA) [34] to obtain complete entities e.g., words, logos and graphical symbols etc. Various features, such as convex hull, width, height, aspect ratio, area and density [4] [23] [19], extracted from connected components are used to classify each connected as a probable logo candidate or otherwise. Some notable classifiers, such as Decision Tree [4] [13] [12], and Bayesian classifier [35], have been used for the classification purposes.

The technique presented in [19] uses a novel anchor line based approach for the detection and recognition of logo in document image. In the training phase by considering only one instance of each logo model, the logo region in the document image is selected. Then, a set of convex hull based features from the logo region are extracted and saved in a database along with the normalized logo image cropped from the document image. All the features are indexed using a hyper-cubic structure, which leads to a fast matching process. In the detection and recognition phases, connected component labeling is performed on the image and then convex hull based features are matched with the features of logos, saved in the database. In order to check the consistency of pairwise distances between feature candidates and logo prototypes, the concept of anchor line is further used in Li et al.[19]. Using a consistent mapping of the features extracted from a testing image, it derives the position of a logo by a bounding box inside input image as the probable logo candidate. This probable region of the logo is then normalized and matched with the saved logo prototypes to ensure a correct matching. No clear description is given for generating connected component of the entire logo region and also segmenting out the text lines, consisting of several isolated characters, representing each isolated connected components. Moreover, when the features extracted from all the connected components existing in a document and are matched with the features of logo, saved in database, there will

be many spurious matches. It is not clear that how the authors have distinguished between good matches against the spurious ones.

Another similar approach is proposed in [23], where the document images are first preprocessed by applying a skew correction followed by a morphological dilation technique and then a binarization. After segmenting out the text lines, the contours of the connected components are detected. The outer contours of the connected components are then grouped together to create outer contour strings (OCSs). An OCS is a sequence of outer contours that are located side by side from left to right. Coarse and finer layers of features are extracted from each OCS. The coarse feature considers graphical information and domain about OCSs, such as OCSs position and size, whereas finer features characterize contour regions using gradient based representation, computed directly from the gray scale images. After extracting these features, Gentle Boost [10] classifier is used for classifying any particular OCS as a logo or a non logo candidates. Due to the segmentation errors, the region of a complete logo can be divided into several parts. To correct this issue, the OCSs are grouped by using probability scores and neighboring criteria between two isolated OCSs. Proper binarization in the presence of noise, occlusions and correct segmentation of text lines are the critical issues of this approach. Moreover, there are several parameters heuristically fixed in this method that make it hard to generalize this technique for other databases.

A layout independent and segmentation free logo detection method using multi-scale boosting strategy is proposed in [37]. An initial two-class classification is performed using Fisher classifier at a coarse image scale on each connected component to discriminate logo regions from non-logo regions. Then, the probable candidate logo region is successively classified at finer image scales by a cascade of classifiers. However, several parameters to be set for using these classifiers that makes this approach dataset dependent. The training and testing sets have not been clearly mentioned in this paper. Moreover, the accuracy and precision of this technique is far from attaining the best values.

Another very similar kind of technique as the one presented in [23] is proposed in [12]. This approach also starts employing a noise reduction followed by an image binarization, and horizontal dilation for joining the neighboring components. Features are then extracted from the connected components and a decision tree classifier is used for classifying a connected component into either a logo or a non-logo candidate. This approach is a quite basic one and there are other improved approaches, which has achieved better

performance such as the one mentioned in [4].

The approach proposed in [13] can be divided into two primary steps. First, a foreground pixel is taken as a seed with a 3×3 window as a feature rectangle candidate. Then, this feature rectangle candidate will grow with boundary extension until a final feature rectangle is obtained or the growing process is stopped. All objects as a whole embraced by the generated feature rectangles are logo candidates. Second, a decision tree classifier is employed to quickly discard the candidates whose likelihood to be logos are very small. Through this two stage processing, most false positives are removed from the logo candidate pool, leaving only a small number of logo candidates to be verified with a logo database for further logo recognition or verification stage. However, one of the primary issue with this method is the need of an already prepared dataset for not only recognizing logos but also for detecting logos. The technique for searching a probable position of a logo is quite rigorous and not definitive, which succumbs to generate many false positives. Moreover, the experimental results are not well explained and not even comparable with other techniques.

3.2. Recognition approach based logo detection

As a new category of logo detection techniques, a unified framework for the detection and recognition of logos in document images has been proposed in [35]. In this work, author has improved the technique, mentioned in [13] by using a Bayesian Belief Network [9]. This approach also relies on candidate pool generation of Feature Rectangles (FR), based on the connected components which are pruned later. In order to make the module dynamic and interactive, the merging and cutting of FR's is also introduced. During this step-by-step framework manipulation process, FR based located candidate regions undergoes an adjustment. The boundary of these rectangles can expand or can also contract. This approach is a multilevel staged step-by-step recognition process. At each step, FR based connected components are labeled and then signature curves are extracted. As mentioned before that the shape of FR evolves also with merging and/or cutting of some neighboring FR. At each step, the signature curves are extracted of the newly evolved logo and then it is matched with the stored signatures of the logos, present in the database. Later, verification is performed based on the global invariants, which is an essential step for the recognition of logos. This approach shows better performance and it's accuracy is comparable with the best performing techniques for logo detection [4] [19].

3.3. Sliding window based logo detection

In this category of logo detection techniques, local detectors/descriptors, such as density, probability and blurred shape have been used [24] [29] [3] [4]. These features/descriptors are computed from the regions of interest or patches extracted by a sliding window, a connected component analysis or a segmentation technique. In this sliding window based approach [24], the spatial density of foreground pixels in each window is calculated by using mountain function. Then a window is classified based on a threshold value as logo or non logo.

Inspired by the approach mentioned in [24], a similar technique is proposed in [3]. This pipelining process starts by binarizing a gray scale image followed by reduction of image size for faster computation. Then, noisy pixels and texts are eliminated from the normalized image and then the image is transformed under stippled lines or isolated points for preserving the object compactness. The reduction of image size is realized by dividing the image into blocks and then transforming the blocks into a single pixel whose monochrome color is decided based on the dominance of either black or white color in the block. Then a morphological dilation is applied to join neighboring components followed by employing a sliding window based approach to discern the feasibility of being a logo or a non-logo region. The feasibility of being a logo or non-logo region is decided based on the foreground pixel's density, calculated using mounting function, which is also been adapted to determine regions of strong colorimetric uniformity. The experimental protocol and dataset used for performance evaluation are not comparable with the state of the art techniques, such as [19].

