ELSEVIER

Signal Processing: *Image Communication* 00 (2000) 000–000

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

MPEG-7 *Videotext* description scheme for superimposed text in images and video

Nevenka Dimitrova[a,*], Lalitha Agnihotri [a], Chitra Dorai[b], Ruud Bolle[b]

[a]*Philips Research, Briarcliff Manor, NY, 10510, USA*
[b]*IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA*

Abstract

Superimposed text and scene text in video, i.e. videotext, brings important semantic clues into content analysis. In this paper we present a videotext description scheme and automatic methods for detection and representation of text in video segments. One of the methods is based on edge characterization while the other is based on region analysis. Applications of the videotext description scheme are numerous ranging from video indexing and annotation, ticker-tape analysis, commercial detection, transcript analysis to cross-modal querying using text and face information.

Keywords: MPEG-7; Content description; Image text detection; Videotext detection; Videotext description scheme

1. Introduction

With the increasing popularity of the World Wide Web (WWW) and the introduction of streamed digital audio and video on the Internet, the amount of multimedia information available to consumers continues to grow. As content becomes readily available, automatic indexing during archiving and searching large volumes of multimedia data will become difficult [1–5,14,17,18,25]. In this context the objective of MPEG-7 standard [23] is to provide descriptions for multimedia data and enable interoperability along the content value chain.

An important and integral part of video that contains high-level semantic information is *superimposed* text and *scene* text. We call this text *videotext* to distinguish it from text in documents or ASCII text. The important aspect of superimposed text is that it represents meta-information, which is intentionally inserted by the video producers according to well-accepted conventions. Good examples of superimposed text are anchor's name, location and event in the TV news. Examples of scene text are product names and street signs. Videotext is a separate channel of information that is not always conveyed as closed-captioned, auditory or transcript data. This text can be manually recorded during video production or automatically detected from archived video material. There are methods in the literature that describe automatic extraction of text from images and video. In this

* Corresponding author. Tel.: (914) 945-6059; fax: (914) 945-6141.

E-mail address: nevenka.dimitrova@philips.com (N. Dimitrova).

0923-5965/00/$ - see front matter
PII: S0923-5965(00)00022-9

paper we will present two such methods proposed to MPEG-7 [7,8] and a description scheme associated with the extracted text.

Videotext related to the superimposed text on the frames can be used for video annotation, indexing, semantic video analysis and search [5,9]. For example, the origin of a broadcast is indicated by a graphic station logo in the right-hand top or bottom of the screen. Such station logos can be automatically recognized and used as annotation. Anchor/correspondent names and locations in a news program are often displayed on the screen and can be recognized by extracting the text showing in the bottom one-third of the video frame. Musician names and music group names, talk show hosts and guests, and other TV personalities are also introduced and identified in a similar fashion. So, by detecting the text box and recognizing the text, the video can be indexed based on a TV personality or a location. This information can then be used for retrieving news clips based on proper names or locations. Sports programs can be indexed by extracting the scores and team or player names.

Videotext is also useful in performing text analysis and categorizing text into different classes such as video topics, person appearances, sports scores, etc. The spatial designator of the text region containing the character block can provide clues about the category of the text such as image caption, channel number, sports score, etc. Hence, the text semantics and the text location within the frame are valuable cues for the important problem of automatic video categorization. The location and size of text in commercials can be used in conjunction with other features for reliable commercial detection [20]. By keeping track of text movements, we can find scrolling, static or flying videotext. For example, the presence of scrolling text often signals the beginning or ending of programs. This can help in finding program boundaries. The subtitles in a video can be analyzed to extract the transcript, index and query video streams.

There are numerous examples where the extraction of a text description scheme can be used in conjunction with other tools in MPEG-7. For example, one can use text detection along with shot detection algorithms to extract important frames (i.e., keyframes) and generate a visual table of contents that is more meaningful for the consumer. Obviously, a keyframe extracted from a football game displaying the current score is better than the one without the score.

An interesting application of videotext detection is recognition of the "ticker" that runs during games, talk shows, and news. This ticker could convey weather updates or stock market figures. In this manner, additional information can be extracted and retrieved for future use. This information can be viewed as something coming from a completely different channel than the broadcast program itself, as it is unrelated to the program in view.

Text recognition in document images has been an active research area for some time. However, text detection and recognition in video frames is a different domain and requires a very different approach. This is because text in printed documents is restricted to uni-colored characters on a uniform background. It only requires a simple thresholding to separate the text from the background. In video, however, characters in scene images suffer from a variety of noise components. Further, the background is a moving one and the characters can be of varied color, sizes and fonts. Ohya et al. [25] perform character extraction by local thresholding and detect character candidate regions by evaluating the gray-level differences between adjacent regions. They follow this by merging detected regions that exist close to each other and have similar gray levels to generate character pattern candidates. Hauptmann and Smith [14] use the spatial context of text and high contrast of text regions in scene images to merge large numbers of horizontal and vertical edges in spatial proximity to detect text. Lienhart and Stuber [17] use a non-linear R'G'B' color system to reduce the number of colors. A subsequent split-and-merge produces homogeneous segments having similar color. Further, they use the heuristic of characters being in the foreground, being monochrome, rigid, having size restrictions, and having a high contrast with surroundings to detect characters in the above homogeneous regions. Jain and Yu use multi-valued image decomposition for separating images into multiple real foreground and background images [15]. More algorithms that could be useful for

character segmentation are described elsewhere [4,19,26].

The rest of this paper is organized as follows. In Section 2 we will introduce the videotext description scheme, and present how it relates to the generic audio visual DS and to the MPEG-7 requirements. Section 3 describes in detail the computational framework and our algorithms used to extract text from video frames. Section 4 contains the results from applying our techniques to a subset of videos from the MPEG-7 data set. Section 5 briefly describes video character recognition and Section 6 concludes the paper.

2. Videotext representation

Elements of the videotext feature are presented in great detail in the appendix a. In this section, we discuss the continually evolving relationship between our videotext feature representation and MPEG-7 description schemes.

The two proposals to MPEG-7 for videotext descriptors were merged [7,8] into a single videotext descriptor [9] (see the appendix) which evolved into a videotext description scheme [10]. The **Videotext DS** is intended to provide information about *each text block* (defined as a group of characters appearing together as a single entity) in *one* particular image (be it a still picture or a single frame of a video sequence). In this section we will present the elements of the videotext description scheme in relation with MPEG-7 description schemes (DSs) although they are evolving during each MPEG meeting. The first version of the MPEG-7 generic visual description scheme is described in [12] and then modified into generic audio visual description scheme [21,24]. The generic AV DS describes the content of an image, video or audio sequence. The generic DS describes both the temporal and spatial aspects of the original content and can be used to describe images and video. The generic DS covers the *syntactic and semantic structure, model DS, summary DS, meta information and media DS*. The videotext DS relates to syntactic, semantic structure and meta information DSs as will be explained in the next subsections. For further detail the reader can refer to the paper on description schemes published in this issue.

2.1. Elements of videotext DS

The videotext description scheme contains information capturing the structural (syntactic) and semantic aspects of text appearing in video segments.

2.1.1. Structural elements of videotext DS

The structural (syntactic) elements of the videotext representation include:

- spatial information about the bounding box of a text region in a frame,
- text motion information,
- editing effects used to render the text, font color related information,
- text type (superimposed text or scene text),
- size and style characteristics of fonts used in composing the text,
- language used (e.g. Italian, English), and
- text string in character codes associated with the text region (see Appendix A for details).

These elements are closely related to the structural aspects of the generic DS, which specifies physical structures, and the signal properties of an image, video (or audio) program, such as color, texture, and motion. The first four elements – spatial, motion, editing effects, and color of the videotext – could be inherited from the MovingRegion DS which are being incorporated in the Generic DS document [24]. The information about text type, font size, type and language is represented with separate attributes or with suitably defined DSs specific to the videotext DS. We should observe here that character codes representing the text string can be automatically extracted using an OCR (as explained in Section 5 in document M4791, [9]) or manually inserted if automatic recognition becomes impractical. Note that font type can be either specified as a string ("Times New Roman Italic") or as an integer using a font look-up table.