The work described in [29] is based on the computation of Blurred Shape Model (BSM) descriptors as features. In order to localize a logo within a document image, a sliding window approach is proposed to compute a normalized two-dimensional cross correlation between the BSM description of a model logo and the BSM description of a document. A peak value should be formed at a location in the document where it has a high probability to be similar to a given logo model. This process is repeated for each logo in the knowledge database and the peak having the highest response would be the better match between a specific zone of the document and logo model, which intrinsically define the class of the document as well. In order to increase the robustness of the method, the normalized cross correlation is further calculated for both the BSM description and inverse of the BSM description. In the final step, the probability maps obtained from

both the normalized cross correlations are multiplied/combined to get rid of the background noise. The advantage of this approach is that it can not only have the class of a document but also the location within a document where a logo may appear. However, key points and features for matching should be calculated off-line, otherwise, this approach is computationally expensive for locating the logos in a full document image. Moreover, the dataset used by the author is not a public one, so the results and experimental protocol are not comparable.

3.4. Logo detection technique based on local descriptors

In this category of systems, the logos are detected based on the detection of key-points followed by the extraction of local descriptors. Local key-point detectors, such as Hessian [14], Harris-Laplace [27], Difference-of-Gaussian (DoG) filter [17], Canny edge and He & Yung detectors [38], are employed at pixel level to extract a set of key points. The extracted key points are then exploited by local descriptors, such as Shape Context (SC) [27], [38], [28], Scale Invariant Feature Transform (SIFT) [29] [14] [17] [16], Binary Robust Independent Elementary Features (BRIEF) [16] and Speeded Up Robust Features (SURF) [14].

The technique mentioned in [14] detects Hessian key points and describe these key points based on SURF descriptors. The SURF descriptor of the patch around each key point is calculated by first equally subdividing a given patch into 4×4 grid. For each subsection, the Haar wavelet response D_x and D_y are calculated in the x and y directions, respectively. To fasten the matching process, feature level indexing is performed to group feature vectors that are distinct along the same dimensions together. A two-step approach is used which take into account at first the orientation information provided by interest points, found using fast Hessian detector. As the second step,a triangle based stricter filter is used to find a resemblance between query and document image feature points. Another local descriptor based logo detection technique is attempted by [27], in which the logo are represented by a local descriptor applied to a set of previously extracted key-points. The key-points are extracted by using Harris-Laplace corner detector. Harris-Laplace corner detector extracts points with high curvatures. The scale of the region is automatically selected to compute the local descriptor. Shape descriptor is further calculated from each key-point to characterize the region. The matching score between key-points extracted from a document and the ones of a logo model is calculated. A bag of visual word model is further used for logo detection in [27]. This approach is not comparable with the results of state-of-the-art techniques for logo detection.

In [16], key points are extracted and described by Scale Invariant Feature Transform (SIFT) descriptor for both a query document image and a given set of logos (logo database). In [16], BRIEF descriptors are are used as features. Then the key points in the SIFT feature space are matched using nearest neighbor rule. The key-points are further filtered using BRIEF descriptor. Then, the matched key-points are clustered by using Density Based Spatial Clustering of Application with Noise (DBSCAN). A cluster containing the maximum number of matched key-points is considered as logo region candidate. Finally, in the decision step, a normalized accumulating histogram is computed to measure dissimilarity between a logo region candidate and each logo in the gallery. Incorrectly matched key-points are rejected by using homography based RANSAC technique. Although this approach has shown some promising results but due to the use of same pipelining techniques as the one done in [27], this approach is not suitable for real time application. Some speed enhancement can be achieved by calculating and indexing the features beforehand. Hence, this approach has high dependency on database and it is difficult to be adapted on new databases.

In [38], authors have proposed the approach for logo detection/matching by extracting corner features. At first, object contours are obtained from the edge image extracted by the Canny edge detector and then the gaps along the contours are filled. Corners are detected from the contour image. The dissimilarity between two shapes are calculated by using four separate dissimilarity techniques such as: Thin Plate Spline Bending Energy, Shape Context Distance, amount of anisotropic scaling between two shapes and another distance measure based on the registration residual errors under the estimated non rigid transformation. An overall shape distance is then calculated by a weighted sum of these separate distance measures. Although, it has shown some promising performance for logo matching but could not achieve the superior performance in comparison to other state-of-the art techniques, e.g. [4] [19].

4. Outline of the Proposed Approach

The approach is outlined in the following section, where initially a method is proposed to automatically choose training documents to train the proposed system. The proposed approach is then tested over a test set. Before going into the details of the proposed approach, the dataset used are described below to understand the nature and characteristics of data used for experiments.

4.1. Dataset Description

For evaluation of the proposed approach 3 different datasets are used. First dataset is Tobacoo-800, which contains 1292 documents of which 412 contain 432 logos [18] [2] [1]. This dataset is publicly accessible document image collection with a realistic scope and complexity, which is important for the document image analysis. A significant percentage of Tobacco-800 dataset are consecutively numbered multi-page business documents which makes it valuable testbed for various content based document image retrieval approaches. The resolution of documents in Tobacco-800 vary significantly from 150 to 300 DPI and the dimensions of images range from 1200×1600 to 2500×3200 pixels.

The second dataset is IteSoft-1 used in [29]. This document consist of 3000 documents of which all of them contains between 1 and 6 logos. Number of classes of documents are reported to be 204.

Third dataset is the Itesoft-2, which is recently provided by Itesoft company in collaboration with Poly-Tech Tours, France. This dataset is composed 8200 real-life document images of which 5748 images contain between 1 to 8 logos. There are 1274 number of classes of logos exists in this dataset.

4.2. Automatic Generation of Training and Testing sets

The ground truth of the Tobacco-800 dataset is generated by following certain configuration. There are a total 36 ($\mathfrak{p}$) different classes of logos present in the database and each class contains multiple instances ($\mathfrak{r}$). Let's denote the classes of logos by ($\mathfrak{c}_{\mathfrak{p}}^{\mathfrak{r}}; \mathfrak{p} = 1, 2, ..., 36; \mathfrak{r} = 1, 2, 3, ...$). The naming convention used for all the cropped logos based on the ground truth information is "$\mathfrak{p}\_\mathfrak{r}$.*png*". The idea is to automatically generate a bias-less and repetitive training set for training the proposed system. Moreover, the training set should contain approximately an equal number of examples, at least one sample, from every class of logos. To achieve this objective, an algorithm for the automatic generation of training set from the dataset corpus is proposed. Two parameters are taken from the user. The first one is the "*number of training samples*" needed ($\mathfrak{T}$) and the second one is the *"minimum number of instances present in every logo class ($\mathfrak{E}$)"* for being considered to participate in the training set. From the Ground Truth (GT) of the Tobacco-800 database, logo images are cropped and manually labeled by using the following naming convention "$\mathfrak{p}\_\mathfrak{r}$.*png*". The following mentioned Algorithm 2 describes the automatic technique of generating training data. At Line-1, the logo class name ($\mathfrak{p}$) and the instance number ($\mathfrak{r}$) are obtained from the image name "$\mathfrak{p}\_\mathfrak{r}$.*png*". Then a Hash-Map data structure ($\mathfrak{H}$) is used for organizing each logo instances (organized by considering instance