Further examination of the MovingRegion DS points to the fact that the description schemes on mosaic and camera motion, which are optional,

generally do not apply to videotext and can be left empty. We believe that some of the MovingRegion DS fields such as time DS, will be an optional field when applied to the Videotext DS. We have decided that the videotext annotation should be associated with a video stream (tape) on a *frame-to-frame* basis. One could design a DS that is associated with text appearing in multiple frames, to inherit the time DS that indicates the duration of the persistence of this text over multiple frames in the stream. A Videotext DS associated with each frame will not result in a lot of extra storage. Basically, the DS will be the same for each frame where the same piece of text is present and a run-length encoding of the annotation stream can efficiently compress this stream. The advantage is that while editing the video, no new computations to reconcile missing and cut pieces of video have to be performed on the DSs to keep them valid. Browsing a video to find when text appears or disappears will be a little less efficient but this does not compare with the computational burden associated with keeping the annotation correct during video editing operations.

2.1.2. Semantic elements of videotext DS

A text string associated with a text region in a frame provides rich independent information about the contents of the frame such as the name of a person, location, topic of the event portrayed, football scores, etc. One can easily imagine a situation where objects and events present in a video frame (e.g. person's face, foul in a soccer match) can be directly linked to text strings extracted from the frame. For example, in anchor shots, an anchor's face is related to the anchor's name displayed in a text box in the bottom left corner of the frame. Therefore, videotext not only provides character strings (syntactic aspect) but also provides meaning to the frame as a whole (e.g. anchor shot) or objects in the frame (e.g. President Clinton).

Videotext has an associated role of providing additional informational links to existing objects in videos. For example, when a person is detected in a frame with a text region right below the face of the person, we usually infer that the text conveys the person's name. Similarly with scene text, the presence of text on an object shown in a frame tells the product name which may not be inferred from the shape or other attributes of the object (e.g. "Starbucks coffee" on a Styrofoam cup). In the context of the Generic DS, a semantic structure DS is used to specify semantic notions of image, video and audio content, in terms of semantic objects and events. The main element in the semantic description related to videotext representation is the object DS. The videotext object DS inherits from the object DS and has the additional attribute that provides the character string, which relates to the object. The character codes convey the "content" of the videotext.

As described in the generic DS document, the object DS contains an arbitrary number of object DSs, object type and annotation [24]. In the case of videotext object DS, we can state the object type to be "videotext." Under annotation the object can have free description such as "name of a person."

We have included character codes both in the syntactic and the semantic elements of the videotext DS [10]. Depending on the type of application, both character code placeholders can be meaningful. For example, a high-level application which uses the semantic information, needs to only access the videotext object DS part. On the other hand, the structural element of VideotextDS can be used to signal and describe the presence of videotext, which can be used in construction of visual summaries for efficient browsing applications. It is conceivable that there can be issues related to inconsistencies between the syntactic and semantic elements of this DS. We proposed that in the event of inconsistent data in character codes of syntactic and semantic parts of the DS, the semantic character codes override those in the syntactic part. We anticipate that there may not be a single solution to address this issue and it will be taken up for further discussion within the MPEG-7 Video, DDL, and DS subgroups.

The relationship between the syntactic and the semantic elements of the videotext DS will be specified by the syntactic-SemanticLinkDS definition to be further specified in the Generic DS document. According to the editor's note [24], the syntactic-semantic link may be replaced by a relation graph that would allow more complex relationships.

2.2. MPEG-7 requirements

The videotext feature and its associated videotext DS provide very powerful image and video annotations that satisfy many of the requirements noted in the "MPEG-7 Requirements Document" [22].

2.2.1. General requirements

Superimposed videotext DS can be looked upon as a valuable production feature [21] since it can provide production information pertaining to the entire length of a program in terms of its date of creation, names of producer, director, crew, and performers, their roles, etc., from processing credits sequences in a movie, sitcom, or other produced and edited footage, leading to cost-effective and scaleable solution to annotation of growing collections of videos. Moreover, videotext, especially when it occurs as part of a scene or an event shown in a video, allows for direct association of labels to objects and events occurring in the scene. For example, in sports and news programs, text indicating proper names, locations, and topics is often flashed beneath the persons shown in close-ups and that text most certainly refers to objective features of the persons themselves and to concept features such as title of events or activities associated with these persons. Finally, because of the fact that videotext is a visual feature that is added to the video during editing, videotext aids in deriving compositional information about the shots in which they appear. It also enables selective manipulation and editing of visual material in a straightforward fashion.

The videotext DS supports cross-modal queries. A query posed to retrieve videos depicting a certain person wearing attire of desired color can not only use color in its specification, but also the *Proper Name* of the person which can be extracted using videotext. The DS also allows for description of multiple pieces of information (sometimes called facts) about the same material. For example, when the videos from the CNN broadcast of the coverage of the impeachment trial of the President of the United States are examined, it can be seen that the text superimposed on those video frames not only provided the broad topic of the day covered, but also sub-topics, names of key players emerging as the day progressed. So videotext allows for the querying of the material at multiple granularities; first a query can be on a broad topic and successive queries can be on more specific issues related to the broader topic of the video. Extracted videotext may facilitate the correspondence of content descriptions to different temporal ranges, both hierarchically and sequentially. Sequential descriptors refer to successive time periods of, for example, a news program, i.e., different news stories which could be extracted with the use of videotext. Hierarchical descriptors, on the other hand, refer to the entire program data (e.g., a basketball game) and subsets of it (first through fourth-quarter). These subsets are often signaled using videotext and hence can be identified. The videotext DS can also include information relating to the language of the text, which can be manually determined. This additional knowledge leads to the capability of translating structured information provided by the videotext DS into a desired language, even if a complete translation of the text is not possible between different languages.

2.2.2. Functional requirements

The videotext DS supports many functionalities in content-based image and video retrieval. It supports text-based retrieval and search for specific content in images and videos related to spatial locations and other features of text. Further, it allows for rank ordering of retrieved items since the relevance of a retrieved video to the query can be computed using traditional rank measures from the information retrieval literature; however, the fact that there may be some uncertainty associated with automatically recognized videotext has to be somehow incorporated in the ranking measures used.

Clearly, if there is additional textual information associated with the images or video from closed-caption or audio recognition, videotext queries can be combined with those traditional text queries. Hence, the use of associated text information attached with images/videos is a useful complement, and can be combined with the visual text to allow for multimodal text queries, i.e., queries on text extracted from video frames in combination with topics generated from text associated with the

frames/images through other means. Videotext further enables interactive querying by the user by searching for matches of one query element at a time. That is, searching for pieces of videotext, one at a time, can interactively narrow down the search space. Interactive querying on videotext also enables user-selected prioritization of matches, where the user somehow indicates how relevant different pieces of visual information are to the query. Images or videos retrieved using videotext can be previewed to determine the relevance of the data to the query. Based on the relevancy, new queries can be formulated or the query can be refined. Smart browsing of image/video databases with the use of videotext is also possible.

2.2.3. Coding requirements

If image/video databases are indexed using the videotext DS, then the processing generates a set of n-tuples of facts for images/videos, which can be efficiently coded. Therefore, it easily satisfies MPEG-7 *coding requirements* relating to deriving efficient representations of data description. For retrieval purposes, video and images can be indexed through videotext in the same fashion as traditional text documents are indexed. Hash tables can be devised where the keys are text and the entries are pointers to images/videos.

2.2.4. Visual requirements

Videotext is not indicated in the MPEG-7 requirement document as a visual feature, such as, color, texture, and motion. The videotext DS however, supports a number of *visual requirements* by being impervious to the coding format of images and videos since visual text is typically extracted from full-resolution images/frames. Further, it is always relevant to the data, being associated information. The exception is a picture-in-picture video layout where each picture may have its own videotext associated with it. Picture-in-picture layouts include situations where there is a "ticker-like" graphic display at the bottom of the video – as in CNN Headline news. At the time of extraction many such instances of videotext may be detected.