number ($\mathfrak{r}$) as Hash-Map values against logo class which is considered as Hash-Map key ($\mathfrak{p}$). In Line-4, one vector data-structure ($\mathfrak{U}$) of the size of Hash-Map ($\mathfrak{H}$) is considered. This vector contains another vector of integers. Line-5 says that run the while loop until the number of training samples mentioned by user are not gathered. For the very initial iteration (Line-6), i.e. when $iCnt = 0$, loop through all the logo classes by looping over the keys of Hash-Map $\mathfrak{H}$ (Line-8). Get the logo class name (Line-9; $\mathfrak{H}(i).first$) and all the instances of this particular class (Line-9; $\mathfrak{H}(i).second$). Then generate a random number between 1 to total number of instances of this particular class (Line-11). Keep this random number ($\mathfrak{r}$) in the vector $\mathfrak{R}$ and then put the vector $\mathfrak{R}$ in the vector $\mathfrak{U}$. Consequently, get the particular image name and keep it as the first training image.

Now from the second time onwards, the control will always enter in the else part (Line-21) as now $iCnt > 0$. In the same fashion, loop through Hash-Map entries (Line 22 to 24) and obtain the logo class name (Hash-Map key; $\mathfrak{H}(i).first$) and corresponding instances (Hash-Map value; $\mathfrak{H}(i).second$). Line-25 obtains a vector containing indexes of the instances, which have already been employed/considered as training samples from this particular logo class ($\mathfrak{U}.get(eCnt)$). The idea is to gather training samples in a cyclic manner. One cycle completes by randomly considering one instance from each class of logos in the training set. After completing first cycle the next cycle initiates. This process continues until all the desired training samples are not gathered. The next task is to check whether we have not surpassed the limit of taking training instances from a particular class. This check is performed first by subtracting $\mathfrak{E}$ from the total number of instances ($allInstances.length()$) of a particular logo class. This subtraction gives the total number of instances that can be considered from this class in the training set. The "IF" condition in Line-26 checks whether the number of elements in $\mathfrak{Y}$ is less than or equal to the aforementioned subtracted value. The second "IF" condition in Line-26 simply checks whether the total number of instances under a particular logo class is more than the user defined parameter ($\mathfrak{E}$). If both conditions satisfied, then we try to randomly find another instance (index of the instance) which have not been yet considered under this particular logo class. This is done by continuously running a while loop (Line-28) until we obtain another unique random number which is not present already. This process is continued for all the logo classes and the cycles are repeated until we obtain the required number of training samples to exit from the loop (Line 35-36).

So, by following the above mentioned approach devised in Algorithm 2, the training set is repetitively and automatically generated. Thanks to this automatic generation of training set that allows us to carry out several experiments in order to evaluate the proposed logo detection system.

4.3. Image Segmentation Technique

We have adapted a technique called as Piece-wise Painting Algorithm for the segmentation of text and graphics regions. This approach is originally proposed in [5] for text-line segmentation. Here, we have adapted this approach for the purpose of logo region localization. The algorithm starts by dividing the image into a number of vertical segments called stripes from the left to right direction. The direction of traversing an input image (from left to right or from right to left) does not affect the performance of the proposed algorithm. In this work, the stripe width ($\mathfrak{W}$) is considered as 5% of the document image width. Since the image is divided from the left to right direction, the width of the last stripe may become smaller or larger than other stripes. To simplify this problem, the width of all other stripes (1^{th} to $(\mathfrak{P}-1)^{th}$) are kept same except last stripe ($\mathfrak{P}^{th}$). The process of getting each stripe from the image is mentioned in the following Algorithm 1.

$$\text{Stripe Width}(\mathfrak{W}) = \frac{\mathtt{I}.Cols \times 5}{100}; \text{No. of Possible Stripes}(\mathfrak{P}) = \frac{\mathtt{I}.Cols}{\mathfrak{W}}; \\ \mathtt{I}.Cols = \text{No. of columns present in the image} \tag{1}$$

After dividing each image into stripes (see Fig. 2a), the stripe image is complemented (see Fig.2b) and the small noises, e.g., dots and/or isolated tiny foreground pixels, are removed by applying a morphological closing followed by a morphological opening operations. The gray value of each pixel (either 0 or 255) in each row of a stripe is modified by changing it with the average gray value of all pixels present in that particular row of a stripe. The Gray Level Mean (GLM) value of each row in every stripe is calculated by the following Equation 2.

$$GLM_{i,k} = \frac{\sum_{j} \mathtt{ImgStripe}_{ij}}{\mathtt{stripeWidth}_{\kappa}}; \quad \kappa = 1...\mathfrak{P}; \\ j = ((\kappa-1) \times (\mathtt{stripeWidth}_{\kappa}+1)) \text{ to } (\kappa \times \mathtt{stripeWidth}_{\kappa}); \quad i = \text{Row Number} \tag{2}$$

Algorithm 1: PSEUDO CODE TO OBTAIN THE STRIPES

```
startCol = 0
for i ← 1 to (𝔓 − 1) do
    stripe.start = startCol
    stripe.end = (startCol + 𝔚)-1
    startCol = startCol + 𝔚
end
lastStripe.start = startCol
lastStripe.end = I.Cols
```

In the Equation 2, $GLM_{i,\kappa}$ is the average gray value of all the pixels placed in the i^{th} row and κ^{th} stripe. $\texttt{ImgStripe}_{i,j}$ is the gray value of the i^{th} row and j^{th} column of the `ImgStripe`. The width of κ^{th} stripe here is $\texttt{stripeWidth}_{\kappa}$. The obtained image stripe by GLM operation is shown in Fig. 2c. After obtaining the GLM image, the Otsu's binarization technique is applied on each stripe with a threshold of 50 (chosen heuristically) to obtain a painted binary image as shown in Fig.2d. The white and black rectangles/bands represent the foreground and background regions respectively, where the foreground pixels principally represents the text-line regions in the considered image stripe.

Algorithm 2: GENERATION OF TRAINING AND TESTING SET

Input: $\mathfrak{T}$ (Number of training samples needed), $\mathfrak{E}$ (Minimum number of logos present in any logo classes)
Output: $\mathfrak{U}$ (Logo image indexes from each class)

```
Get logo class (𝔭) and instance number(𝔯) from image name i.e. "𝔭_𝔯.png"
Use hash map ℌ to arrange the logo image names by using 𝔭 as key and 𝔯 as values
iCnt = 0; itemCnt = 0
Initialize a vector 𝔘 of the size ℌ, which will contain another vector of integers
while itemCnt< 𝔗 do
    if iCnt == 0 then
        eCnt = 0
        for i ← 1 to (ℌ.length()) do
            className = ℌ(i).first
            allInstances = ℌ(i).second
            if allInstances.length() > 𝔈 then
                Generate a random number 𝔯 between 1 to allInstances.length()
                Vector < Integer > ℜ
                ℜ.put(𝔯)
                𝔘.put(ℜ)
            end
            eCnt + +
        end
        iCnt + +
    end
    else
        for i ← 1 to (ℌ.length()) do
            className = ℌ(i).first
            allInstances = ℌ(i).second
            Vector < Integer > 𝔜 = 𝔘.get(eCnt)
            if ((𝔜.size() <= (allInstances.size() − 𝔈)) && (allInstances.size() > 𝔈)) then
                ▹ continue while loop until an unique random number is not obtained.
                while True do
                    Generate a random number 𝔯 between 1 to allInstances.length()
                    boolean rightRandFlag = false
                    Check whether this random number is already present in 𝔜 or not
                end
                𝔜.put(𝔯)
                𝔘.put(ℜ)
                if iCnt== 𝔗 then
                    Exit from all the loop
                end
            end
        end
    end
end
```

As shown in Fig. 2e, the tiny black space between two white blocks, e.g., the one which is shown by

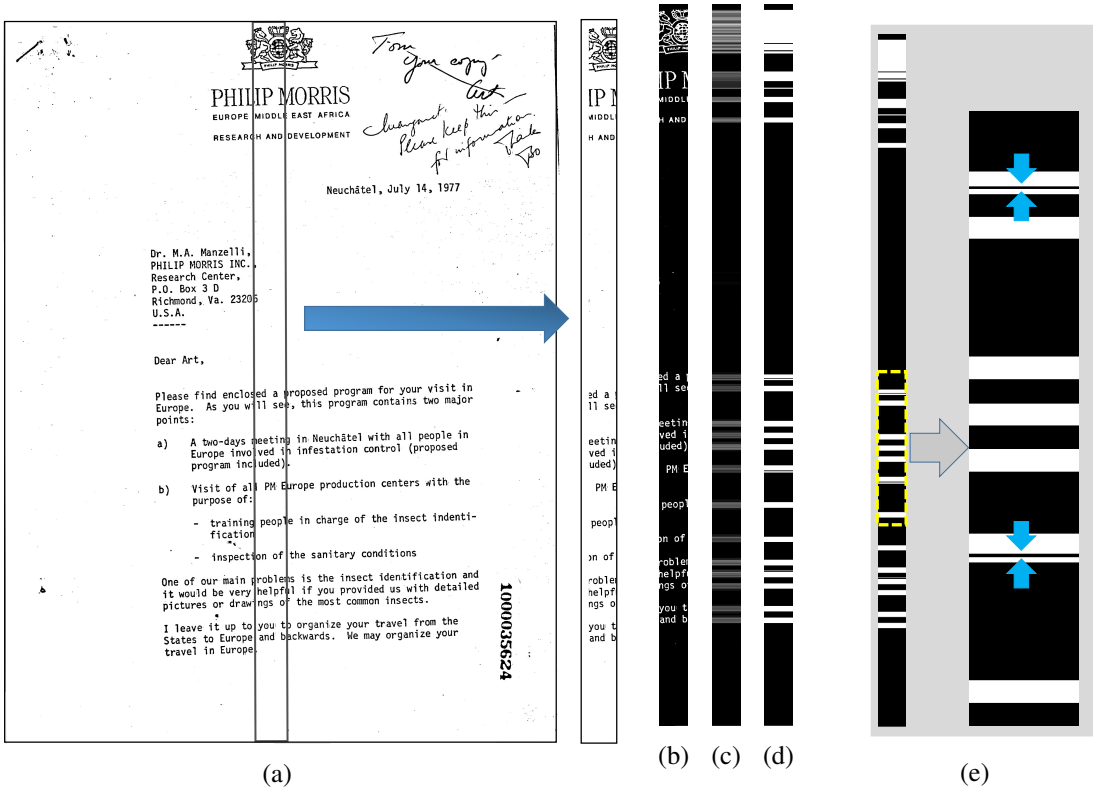

Figure 2: The result of each steps, obtained from the application of Piecewise Painting Algorithm on a document image. (a): A vertical stripes obtained from the full image. (b): The binarized, complemented and noise removed image. (c): Row averaged stripe image. (d): Otsu's binarization result on row averaged image. (e): White run gap shown by both sided sky colored arrow which is different from the gap between the text lines.

two sky-colored arrow in Fig.2e, should be converted into white (the magnified view of the ROI marked by dotted yellow color of left image is shown at the right of Fig.2e) as this region should be the foreground region. But the problem is how to distinguish between this kind of black spaces and a general interline spaces (also a black region). To correctly obtain intra text line gap between intra text line (shown by two sky-colored arrow in Fig.2e), following Algorithm 3 is devised. Please note that in Algorithm 3, it is considered that foreground pixel is black (0) and background pixel is white (255), which is contrary to the image shown in Fig. 2e. The idea is to obtain white background pixels which starts with a black (foreground) pixel and ends with another black pixel. Based on this logic, a series of background pixels located at the beginning rows (due to the presence of border, any initial background pixel should not start with these series of background pixels should not start with foreground pixel/s) and at the ending rows (due to the presence of page border, a series of background pixels should not end with a foreground pixel/s) could be ignored. Line 7 in Algorithm 3 checks whether the current pixel is a background pixel or not then

(a) (b) (c) (d) (e)

Figure 3: The result obtained after filling-up the gaps within a text line. (a): The vertical stripes obtained from the full image. (b): The image after binarization of the row averaged image. (c): The binarized stripe image obtained after filling-up the gaps exits within text lines. (d): The complete binarized image after row averaging operation. (e): The complete binarized image obtained after filling-up the gaps exits within text lines (the filled up gaps are painted in orange color just for visualization purpose but in the case of our application these gaps are painted in black/ foreground pixels).

it checks in Line 8 whether the previous pixel was a foreground pixel or not. If this condition satisfies then we obtain the very first required background pixel. After obtaining the first background pixel of `oneFlag` becomes true and then all the other adjoining pixel in this same background band is obtained by entering in the ”ELSE-IF” loop mentioned in Line 12-13.

Otherwise, if the current pixel is the very first (checked in Line 15 by verifying the status of `zeroFlag`) foreground pixel (Line 14), the process of counting and storing the location of first foreground pixel is given in Line 16-18. In Line-19, the 1^{st} ”IF” condition checks whether the previous pixel was a background pixel or not, whereas the 2^{nd} condition checks whether the control has not reached at the last row of the image

(moreover pixel at the last row should not be a foreground pixel and the background band[1] before it

[1]Fore-ground and back-ground band means a series of fore-ground and back-ground pixels

Algorithm 3: OBTAINING AND FILLING-UP THE GAP WITHIN A TEXT LINE

Input: $\mathbb{I}$ (The stripe image after row averaging and binarization)
Output: All the starting and ending positions, lengths of foreground and background bands.