3. Algorithms for text detection and extraction

Text can appear in video anywhere in the frame and in different contexts. It appears as either *scene text* or as *superimposed text*. Text that appears as part of the scene and recorded with the scene is referred to as scene text and its presence can be in the scene as part of street and shop name boards, or on a person's clothing. It is difficult to extract scene text reliably due to the unconstrained nature of its appearance. On the other hand, superimposed text is intended to carry and stress important information in video. It is typically generated by video title machines or graphical font generators in studios. The algorithms presented here are designed to extract superimposed text and scene text which possesses typical (superimposed) text attributes. We do not assume any prior knowledge about frame resolution, text location, font styles, and text appearance modes such as *normal* and *inverse* video. Some common characteristics of text are exploited in the algorithms including monochromaticity of individual characters, size restrictions (characters cannot be too small to be read by humans or too big to occupy a large portion of the frame), and horizontal alignment of text (preferred for ease of reading).

3.1. A computational framework for text extraction

Approaches to extracting text from video can be broadly classified into three categories: (i) methods that use region analysis to extract text components, (ii) methods that perform edge analysis to extract characters, and (iii) methods that use texture features to locate the presence of text. However, a common framework can be designed to describe a generic videotext extraction system. Given an image or a sequence of frames with color or gray-scale values, we propose the following framework that employs three major steps to extract text embedded in the frames.

Step 1: *Background removal*. From each frame, remove background (regions pertaining to non-text scene content) and obtain candidate text regions in the frame. This step can be accomplished in many ways. An approach employing region analysis will determine homogeneous (in color or gray scale)

and spatially connected components in the frame by grouping similar color or intensity valued pixels and discard too large or too small regions to retain only candidate text regions. An approach that employs edge analysis will detect edges in the frame and analyze their geometrical arrangement in the frame. Edges are filtered on the basis of edge directions to result in text lines only. Some approaches also use texture-based techniques to detect candidate text regions since text regions can be said to exhibit a special *texture* in terms of their contrast with the image background and their periodic horizontal intensity variations due to spacing of characters.

Step 2: *Text characteristics verification*. Once candidate text regions are extracted in the frame, candidate regions are analyzed individually, or first grouped using some form of connected component analysis and then analyzed. Some common characteristics of text are employed as criteria to examine whether candidate regions exhibit typical text attributes and to discard false positive text regions. These text properties may include monochromaticity of individual characters, character size restrictions, horizontal alignment of text, consistent intercharacter spacing, etc.

Step 3: *Consistency analysis for output*. The last step typically prepares the remaining text regions for the final usage intended for the detected text. Some approaches analyze regions to group characters to words, words to text lines, and obtain their bounding blocks. These approaches emphasize only automatic location of text, which therefore entails human involvement to recognize the characters present in the bounding blocks. Some approaches further analyze their text regions to obtain clean character boundaries so as to output frames ready for optical character recognition (OCR). These clean OCR-ready character bitmaps can then be directly input to a character recognition system for automatic recognition of the characters. The recognized text and its properties can then be stored and used as annotation indices for querying a video database.

In conclusion, different techniques may be employed to perform each of these three major steps to obtain character blocks in a video frame. Further, observe that some may directly work on compressed data at least during the first step [28], while others may require decompression of encoded digital video bitstreams into a sequence of color or intensity frames for analysis.

3.2. Text detection based on edge characterization

The algorithm for text detection presented in [2,6] is based on edge characterization according to the general framework presented in Section 3.1. The text properties are exploited on the connected components (CC) of the edges obtained, namely, the height, width, and area constraints. Further, horizontal alignment is used to merge multiple CCs into a single line of text. Finally, we output a thresholded image of the detected text lines with text as foreground in black on a white background. These regions can be further supplied as input to an OCR algorithm to output the text characters.

Text extraction is performed on individual video frames. In this section we will explain the steps involved in text extraction. The origin $(0, 0)$ of the frame is the top left corner. A pixel is referenced by (x, y) where x is the position in columns and y in rows.

3.2.1. Channel separation

The first step is to take an input image from a video capture board. The choice of color space depends on the ease of color extraction. Currently, we are using the red frame of the RGB color space to make it easy to differentiate the colors white, yellow, and black, which are predominantly used for text in video. By using the red frame we obtain sharp high contrast edges for the frequent text colors. However, other color spaces can be used such as HSB or YUV.

3.2.2. Image enhancement

Before any further processing is performed, the frame's edges are enhanced using a 3x3 mask [13]. Salt and pepper noise removal is performed to remove any remaining noise. For this purpose we use a median filter [13].

3.2.3. Edge detection

On the enhanced image, edge detection is performed using the following 3×3 mask.

$$\begin{matrix} -1 & -1 & -1, \\ -1 & 12 & -1, \\ -1 & -1 & -1, \end{matrix}$$

The following equation gives the formula used for edge detection.

$$\sum_{i=-1}^{1} \sum_{j=-1}^{1} w_{i,j} F_{x+1,y+j} < \text{EdgeThreshold},$$

where $w_{i,j}$ are the weights from the edge mask and $F_{.x+i,y+j}$ represents a pixel of the image F.

Edge thresholding is performed using a preset threshold (e.g. 200 for pixel values ranging from 0 to 255). The top and the bottom rows and the left and the right columns of the image are ignored for finding the edges. Currently, the threshold is a fixed one, however, a variable threshold can be used. A fixed threshold currently results in a lot of salt and pepper noise. Also, the edges around the text may be broken and not connected, resulting in a split character.

3.2.4. Edge filtering

Once the edges are detected, a preliminary edge filtering is performed to remove areas, which possibly do not contain text, or, even if they do, they cannot be reliably detected. Edge filtering can be performed at different levels. One is at a frame level and the other is at a sub-frame level. At the frame level, if more than a reasonable portion (e.g. over 25%) of the frame is composed of edge pixels, which might be due to high density of objects in the frame, we disregard that frame and take the next input frame. Using frame level needs only one counter to be maintained to keep the count of the number of edge pixels in the image. This technique however, can lead to the loss of text in some clean areas in an image and may result in false negatives. In order to overcome this problem, edge filtering is performed at a sub-frame level. To find text in an "overcrowded" frame, we maintain six counters to keep the count of the subdivided portions of the image. Three counters are used for three vertical portions of the image (one for each third of the area of the frame). Similarly, three counters are used for three horizontal stripes. Text lines found in high-density edge areas (stripes) are rejected at a subsequent step. This filtering can be performed using smaller areas, in order to retain areas that are clean and contain text in a region smaller than one-third of an image.

3.2.5. Character detection

A connected component (CC) analysis is performed on edges generated in the previous step. Each character is assumed to give rise to a connected component or a part thereof. All the edge pixels that within an eight-pixel neighborhood of an edge pixel are merged into a single CC structure. This CC structure contains the location of the pixels that are connected together. This structure also contains the value of the leftmost, rightmost, top, and bottom pixel in the structure along with the location of the center in the x and y directions. It also contains the count of the pixels giving rise to the CC. This is the area of the CC. There are preset thresholds for the maximum and minimum limits for area, height and width allowable for a CC to be passed on to the next stage. Each of the CCs is tested for size, height and width criteria before passing on to the next stage.

3.2.6. Text box detection

The connected components that pass the criteria in the previous step are sorted in ascending order based on the location of the bottom left pixel of the CC box. The bottom left pixel's x location is the left-most pixel of the CC and y is the lower-most pixel of the CC. This list is traversed and the CCs are merged together to form boxes of text. The first connected component, CC_1 is assigned to the first box. Each subsequent CC_i is tested to see if the bottom most pixel lies within a preset acceptable "row" threshold from the bottom most pixel of the current text box. If the CC_i lies within a few rows (in our case 2 rows) of the current box, there is a good chance that they belong to the same line of text. The row difference threshold, currently is a fixed one, but could be a variable one also. It could be made a fraction of the height of the current text box. In order to avoid merging CCs that are too far away in the image, a second test is

performed to see if the column distance between CC_i and the text boxes is less than a column threshold. This threshold is variable and is a multiple of the width of CC_i. CC_i is merged to the current text box if the above tests are passed. If CC_i does not merge into the current text box, then a new text box is started with CC_i as its first component and the traversing is continued.

The above process could result in multiple text boxes for a single line of text in the image. Now for each of the text boxes formed by the character merging, a second level of merging is performed. This is to merge the text boxes that might have been mistakenly taken as separate lines of text, either due to strict CC merging criteria or due to poor edge detection process resulting in multiple CCs for the same character.

Each box is compared to the text boxes following it for a set of conditions. If two boxes are merged, the second box is deleted from the list of text boxes and merged into the first box. The multiple test conditions for two text boxes are: (a) The bottom of one box is within the row difference threshold of the other. Also the distance between the two boxes in the horizontal direction is less than a variable threshold depending on the average width of characters in the first box; (b) the center of either of the boxes lies within the area of the other text box; or (c) the top of the first text box overlaps with the bottom of the second text box and their left sides are within few pixels as well as their right sides. If any of the above conditions is satisfied, the two text boxes are merged until all text boxes are tested against each other.

3.2.7. Text line detection and enhancement

The text boxes obtained from the previous step are accepted as text lines if they conform to the constraints of area, width and height. For each of the text boxes, we extract the sub-image corresponding to the text box from the original image. This sub-image is now thresholded in order to obtain the text as foreground in black and everything else around it as background in white. This convention is required so that the output of this stage can be fed to an OCR. Thresholding is performed as follows: The average grayscale value of the pixels in the text box is calculated. The average

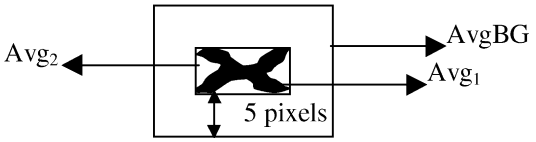

Fig. 1. Separation of text foreground from background.

gray scale, avgBG, value of a region (five pixels in our case) around the text box is also calculated. Within the text box, anything above the average is marked as white and anything below it is marked as black. The gray-scale average for the pixels being marked as white, avg_1, is calculated along with the gray-scale average for the pixels being marked as black, avg_2. Once, the text box is converted to a binary image, the average of the "white region"(avg_1) and the average of the "black region"(avg_2) are compared to the avgBG (as shown in Fig. 1). The region that has its average closer to the avgBG is assigned to be the white background and the other region is assigned to be the black foreground. We illustrate the steps of the text detection process in Fig. 2. Fig. 2(a) shows a sub-image from original video frame of size 320×240 extracted from a movie from the MPEG-7 test content set. Fig. 2(b) shows image after edge detection step. Fig. 2(c) shows the image after edge thresholding, and connected components analysis. Fig. 2(d) shows the sub-image after connected components are merged into text boxes. Fig. 2(e) is the resultant image once non-text regions are removed. Fig. 2(f) shows the thresholded image containing text as black letters on white background. This image is ready for use as an input to an OCR system where automatically recognized characters are shown.

3.3. Videotext extraction using region analysis

The algorithm for text detection presented in [8,32] is based on region analysis according to the general framework presented in Section 3.1. The input is a gray-scale image or a sequence of gray-level images from a video stream. The primary goals of the system are (i) isolating regions that may contain text characters in an image from other image contents, (ii) separating each character

Fig. 2. (a) The original sub-image, (b) edge image, (c) accepted connected components, (d) CCs merged in the same text box, (e) locally thresholded image containing text as black on white.

region from its surroundings, and (iii) verifying the presence of text by consistency analysis. The terms "image" and "frame" are used interchangeably.

3.3.1. Candidate text region extraction

The objective of this first step is to remove background from an input gray-scale image where the background is interpreted as containing regions such as faces of the speakers, players, and other non-text scene contents. The generalized region labeling (GRL) algorithm [29] is employed to extract homogeneous regions from the input image. This algorithm labels pixels in an image based on a given criterion (e.g. gray-scale homogeneity) using contour traversal, thus partitioning the image into multiple regions. It then groups pixels belonging to a region by determining its outer and inner boundaries, if there are holes within the region, and extracts region features such as its minimum bounding rectangle (MBR), area, mean gray level, etc. The criterion used to group pixels into a region is that the gray level difference between any pair of pixels within the region cannot exceed ± 10.

By using the GRL algorithm to segment the image into non-overlapping homogenous regions, we have obtained complete region information such as its label, outer and inner boundaries, number of holes within the regions, area, average gray level, gray-level variance, centroid, and the MBR. Having obtained a number of homogenous regions in the image, non-text background regions are removed based on their size. For example, since text fonts are usually not larger than 24×32 in a 320×240 (SIF resolution) image, a region is removed if the width and height of its MBR are greater than 24 and 32, respectively. This size constraint can be adaptively modified depending on the image resolution by maintaining a knowledge base of typical ranges of text proportions observed with different image resolutions. Thus, by employing a constraint that emphasizes the spatial proportion of text characters rather than the region area which is often used by other researchers, large regions of homogeneity that are unlikely to be text are effectively removed. Within the remaining candidate regions, characters can be fragmented into multiple regions because of varying contrast in regions surrounding the characters. In order to group multiple touching regions into a single character region, we generate a binary image from the labeled region image where all the regions which do not satisfy the size constraint are marked "0" and the remaining regions are marked with "1". This binary image is processed using the GRL algorithm with a new grouping algorithm to obtain new connected regions. With the creation of a binary image, followed by a relabeling step, many small connected fragments of a whole candidate text region (that is likely to be a single character) are merged together.

3.3.2. Text region refinement

In this stage, the basic idea is to apply appropriate criteria to extract character segments within the candidate regions. Within a region, characters can be present embedded in a complex background and since OCR systems require text to be printed against a clean background for processing, the second stage attempts to remove the background within the regions while preserving the text. Since character outlines in these regions can be degraded and merged with the background, an iterative local thresholding operation is performed in each

candidate region to separate the text from its surroundings and from other extraneous background contained within its interior. Once thresholds are determined automatically for all candidate regions, we compute positive and negative images, where the positive image contains region pixels whose gray levels are above their respective local thresholds and the negative image contains region pixels whose gray levels fall below their respective thresholds. Observe that the negative image will contain candidate text regions if that text appears in inverse video mode in the input. All the remaining processing steps are performed on both positive and negative images and their results are combined at the end of the last stage.

We further sharpen and separate the character region boundaries by performing a region boundary analysis. This is necessary especially when characters within a text string appear connected with each other and they need to be separated for accurate text identification. This is achieved by examining the gray-level contrast between the character region boundaries and the regions themselves. For each candidate region R, a threshold T is computed using the following equation:

$$\sum_{i=-1}^{1}\sum_{j=-1}^{1} w_{i,j}F_{x+i,y+j} < \text{EdgeThreshold},$$

where I_{cbk} is the gray level of the pixel k on the circumscribing boundaries of the region and I_{il} is the gray level of the pixel l belonging to R (including interior and region boundaries), N_{cb} is the number of pixels on the circumscribing boundaries of the region, and N_i is the total number of pixels in the region. A pixel is defined to be on the circumscribing boundary of a region if it does not belong to the region but at least one of its four neighbors (using four-connectivity) does. Those pixels in R whose gray level is less than T are marked as belonging to the background and discarded, while the others are retained in the region. Note that this condition is reversed for the negative image. This step is repeated until the value of T does not change over two consecutive iterations.