▷ It is considered that foreground is black (0) and background is white (255)

```
oneCnt = 0; oneFlag = false
zeroCnt = 0; zeroFlag = false
j =(𝕀.Rows)/2                    ▷ only considering the middle column
startBlackRun = −1; endBlackRun = −1
startWhiteRun = −1; endWhiteRun = −1
for i ← 2 to 𝕀.Rows do
    ▷ Only take those white runs (back-ground) which are beginning and ending with black pixel (fore-ground)
    if (𝕀_{i,j} == 255) then
        if (𝕀_{i−1,j} == 0) then
            ▷ If the current pixel is white then check whether the pixel before was black
            oneCnt = 1
            startWhiteRun = i
            oneFlag = true
        end
        else if (oneFlag) & (i ≠ (𝕀.Rows) ) then
            oneCnt = ++
        end
        else if (𝕀_{i,j} == 0) then
            if (zeroFlag == false) then
                ▷ Very first pixel of the black or foreground stripe
                zeroCnt = 1
                zeroFlag = true
                startBlackRun = i
                ▷ Following oneFlag variable in "IF" condition confirms that we have already traversed a white band which starts and ends with black pixel.
                if ((𝕀_{i−1,j} == 255) & (i ≠ (𝕀.Rows)) & (oneFlag == true)) then
                    endWhiteRun = i
                    startWhiteRun = −1          ▷ reset the value
                    endWhiteRun = −1            ▷ reset the value
                    oneCnt = 0; oneFlag = false
                end
            end
            else if (zeroFlag == true) then
                zeroCnt ++
            end
            ▷ We have encountered the last black pixel of the black band and the next pixel is white or the next pixel is the end pixel of image
            if ((𝕀_{i+1,j} == 255) OR (i == (𝕀.Rows − 1))) then
                zeroFlag = false
                endBlackRun = i
                blackCnt ++
                zeroCnt = 0
            end
        end
    end
end
```

should not exist. But if it exits then it is irrelevant and should be ignored). The third "IF" condition checks

whether the control has already passed through the background pixel/s before arriving on this line of code. If all of these three conditions satisfies then it confirms that the ending of a required black band, hence the ending position is stored and other needed variables are reset (Line 19-23). Line 24-25 counts the continuous adjoining foreground pixels if we already have encountered (Line 24; `zeroFlag == true`) the first white (foreground) pixel. At last, the "IF" loop mentioned in Line-26 checks whether we are currently encountering the last foreground pixel or we have reached the last row of the image then store the last foreground pixel location and increase the counter for foreground pixel.

After obtaining all the starting and ending positions, lengths of foreground and background bands from all the stripes, the mean height of the foreground bands is calculated to understand the average height of text lines in the image. The technique to calculate mean value from an array of elements is given in Algorithm 4. The Algorithm 4 starts by sorting the array of elements in ascending order. We have proposed a binning based approach to divide the elements into 10 bins (this number can be changed but it is experimented that dividing into 10 bins give good results in most of the cases). Line 7-10 computes the lower and upper limit of values whereas Line 6 defines the width of each bin. In Line 13, the value is checked against the computed bin width, if the value is lesser than bin width then the value is stored in first bin otherwise the remainder after division (`getDivRemainder`) is calculated. If `getDivRemainder` is greater than zero then the probable bin in which the value would belong can be calculated by $\frac{\texttt{getVal}}{\texttt{binWidth+1}}$ otherwise the probable bin would be $\frac{\texttt{getVal}}{\texttt{binWidth}}$. After computing the probable bin index for each values, the values are rechecked to verify that does it really belongs to that particular bin or not (Line 16). Based on the number of elements in each bin, the top 2 bins are considered and then mean, standard deviation of the values belonging in these bins are calculated. Now all the values in the array is checked against this condition of (`(meanVal-stdVal)` and `(meanVal+stdVal)`). Only those values are considered which satisfies this condition (Line 29) and then at last the mean is computed from these values only.

After computing mean height of foreground bands (`avgForegroundHeight`), we look into each stripe and first checks only those background bands which are initiated and ended with foreground bands (background band is in between two foreground bands). It is first checked that the background band height should be less than `avgForegroundHeight`. Then the following constraint is verified for considering

Algorithm 4: CALCULATE MEAN VALUE OF AN ARRAY OF ELEMENTS

Input: An array of values
Output: The mean of the values in the array

```
Sort the array elements in ascending order
Get last element of the sorted array as the maximum value (maxVal)
Get first element of sorted array as the minimum value (minVal)
noOfBins = 10
if (maxVal - minVal) > 50 then
    binWidth = (maxVal - minVal) / noOfBins
    for i ← 1 to noOfBins do
        binLowerBound = minVal + (binWidth×i)
        binUpperBound = minVal + (binWidth×(i + 1))
        For each bin, store it's lower bound (binLowerBound) and upper bound (binUpperBound)
    end
    for k ← 1 to noOfArrElements do
        getVal = sortedArr.at(k)
        if getVal > binWidth then
            getDivRemainder = getVal % binWidth
            binApprox = (getDivRemainder > 0) ?  (getVal/binWidth+1) :  (getVal/binWidth)
            if (binWidth×(binApprox-1))≤getVal≤(binWidth×binApprox) then
                if binApprox>noOfBins then
                    binApprox = noOfBins
                end
                if binApprox<1 then
                    binApprox = 1
                end
                Store array indexes and values in corresponding bins
            end

        end
        else
            binApprox = 1
            Store array indexes and values in corresponding bins
        end
    end
    Based on the number of elements in each bin, consider top 2 bins
    Calculate mean (meanVal) and standard deviation (stdVal) of these elements obtained from top 2 bins
    for k ← 1 to noOfArrElements do
        getVal = sortedArr.at(k)
        if ((meanVal-stdVal)≤getVal≤(meanVal+stdVal))&(getVal > 8) then
            Store these elements in an new array called refinedArray
            If size of refinedArray is less than 3 then start over again by removing this condition (getVal > 8)
        end
    end
    Calculate mean value of the elements stored in refinedArray
end
else
    Calculate mean value of the elements stored in array
end
```

any background band pixels to be converted into foreground pixels. The total height of top foreground band (`topForegroundBand`), bottom foreground band (`bottomForegroundBand`) and the in-between background band (`backGroundBand`) should be less than: (`avgForegroundHeight` + `avgForegr--oundHeight` $\times \frac{30}{100}$). So, by applying aforementioned approach, the gap exists within a text line could be distinguished and could be converted into foreground pixels.