3.3.3. Text characteristics verification

The few candidate character regions that remain in the image are then subject to a verification step where they are tested for exhibiting typical text font characteristics. A candidate region is removed (i) if its area or its height is very small (e.g. its area is less than 12 or its height is less than four pixels for a SIF resolution frame) because small fonts are difficult for OCR systems to recognize; (ii) if the fill factor which is computed as the ratio of the area of its MBR to the region area, is greater than 4; (iii) if the gray-level contrast with the background is low, i.e., if

$$\sum_{i=-1}^{1}\sum_{j=-1}^{1} w_{i,j}F_{x+i,y+j} < \text{EdgeTbl},$$

where I_{cbk} is the gray level of the pixel k on the circumscribing boundaries of the region and I_{bl} is the gray level of the pixel l on the boundaries of the region. Note that the region boundaries (both outer and inner, if the region has holes) have already been obtained using the GRL algorithm and therefore this new boundary-based test can be easily performed to handle the removal of noisy non-text regions.

3.3.4. Text consistency analysis

Consistency between neighboring text regions is verified to eliminate false positive regions. Unlike many systems, our system attempts to ensure that the adjacent regions in a line in the image exhibit the characteristics of a text string, thus verifying the global structure of a row of text in a local manner. This text consistency test includes (i) position analysis that checks intercharacter spacing. For a SIF resolution frame, the width between the centroids of the MBRs of a pair of neighboring character regions that are retained is less than 50 pixels; (ii) horizontal alignment analysis of characters. The vertical centers of the MBRs of neighboring characters is within six pixels of one another; (iii) vertical proportions analysis of adjacent character regions. The height of the larger of the two regions is less than twice the height of the smaller region.

Given a candidate text string, we also perform a final series of tests involving the MBRs of characters present in the string. The MBRs of the regions (characters) are first verified to be present along a line within a given tolerance of two pixels. Observe that characters present along a diagonal line

can be easily identified as a string by our system. The intercharacter distance in the string is verified to be less than 16 pixel. We also ensure that the MBRs of adjacent characters do not overlap by more than two pixels. If all three conditions are satisfied, we retain the candidate word region as a text string. The final output is a binary image containing the text characters that can be directly used as input to an OCR system to get the text string in ASCII and also a text file containing information about features of the character regions. Fig. 3 illustrates the entire sequence of processing of a video frame in our system.

Fig. 3(a) shows an original video frame of size 320×240 extracted from an MPEG-1 encoded IBM news video bitstream. Fig. 3(b) shows a binary image containing candidate text regions shown in black. Fig. 3(c) shows locally thresholded text regions. Fig. 3(d) shows the image containing boundary refined text regions. Fig. 3(e) is the resultant image once non-text regions are removed. Fig. 3(f) shows the characters extracted using text consistency analysis. Fig. 3(g) is the screen dump of the output of an experimental Cunei3.0 OCR system where automatically recognized characters are shown. The highlighted words in this image only signify that they were to be added to the OCR dictionary.

3.4. Interframe analysis for text refinement

Intraframe processing can be followed by an optional interframe verification [30] as text in video streams persists over multiple consecutive frames. Given a sequence of images from a video stream, text regions determined from five consecut-

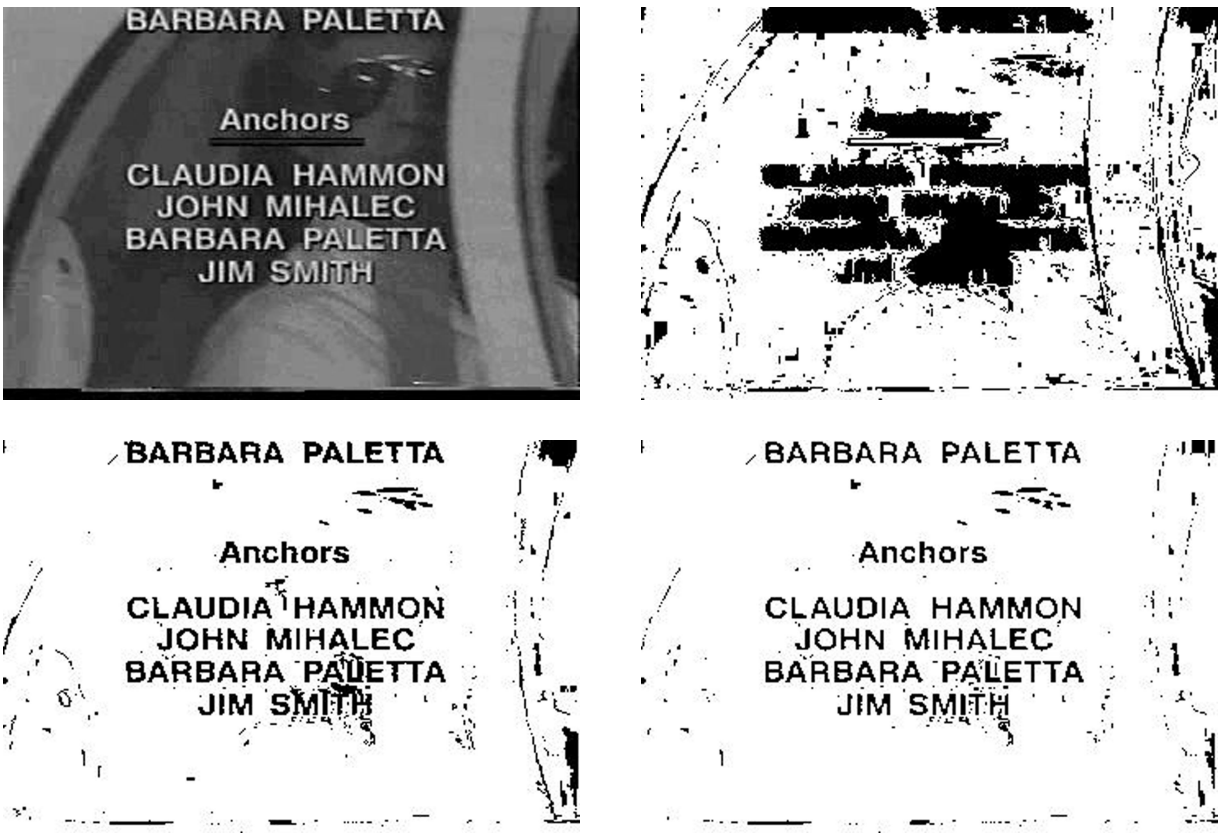

Fig. 3. Videotext extraction results.

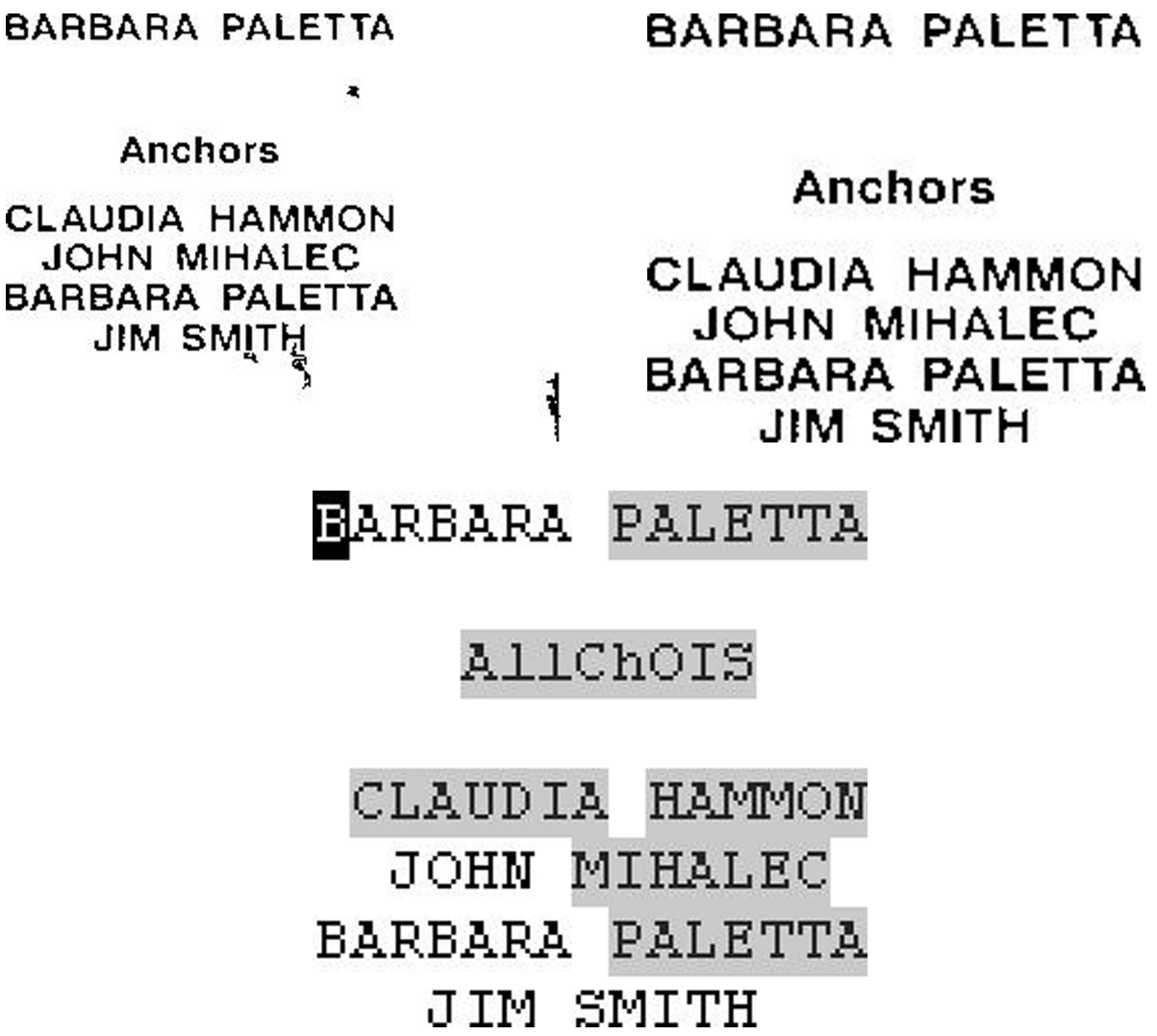

Fig. 3. Continued.

ive frames are analyzed together to add missing characters in those frames and to delete incorrect regions posing as text. This interframe analysis involves examination of the similarity of valid text regions in terms of their positions, intensities, and shape features and aids in omitting false positive regions. Each set of five consecutive frames is analyzed to verify the persistence of the presence of text strings. In each frame, we construct a structure called the *character group line* (CGL) [30] which is a line connecting the centers of gravity (CoG) of adjacent character regions that are horizontally aligned and that also satisfy neighborhood proximity constraints in terms of intercharacter and interword spacing. A frame may contain multiple CGLs in different scan lines. Given a frame and its character group lines, each CGL is matched with those in the four neighboring frames to ensure that the same CGL is present in at least three frames. Those CG lines that persist over three frames and that are within the allowed variations in their CoG positions and their lengths are retained to output their

corresponding characters while others are omitted as noisy regions owing to lack of sufficient neighborhood support. This analysis has shown to result in the reduction of false positive rates in text extraction.

4. Experimental results

Both the region-based (henceforth denoted as RB) and edge-based (EB) videotext extraction systems have been implemented and tested on various types of video content. The RB algorithm has been implemented in Visual C++ on a personal computer with a 133 MHz Pentium processor and 32 MB memory, running *Windows 95*. This system extracted text from each SIF resolution frame in about 1.7 s on the average and this certainly can be improved with code optimization and a faster processor. The EB algorithm has been implemented in Visual C++ on a personal computer with a 450 MHz Pentium processor and 256 MB memory, running *Windows NT*. This system extracted text from each SIF resolution frame in about 0.033 s on the average.

The experimental data sets included commercials, news, sports, program credit sequences, etc. For the experiments reported here, and in order to understand the relative strengths and weaknesses of these two systems, a subset of video frames were chosen from the MPEG-7 test data and both the systems were tested with this data set. A worker from the Philips organization manually extracted about 92 different frames containing superimposed text and embedded text from the MPEG-7 data set. In this set of frames, 22 frames contained either opening or closing credits and 30 frames contained the names of anchorpersons or game show hosts. Also, 40 different frames were analyzed, some of which contained scores from a football game and some contained the TV station name and the logo. Two persons independently determined the ground truth on the videotext in these test frames. Tables 1 and 2 summarize the results of RB and EB systems, respectively, on this common data set. The second column of Table 1 gives the number of frames in each category. Third column provides the total number of ground truth text lines in each category. Fourth column is the count of the correctly identified text lines. Numbers pertaining to false positives (non-text regions *incorrectly* identified as text lines) and the missed text lines (text lines that were not detected by the systems) are given in the next two columns.

The results of the RB system on a subset of frames selected from the MPEG-7 data set are shown in Table 1 and the results of the EB system on the same subset of frames are shown in Table 2.

Table 1
Performance of the RB system on the test data from the MPEG-7 data set.

Data set	Total number of frames	Total number of text lines	Number of correct text lines	Number of false positive text lines	Number of missed text lines
Names	30	81	71	80	10
Misc.	40	130	106	102	24
Credits	22	88	78	24	10

Table 2
Performance of the EB system on the test data from the MPEG-7 data set.

Data set	Total number of frames	Total number of text lines	Number of correct text lines	Number of false positive text lines	Number of missed text lines
Names	30	81	61	20	20
Misc.	40	130	80	9	50
Credits	22	88	51	10	37

As can be observed by from these two tables, the RB system does very well in terms of low miss rate by not missing many of the true text lines, while the EB system fares worse by missing as much as two to three times as that of RB. On the other hand, RB detects more false positive text regions than EB. Observe that these experiments were conducted without utilizing the interframe-based refinement (discussed as part of the RB algorithm) to ensure a fair comparison between the two algorithms. We anticipate that in conjunction with this refinement step, many small noisy regions that remain as false positives can be eliminated.

Depending on whether the end application's emphasis is on *completeness* (implying that the miss rate must be zero) or on *soundness* (which implies that the false positive rate must be zero), one of these two algorithms can be chosen to perform automatic videotext extraction. For example, in videotext-based search and retrieval applications, it is crucial that the result list returned by a system with a query based on a keyword extracted from the videotext does not miss correct items from the video collection; hence low videotext miss rate becomes critical for a system that performs automatic videotext extraction. The RB system will be a very good fit for such an application. Any false positive text can also be easily corrected with human verification of the automatically generated videotext outputs. The additional strength of this system also lies in the fact that a prototype video character recognition engine is being built to recognize the videotext frames and to generate ASCII strings of the recognized text. This is described in the next section.

In applications where fast browsing of video is needed, for example based on the presence of text in the frames, too many falsely retrieved videos based on falsely identified videotext can become a bottleneck. In such a situation the EB system can be used wherein it not only results in low numbers of falsely detected text regions but also detects text in video frames quickly.

5. Video character recognition

A video character recognition engine is required to build a completely automated end-to-end videotext extraction and recognition system. Though many approaches for videotext-based content annotation systems believe and claim that their extracted videotext will be easily recognized by commercial OCRs to generate the final output in ASCII strings, our experiments with many commercial OCRs tested on OCR-ready videotext bitmaps did not result in sufficient recognition accuracy to support this belief. Commercial OCRs require images of at least 300 dpi (dots per inch) resolution in order to achieve good recognition results, while images, especially from videos can be very low in resolution. This leads to severe problems during recognition. Further with video frames, extracted videotext bitmaps may not be as clean – with characters fully separated from one another, and fully connected without any intracharacter breakage – as required by many of these OCRs for accurate recognition. The encoding and the decompressing process involved with digital videos stored in compressed form can also introduce artifacts in videotext extraction, thus resulting in poor recognition accuracy. Hence, efforts are underway [31] to build a video character recognition engine in-house that can work with OCR-ready videotext bitmaps to provide a fully automated system for videotext extraction and recognition.