4.4. Feature Extraction

By employing the above mentioned segmentation technique for localizing the probable position of logos in the image several patches are provided. However, among all the extracted patches, only a small number or none of them contain a logo or parts of a logo. To prune the unnecessary patches and to obtain the most probable logo patch(es), in this research work two novel probability based feature extraction techniques by considering prior knowledge of logo positions are proposed. Moreover, a few geometric shape based features are also included in the feature set for this purpose. Details of these proposed feature extraction techniques are presented in the following sections.

4.4.1. Frequency Probability Based Feature

With the help of prior knowledge of a logo position, gathered from the training set, a document can be represented in different ways, such as probability map for blurred shape model [29] and geometrical position [23] [37] [38]. The frequency probability map is computed based on frequent appearances of logos in different positions of document images during training step as follows.

Suppose a document image is denoted by $\mathbb{I}^K_{P\times Q}$, where P and Q are the height and width of the image $\mathbb{I}^K$, where K varies from 1 to $\mathfrak{T}$. $\mathfrak{T}$ denotes the no. of document images used as training samples. Let's define a matrix $\mathfrak{FM}_{P\times Q}$ as the frequency probability matrix of size $P \times Q$ which is initialized by setting 0 at all the cells. The height (P) and width (Q) of the matrix is calculated by taking maximum of the heights and widths of all the training images. Throughout the training phase, the matrix $\mathfrak{FM}$ is modified by employing $\mathfrak{FM}_{i,j} = \mathfrak{FM}_{i,j} + 1$ if $\mathbb{I}^K_{i,j}$, which is a pixel of logo component, positioned at the coordinate (i, j) in the image I^K. Otherwise, if the pixel is not a part of a logo component, no change will be performed in $\mathfrak{FM}_{i,j}$. Please note that the original $(i, j)^{th}$ location of the pixel in image I^K is mapped into the Frequency Probability matrix ($\mathfrak{FM}_{P\times Q}$). To normalize the values of $\mathfrak{FM}$ matrix between 0 and 1, each cell value is divided by the

maximum value of $\mathfrak{FM}$ by using the formula $\mathfrak{FM}_{i,j} = \frac{\mathfrak{FM}_{i,j}}{Max\{\mathfrak{FM}\}}; i = 1, 2, ...P; j = 1, 2, ...Q$. As a result, all the elements of FM have a value between 0 and 1 and the matrix cells/elements close to value 1 signify a higher probability of the presence of the logo and the elements with values close to 0 indicate lesser probability of representing logos at those positions. To obtain the Frequency Probability (FP) feature of a patch, the patch is mapped on the Frequency Probability Matrix ($\mathfrak{FM}$) by considering the coordinates of the patch. Then, the average probability value is computed from the probability values corresponding to all the foreground pixels present in that patch in ($\mathfrak{FM}$) matrix.

4.4.2. Gaussian Probability Feature

In the work mentioned in [36], the positions of logos in video frame have been characterized by using four different mixed Gaussian distribution. Similar to the video frames, logos in document images have also a clear positional preferences in contrast to text content and other information. The logos have quite a high probability of occurring at the top and at the bottom than at the center of the documents. To calculate this attribute, except using the FP feature, the distance of a patch from predefined positions of different logos is also considered as a feature in the proposed feature set. To compute this feature, we use the same Frequency Probability matrix ($\mathfrak{FM}$). Based on the analogy performed to find the higher probability of logo locality, it is observed that the logos belongs to the top and bottom than at the center of the documents. Due to this reason, the Frequency Probability matrix $\mathfrak{FM}$ is divided into 3 vertical blocks. Now each vertical block is divided into 3 horizontal blocks result in 9 blocks in total. Only top 3 and bottom 3 blocks are considered to approximate the probability by a mixed Gaussian distribution. Since, six blocks are considered for modeling the logo positions, six bivariate Gaussian distributions are used to represent this probability. The coordinates of non-zero pixels and corresponding center of gravity (C.G.) of non-zero pixels are obtained in each block. From the coordinates of each of such six C.G, the Gaussian probability (GP) feature for an extracted patch is calculated by using the following formula :

Mean of all the non-zero pixels's X coordinates, present in each block: $\mu_x^i; i = 1, 2, ..., 6$

Mean of all non-zero pixels's Y coordinates, present in each block: $\mu_y^i; i = 1, 2, ..., 6$

Standard-Deviation of all the non-zero pixel's X coordinates, present in each block: $\zeta_x^i; i = 1, 2, ..., 6$

Standard-Deviation of all the non-zero pixel's Y coordinates, present in each block: $\zeta_y^i; i = 1, 2, ..., 6$

Let's assume there are p number of non-zero pixels present in any particular block. Another parameter (ρ)

is calculated for each block by using the following Equation3. So, for each of the 6 blocks ($i = 1, 2, .., 6$), these parameters $(\mu_x, \mu_y, \zeta_x, \zeta_y, \rho)$ are calculated by using only the training set.

$$\rho^i = \frac{\sum_{t=1}^{p} (X_t^i - \mu_x^i) \times (Y_t^i - \mu_y^i)}{p} \tag{3}$$

$$\mathbb{P}(\frac{L_{\mathfrak{p}}}{P_{\mathfrak{p}}} = 1) = \frac{1}{2\pi\zeta_x\zeta_y\sqrt{(1-\rho^2)}} \times e^{-\frac{1}{2\times(1-\rho^2)} \times \frac{(x-\mu_x)^2}{\zeta_x^2} + \frac{(y-\mu_y)^2}{\zeta_y^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\zeta_x\zeta_y}}; \mathfrak{p} = 1, 2, ..., \mathfrak{n} \tag{4}$$

While testing, after obtaining the patches from the test image, the centroid of the probable logo patches are considered for the computation of Multi-modal Gaussian Feature. In the above mentioned Equation4, the centroid of patch P is denoted by $L_p = (x, y)$. Based on the location of the patch's centroid in any particular block out of considered six (blocks top left, top center, top right, bottom right, bottom center and bottom left respectively), corresponding parameters of that specific block is considered to calculate this feature by using the above mentioned Equation 4. $\mathbb{P}(\frac{L_{\mathfrak{p}}}{P_{\mathfrak{p}}} = 1)$ is the Gaussian probability (GP) of the $\mathfrak{p}^{th}$ patch by assuming that there are $\mathfrak{n}$ number of patches exists in this test image. Where, $L_{\mathfrak{p}} = (x, y)$ denotes location of the centroid of the $\mathfrak{p}^{th}$ patch.

4.4.3. Convex Hull Based features

In order to better describe a logo prototype using connected components, a convex hull is formed for every connected component. These features of convex hull have been successfully used for logo detection and recognition in [19]. For a convex hull, its orientation θ using the 2^{nd} order moments, the square root of the variance of the main axis with respect to the orientation σ_{max}, the square root of the variance on its orthogonal direction σ_{min}, the maximum length l_{max}, minimum edge l_{min} and the square root of its size l_{size} are also calculated. By using these values, a connected component descriptor **ConComp** is constructed as:

$$\textbf{ConComp} = \{\urcorner_r, \urcorner_\sigma, \urcorner_S, \urcorner_l, \urcorner_b, l_{size}, IDs, \vec{ac}\}; \urcorner_r = \frac{l_{max}}{l_{min}}; \urcorner_\sigma = \frac{\sigma_{max}}{\sigma_{min}}; \urcorner_S = \frac{l_{size}}{\sigma_{max}}; \urcorner_l = \frac{l_{size}}{l_{max}} \tag{5}$$

In the above mentioned Equation 5, $\urcorner_b$ means blackness which is calculated by dividing the number of pixels of the connected component and the area of the convex hull. The descriptor elements $\urcorner_r, \urcorner_\sigma, \urcorner_S, \urcorner_l, \urcorner_b$ are

invariant to image scaling and rotation. The feature level matching using them is only a calculation of Euclidean distance. Let's consider that there are $\mathfrak{p}$ classes of logos present in database. Any one particular kind of logo may appear one or more than one time in the same document. The IDs is used to save logo class information corresponding to any particular patch. The parameter $\vec{ac}$ represents the anchor line of the connected component.

4.4.4. Identification of logo-patches by using decision tree

Logo detection in document images is a particular field where only a few instances per class are available for training. Moreover, the text content (non-logo portion) is much more than the logo contents in document images which results in a class imbalanced data problem. See the result of employing PPA on a document images where among all the extracted patch only one patch is the logo patch. Decision tree (DT) perform well in such a scenario and it has been reported being more suitable to deal with such problems [26]. Furthermore, DTs are designed without assumption about the features space distribution and they can perform better using geometric and domain-based features; as these features are statistically independent, non-homogeneous and non uniformly distributed in most of the cases [26].

In this research work, a decision tree (DT) is used as classifier for the coarse classification of patches into logo and non-logo patches following feature extraction. To design a DT, several automatic algorithms such as CART, C4.5 and CS-C4.5 have been proposed in the literature. Here the CS-C4.5 is used to design the DT. The CS-C4.5 is a cost sensitive C4.5 DT, which considers a cost matrix during the pruning step to take care about the misclassification of logos. It also uses gain ratio as splitting criterion to deal better with the class imbalanced dataset [26].

Employing the proposed DT, a small number of patches (logo-patches) with higher probability of containing logos are obtained. Due to the degradation, low quality image and multi-part logos, we sometimes may not get appropriate patches for logos during the painting operation. Furthermore, very small logo-patches may be eliminated using the proposed DT. To accomplish this problem, the results obtained by employing the proposed DT is subjected to a dilation operation using a structuring element of length $\mathfrak{MG}$ and width 1, where $\mathfrak{MG}$ is average gap between extracted patches. The length of structuring element is decided based on experimentation. These patches are called Region of Interest (RIs). For each RI, we fix a minimum bounding box, which is then mapped on the original document image. These bounding boxes

shown by red may be extended or tightened to have a finer minimum bounding box (green). The foreground information bounded by minimum bounding boxes (shown in green) is considered for further process at the fine level of our scheme. The logo detection results obtained at the coarse level of our proposed scheme using different dataset are shown in Table II III and IV.

4.4.5. Extraction of Shape Context Features and Classification

Shape context descriptor as a geometric invariant descriptor which has been used for the recognition of logos, objects and patterns [27] [7]. The shape context at a reference point captures the distribution of the remaining points relative to that reference point and provides a globally discriminative characterization. There is another approach called "Shapeme Histogram Descriptor", which is inspired by shape context descriptor and bag of visual model [27] [21]. But before going into the details of this "Shapeme Histogram Descriptor", we would like to throw some light on the brief description of "Shape Context Features".

In the initial step, a set of points are select ed from the logos by detecting edges with the application of canny edge detector. After the detection of edges, the edge points are sampled in order to obtain a fixed number of n points p_i per logo l. Based on these n points, the distribution of points within the plane relative to each point of the shape can be measured. Then a histogram using log-polar coordinates counts the number of points inside each bin. For a point p_i of the shape, a histogram h_i of the coordinates of the nearby points q is computed as:

$$h_i(k) = \#\{q \neq p_i : q \in bin_{p_i}(k)\} \tag{6}$$

The same experimental setup as the one mentioned in [27] has been followed in our case. We have chosen 5 bins for $logr$ and 12 bins for θ. Shape context descriptor is translation and scale invariant. Translational invariance comes due to the computation of histograms from reference points whereas scale invariance comes by normalizing all the radial distances by the mean distance between all the point pair int he shape. The rotation invariance can be given by measuring angles at each point relative to the direction of the tangent at that point. After all the n points in a shape are described by their shape context histogram, two shapes are matched by finding the point of correspondences. One of the simplest way to compute the matching between two set of points is by using a bipartite graph matching approach which calculate correspondence

between points having similar shape context descriptions. More robust level of matching can be obtained by the computation of affine transform which matches the set of points from one shape to another shape. But as shape context descriptor provides local description of key-points which requires a point correspondence matching inhibits it's applicability to the retrieval problem in large collections. To overcome this issue of shape context descriptor, one variant is proposed which aims to describe the logos globally and is known as "Shapeme Histogram Descriptor". Mori et al. [21] presented the Shapeme Histogram Descriptor which is inspired by aforementioned shape context descriptor and bag-of-words model. The main idea of this descriptor is to compute shape context descriptor for all the interested points extracted from a symbol and then use vector quantization in the space of shape contexts. For fast matching, the vector quantization involves a clustering stage of the shape context feature space. Once the clustering is computed, each shape context descriptor can be identified by the index of the cluster in which it belongs to. These clusters are called "Shapemes". Each logo is then described by a single histogram which represents the frequency of appearance of each shapeme.

During the learning stage, the shape context descriptors can be computed and then clustering of the space by means of K-means algorithm can be performed by using training set. The total of K cluster centers can be obtained and these centers can be assigned to a given integer index $I \in [1, k]$. Then, during the recognition stage, given a logo l, and it's n sampled points obtained from it's edge map, the shape context descriptors ($h_i, \forall i \in [0, n]$) can be computed. Each shape context descriptor of the points p_i is then projected to the clustered space and can be identified by a single index I_i. Hence, the logo l can be represented by a histogram coding of the frequency of appearance of each of the k shapeme indices. In this manner, we can globally describe each logo by using a unique histogram $\mathfrak{SH}$ by applying the following equation:

$$\mathfrak{SH} = \#I_i == x : I_i \in [0, k] \tag{7}$$

In our application, at first the RIs are ranked based on the average of Frequency Probability features, Multi-modal Gaussian Probability features and Convex Hull based features. So the RI with the highest rank in the ranked list of RIs has the priority for further process. Since, each document in the Tobacco-800 contains between 0 and 5 logos, five RIs are needed to be further processed in the finer-level. The shape context feature are thus extracted for each RI to identify it as a logo or a non-logo at the fine level of the

proposed logo detection scheme using a Nearest Neighbor Classification with a correlation distance matrix.