6. Discussion

We have described the *videotext* feature proposed as a description scheme to the MPEG-7 standardization committee. First, the syntax and semantics of the videotext DS are introduced and its relationship with respect to the generic visual DS and how videotext satisfies MPEG-7 requirements. Then we briefly introduce two algorithms for extracting text from video: an edge-based method proposed by Philips and a region-based method proposed by IBM. Experimental results of both algorithms on a common dataset are given for automatic text detection.

We have described those applications and tools where videotext is of merit. From the wide range of applications that cover video indexing and annotation, commercial detection, and scene analysis it can be concluded that the videotext feature is quite

powerful. Of all the descriptions that can be automatically extracted, videotext is probably the one that gives the highest-level semantic information. That is, it gives annotations that could refer to events and objects in the video, which otherwise cannot be automatically extracted with today's video content analysis technology. For archived video, the information extracted using videotext can automatically be inserted in the meta-data fields since title and credits in TV programs and movies are conveyed with superimposed text.

7. Unlinked References

[11,16,27]

Acknowledgements

Chitra Dorai and Ruud Bolle would like to gratefully acknowledge the contributions of Professor Jae-Chang Shim from Andong National University, South Korea to the design and testing of IBM's videotext extraction algorithm. Nevenka Dimitrova and Lalitha Agnihotri would like to thank Herman Elenbaas from Philips Consumer Electronics for his contribution to the Philips' text detection algorithm.

Appendix A. Videotext information representation

The videotext feature records the presence of text blocks, optionally recognized text, locations of text blocks, movement of the text (blocks), font information, and editing information in a frame of a video of a scene. This is achieved through the use of description scheme that can be derived from the generic audio visual description scheme. The syntax and the specification for each field are given below. The description scheme evolved from the two coalesced videotext descriptors. In this appendix we present the essence of videotext information descriptor [10].

There are several ways in which each of the fields can be specified that may be appropriate depending on the end-use of the material. The simplest is describing each character block region that provides a single text string in each frame using a data structure (see Fig. 4) containing the following fields:

character_code: The text string as conveyed by the character block (text box) present in the frame in terms of international character codes. This is the ONLY required filed. If the string is set to the empty_string it means that no text has been recognized in the image or frame.

text_type: An optional Boolean value which is TRUE if the text present has been generated using video title machines or graphical font generators,

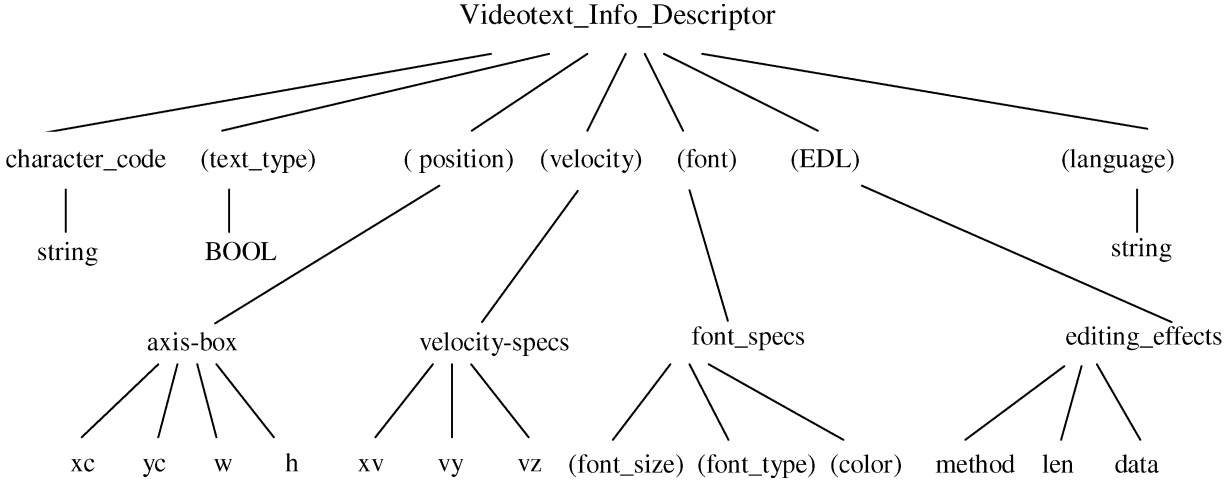

Fig. 4. Diagram of Videotext_Info_Descriptor.

i.e., when the text is superimposed text. The Boolean value is FALSE if the text is part of the scene (embedded text) and is recorded with the scene in the field. It is set to NULL if the type cannot be determined.

position: This optional structure represents an axis-parallel minimal rectangular box bounding the text in the image. It points to the array axis_box structure and may be set to NULL if the position has not been extracted or the system is not interested in recording this detail for the text block. This position is described with the minimum bounding box enclosing the character block in the frame in appropriate dimensional units. These four fields (xc, yc, w, h) together form the spatial designator for the region containing the character block. This is useful to know as the location of the text string can provide clues about the category of the text such as caption, channel number, sports score, etc. The methodology for measuring image coordinates to compute these fields is a generic problem and may already be defined elsewhere in the MPEG-7 specification. A reasonable suggestion would be to measure not in pixels, but in fractional dimensions. This allows the content to be resized to whatever formats the application demands without requiring the annotation stream to be altered. However, such a choice still has a problem with different picture aspect ratios (e.g. 16 : 9 vs. 4 : 3) and individual pixel aspect ratios (e.g. square vs. CCIR 601). Therefore, it is advantageous to measure from the center of the image rather than a corner in order to automatically handle "letter-boxing" (converting 16 : 9 aspect ratio material to fit a 4 : 3 screen by adding black bars on the top and bottom). Possibly the best choice is to declare one of the images dimensions as the "reference dimension" and measure all values relative to this.

velocity: This field allows the system to record the velocity of the movement of the text block from frame to frame. The field points to the array velocity_specs and may be set to NULL if the velocity is unknown. If the text is stationary, the zero velocity can be explicitly indicated with the structure. This field is optional. The vx, vy, vz together describe the velocity by which the text block translates and zooms in from frame to frame. It can be useful for determining text appearance and disappearance points or for finding camera zoom (this effect often employed to grab a viewer's attention). These features do not make sense for images that are analyzed in isolation as opposed to those that are considered as part of a sequence. Once again, the units of this measurement need to be normalized for image shape, pixel shape, and frame rate. A convenient choice is to use the frame-to-frame differences in the normalized spatial coordinates described above and then multiply by the frame rate. Thus, the velocity tells how far across the image a character block would move in 1 second (assuming its speed is constant).

Font: This field allows the annotation of the type of font and font size that has been used. It is an optional variable and may be set to NULL.

EDL: This field allows the system to record some or all of the information that is available about the edit decision list (EDL) pertaining to the frame. The field points to the array editing_effects and may be set to NULL. The structure is essentially a "blob" so that various EDL description methods, which are globally defined, can be used. This structure is a generic wrapper for a "blob" to describe the editing effects used with the text and it can contain many types of EDL data. This includes text, text location, font information, moving text, stationary text, vertical scrolling, horizontally scrolling, and other effects. This is a way of incorporating videotext information if the text is not automatically extracted but already known from the edit process.

language: This optional field allows the system to record the language of the text.

References

[1] M. Abdel-Mottaleb, N. Dimitrova, R. Desai, J. Martino, CONIVAS: Content-based Image and Video Access System, Proc. of ACM Multimedia, Boston 1996, pp. 427–428.

[2] L. Agnihotri, N. Dimitrova, Text Detection for Video Analysis, Workshop on Content Based Image and Video Libraries, held in conjunction with CVPR, Colorado, 1999, pp. 109–113.

[3] D. Beeferman, A. Berger, J. Lafferty, Text segmentation using exponential models, Proceedings of Empirical Methods in Natural Language Processing, AAAI 97, Providence, RI, 1997.