4.5. Results and Discussion

Acknowledgment

[1] "The Legacy Tobacco Document Library ({LTDL})," 2007.

[2] G. Agam, S. Argamon, O. Frieder, D. Grossman, and D. Lewis, "The Complex Document Image Processing ({CDIP}) Test Collection Project," 2006.

[3] Z. Ahmed and H. Fella, "Logos extraction on picture documents using shape and color density," *IEEE International Symposium on Industrial Electronics*, pp. 2492–2496, 2008.

[4] A. Alaei, M. Delalandre, and N. Girard, "Logo detection using painting based representation and probability features," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, no. August, pp. 1235–1239, aug 2013.

[5] A. Alaei, U. Pal, and P. Nagabhushan, "A new scheme for unconstrained handwritten text-line segmentation," *Pattern Recognition*, vol. 44, no. 4, pp. 917–928, 2011.

[6] A. Alaei, P. P. Roy, and U. Pal, "Logo and seal based administrative document image retrieval: A survey," *Computer Science Review*, vol. 22, pp. 47–63, 2016.

[7] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[8] J. Chen, M. K. Leung, and Y. Gao, "Noisy logo recognition using line segment Hausdorff distance," *Pattern Recognition*, vol. 36, no. 4, pp. 943–955, 2003.

[9] R. O. Duda, P. E. P. E. Hart, and D. G. Stork, *Pattern classification*. Wiley, 2001.

[10] Y. Freund and R. R. E. Schapire, "Experiments with a New Boosting Algorithm," *International Conference on Machine Learning*, pp. 148–156, 1996.

[11] S. M. Halawani and I. A. Albidewi, "Logo Matching Technique Based on Principle Component Analysis," vol. 1, no. 03, pp. 1–5, 2010.

[12] S. Hassanzadeh and H. Pourghassem, "A Novel Logo Detection and Recognition Framework for Separated Part Logos in Document Images," *Australian Journal of Basic and Applied Sciences*, vol. 5, no. 9, pp. 936–946, 2011.

[13] W. Hongye and C. Youbin, "Logo detection in document images based on boundary extension of feature rectangles," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, no. c, pp. 1335–1339, 2009.

[14] R. Jain and D. Doermann, "Logo retrieval in document images," *Proceedings - 10th IAPR International Workshop on Document Analysis Systems, DAS 2012*, pp. 135–139, 2012.

[15] A. Joly and O. Buisson, "Logo retrieval with a contrario visual query expansion," *Proceedings of the seventeen ACM international conference on Multimedia - MM '09*, p. 581, 2009.

[16] V. P. Le, N. Nayef, M. Visani, J.-M. Ogier, and C. D. Tran, "Document Retrieval Based on Logo Spotting Using Key-Point Matching," in *2014 22nd International Conference on Pattern Recognition*. IEEE, aug 2014, pp. 3056–3061.

[17] V. P. Le, M. Visani, C. D. Tran, and J.-M. M. Ogier, "Improving logo spotting and matching for document categorization by a post-filter based on homography," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, no. 1. Ieee, aug 2013, pp. 270–274.

[18] D. Lewis, G. Agam, S. Argamon, O. Frieder, D. Grossman, and J.Heard, "Building a Test Collection for Complex Document Information Processing," in *Proc. 29th Annual Int. ACM SIGIR Conference*, 2006, pp. 665–666.

[19] Z. Li, M. Schulte-Austum, and M. Neschen, "Fast logo detection and recognition in document images," *Proceedings - International Conference on Pattern Recognition*, pp. 2716–2719, 2010.

[20] S. Lowther, V. Chandran, and S. Sridharan, "Recognition of Logo Images Using Invariants Defined from Higher-Order Spectra," no. January, pp. 1–4, 2001.

[21] G. Mori, S. Belongie, and J. Malik, "Efficient shape matching using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1832–1837, 2005.

[22] O. Of, "Robust Logo Recognition for Mobile Phone Applications," vol. 559, pp. 545–559, 2011.

[23] T. A. Pham, M. Delalandre, and S. Barrat, "A contour-based method for logo detection," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 718–722, 2011.

[24] T. D. Pham, "Unconstrained logo detection in document images," *Pattern Recognition*, vol. 36, no. 12, pp. 3023–3025, 2003.

[25] H. Qi, K. Li, Y. Shen, and W. Qu, "An effective solution for trademark image retrieval by combining shape description and feature matching," *Pattern Recognition*, vol. 43, no. 6, pp. 2017–2027, 2010.

[26] L. Rokach and O. Maimon, "Top-Down Induction of Decision Trees ClassifiersA Survey," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 4, pp. 476–487, nov 2005.

[27] M. Rusiñol and J. Lladós, "Logo spotting by a bag-of-words approach for document categorization," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 111–115, 2009.

[28] ——, "Efficient logo retrieval through hashing shape context descriptors," *Proceedings of the 8th IAPR International Workshop on Document Analysis Systems - DAS '10*, pp. 215–222, 2010.

[29] M. Rusiñol, V. Poulain D'Andecy, D. Karatzas, and J. Lladós, "Classification of administrative document images by logo identification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7423 LNCS, pp. 49–58, 2013.

[30] K.-t. Sam and X.-l. Tian, "Vehicle Logo Recognition Using Modest AdaBoost and Radial Tchebichef Moments," vol. 25, pp. 91–95, 2012.

[31] S. Seiden, "Logo Detection in Document Images 1 Introduction 2 Feature Extraction," *Interpretation A Journal Of Bible And Theology*, pp. 1–6, 1997.

[32] K. Tieu and P. Viola, "Boosting Image Database Retrieval," *Aim-1669*, vol. 56, no. 1669, pp. 17–36, 1999.

[33] K. Tombre and B. Lamiroy, "Pattern Recognition Methods for Querying and Browsing Technical Documentation." Springer, Berlin, Heidelberg, 2008, pp. 504–518.

[34] F. M. Wahl, K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer Graphics and Image Processing*, vol. 20, no. 4, pp. 375–390, dec 1982.

[35] H. Wang, "Document Logo Detection and Recognition Using Bayesian Model," *2010 20th International Conference on Pattern Recognition*, no. c, pp. 1961–1964, 2010.

[36] W. Q. Yan, J. Wang, and M. S. Kankanhalli, "Automatic video logo detection and removal," *Multimedia Systems*, vol. 10, no. 5, pp. 379–391, 2005.

[37] G. Zhu and D. Doermann, "Automatic document logo detection," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 2, pp. 864–868, 2007.

[38] ——, "Logo matching for document image retrieval," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2009, pp. 606–610.