[4] S.-F. Chang, W. Chen, H.E. Meng, H. Sundaram, D. Zong, VideoQ: An automated content based video search system using visual cues, Proceedings of the ACM Multimedia, Seattle, 1994, pp. 313–324.
[5] M. Christel, T. Kanade, M. Mauldin, R. Reddy, M. Sirbu, S. Stevens, H. Wactlar, Informedia digital video library, Comm. ACM, 38 (4) (1995) 57–58.
[6] N. Dimitrova, T. McGee, H. Elenbaas, J. Martino, Video content management in consumer devices, IEEE Trans. Knowledge Data Eng. November 1998.
[7] N. Dimitrova, L. Agnihotri, Text detection and representation, ISO/IEC/JTC1/SC29/WG11/P639, February 1999.
[8] C. Dorai, R. Bolle, Videotext descriptor, ISO/IEC/JTC1/SC29/WG11/P183, February 1999.
[9] C. Dorai, R. Bolle, L. Agnihotri, N. Dimitrova, MPEG-7 Videotext Descriptor Proposal, Doc. ISO/MPEG M4791, MPEG Vancouver Meeting, July 1999.
[10] C. Dorai, R. Bolle, N. Dimitrova, L. Agnihotri, MPEG-7 Videotext Description Scheme, Doc. ISO/MPEG M5206, MPEG Melbourne Meeting, October 1999.
[11] U. Gargi, S. Antani, R. Kasturi, Indexing text events in digital video databases, International Conference on Pattern Recognition, Brisbane, August 1998, pp. 916–918.
[12] Generic Visual Description Scheme for MPEG-7, ISO/IEC/JTC1/SC29/WG11/N2694, March 1999.
[13] R.C. Gonzalez, R.E. Woods, Digital image processing, Addison-Wesley, Reading MA, 1992.
[14] A. Hauptmann, M. Smith, Text, speech, and vision for video segmentation: the informedia project, AAAI Fall 1995 Symposium on Computational Models for Integrating Language and Vision 1995.
[15] A.K. Jain, B. Yu, Automatic text location in images and video frames, Proc. IEEE Pattern Recognition 31 (12) (1998) 2055–2076.
[16] R. Lienhart, C. Kuhmunch, W. Effelsberg, On the detection and recognition of television commercials, Proceedings of the IEEE International Conference on Multimedia Computing and Systems, 1997, pp. 509–516.
[17] R. Lienhart, F. Stuber, Automatic text recognition for video indexing, SPIE Conference on Image and Video Processing, January 1996.
[18] M.K. Mandal, T. Aboulnasr, S. Panchanatan, Image indexing using moments and wavelets, IEEE Trans. Consumer Electron. 42 (3) (August 1996).
[19] K.V. Mardia, T.J. Hainsworth, A spatial thresholding method for image segmentation, IEEE Trans. PAMI 10 (1988) 919–927.
[20] T. McGee, N. Dimitrova, Parsing TV programs for identification and removal of non-story segments, SPIE Conference on Storage and Retrieval in Image and Video Databases, San Jose, January 1999.
[21] MPEG-7 Description Schemes, ISO/IEC/JTC1/SC29/WG11/N2844, July 1999.
[22] MPEG Requirements Group, MPEG-7 Requirements Document, Doc. ISO/MPEG N2461, MPEG Atlantic City Meeting, October 1998.
[23] MPEG Requirements Group, MPEG-7 Context and Objectives, Doc. ISO/MPEG N2460, MPEG Atlantic City Meeting, October 1998.
[24] MPEG-7 Description Schemes (V0.6), ISO/IEC/JTC1/SC29/WG11/M5040, Version 0.6-a, September 1999.
[25] J. Ohya, A. Shio, S. Akamatsu, "Recognizing characters in scene images," IEEE Trans. Pattern Anal Mach Intell, 16 (February 1994) 214–224.
[26] A. Perez, R. Gonzalez, An iterative thresholding method for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 9 (1987) 742–751.
[27] S. Pfeiffer, R. Lienhart, S. Fischer, W. Effelsberg, Abstracting digital moves automatically, J. Visual Commun. Image Representation 7 (4) (1996) 345–353.
[28] B. Shen, Ishwar K. Sethi, Convolution-based edge detection for image/video in block-DCT domain, J. Visual Commun Image Representation 7 (4) (December 1996) 411–423.
[29] J.-C. Shim, C. Dorai, A generalized region labeling algorithm for image coding, restoration, and segmentationfast and generalized region labeling algorithm, in: Proceedings of the International Conference on Image Processing, Kobe, Japan, October 1999, to appear. Working Document, IBM T. J. Watson Research Center, Yorktown Heights, 1998.
[30] J.-C. Shim, C. Dorai, Interframe analysis for improved text extraction from video, Working Document, IBM T.J. Watson Research Center, Yorktown Heights, January 1998.
[31] J.-C. Shim, C. Dorai, An end-to-end video character recognition system for automated video annotation and search, Working Document, IBM T.J. Watson Research Center, Yorktown Heights, August 1999.
[32] J.-C. Shim, C. Dorai, R. Bolle, Automatic text extraction from video for content-based annotation and retrieval, Proceedings of the International Conference on Pattern Recognition, 1998, pp. 618–620.

Nevenka Dimitrova is a Principal Member of Research Staff at Philips Research. She obtained her Ph.D. (1995) and MS (1991) in Computer Science from Arizona State University (USA), and BS (1984) in Mathematics and Computer Science from University of Kiril and Metodij, Skopje, Macedonia. Her main research interests are in the areas of consumer information management, digital television, video content analysis and retrieval, compression based retrieval, MPEG-7 and their applications in advanced multimedia systems. She is on the editorial board of IEEE Multimedia,

and IEEE Communications Society on-line magazine. She has served on program committees of 15 conferences including ACM Multimedia, IEEE IPCCC, SPIE storage and Retrieval in Media Databases, SPIE Conference on Multimedia Storage and Archiving Systems, and others.

Lalitha Agnihotri is a Member of Research Staff at Philips Research. She received her MS in 1998 from Pennsylvania State University and Bachelor of Engineering from New Delhi University, India in 1996. Her research activities include computer vision, video analysis, audio analysis and characterization of video, specialized media processors for real-time video processing, and MPEG-7. She has performed extensive research in text detection and recognition including the first OCR for Gujarati script.

Chitra Dorai is a Research Staff Member with the Exploratory Computer Vision and Pattern Recognition group at the IBM Thomas J. Watson Research Center, Hawthorne, New York. She received the B. Tech. degree from the Indian Institute of Technology, Madras, in 1987, the M.S. degree from the Indian Institute of Science, Bangalore, in 1991, and Ph.D. from the Department of Computer Science at Michigan State University, in 1996. Her research interests are in the areas of digital video analysis, computer vision, pattern recognition, multimedia systems, computer graphics, and neural networks. Her recent work on automatic videotext extraction won the Best Industrial-Related Paper Award at the International Conference on Pattern Recognition in August 1998. Her Pattern Recognition Journal article in 1997 received honorable mention in the 24th Annual Best Paper Award Contest of the Pattern Recognition Society. She is a member of the IEEE, IEEE Computer Society, and the ACM.

Ruud M. Bolle was born in Voorburg, The Netherlands. He received the Bachelor's Degree in Analog Electronics in 1977 and the Master's Degree in Electrical Engineering in 1980, both from Delft University of Technology, Delft, The Netherlands. In 1983 he received the Master's Degree in Applied Mathematics and in 1984 the Ph.D. in Electrical Engineering from Brown University, Providence, Rhode Island. In 1984 he became a Research Staff Member at the IBM Thomas J. Watson Research Center in the Artificial Intelligence Department of the Computer Science Department. In 1988 he became manager of the newly formed Exploratory Computer Vision Group which is part of IBM's Digital Library effort. Currently, his research interests are focussed on video database indexing, video processing, visual human-computer interaction and biometry applications.

Ruud M. Bolle is a Fellow of the IEEE and the AIPR. He is on the Advisory Council of IEEE TPAMI; he is Associate Editor of Computer Vision and Image Understanding, the Journal of Mathematical Imaging and Vision, and Pattern Recognition. Ruud M. Bolle is a Member of the IBM Academy of Technology.