4.1 Introduction

Compared with the text in still images, the text occurring in video documents has two valuable properties for text extraction: (i) chronological redundancy, a text typically lasts for more than one second to give human viewers the necessary time to read it. For example, a text in the any standard national television program, whose refresh rate is 30 frames per second (FPS), appears in at least 30 consecutive video frames; (ii) spatial similarity, although a text may have some motions due to zooming, rotation, and scrolling in video, the change of its spatial locations between two neighboring frames is typically small in order to keep the smooth motion of the text event to allow human reviewers read text easily.

Based on the above two properties of text in video document, we define that the text objects appearing in consecutive frames with the same characters and similar spatial locations form a text event. The goal of text tracking is to follow a text event as it moves or changes over time and determines its temporal and spatial locations and extents. Moreover, every text event is given a unique ID in order to build an index of all text events occurring in video for content-retrieval purpose [2] [15].

In addition, text tracking can also serve as a text verification step to
remove false positives. We know that a text event persists for multiple frames. Therefore, if a text object is detected in a region in one frame but no text object detected in the neighboring regions in the previous or next frames, this text object should be discarded as a false positive. A tracker is necessary to follow text as it moves. A text tracker could have several purposes in a video catalogue system, resembling:

- **Determination of Text Events:**

We would like to build an index of the text occurring in video for content-based retrieval purposes. The index would not include entries for individual frames, but instead for each text event that appears, persists for some time, and then disappears. That is, we would like to find the temporal location and extent of a text event, as well as the spatial location and extent in each frame. The tracker can be used to combine the localized text regions of individual frames into text events [18][23]. Figure 4.1 demonstrates the temporal redundancy and spatial similarity of text event using three examples. Figure 4.1 (a) illustrates four scene text events with horizontal motions, Figure 4.1(b) shows two caption text events with linear rotating motions, and Figure 4.1(c) illustrates a caption text event changing size with time.
Figure 4.1 Examples of caption text behavior over time. (a) Stationary text (b) Rotating rigid text exhibiting simple, linear motion (c) Text changing size with time.

- **Verification of Text Localization**: Since it is assumed that text persists for multiple frames, a region localized as text in one frame but not in neighboring frames indicates that it is a false alarm and should be discarded. Assuming all text is stationary, a candidate region could be discarded if no region exists at the same location in the neighboring frames. But this would fail for moving text. A text tracker is required to verify that motion in the localization output which is consistent with motion in the video [15][28].
4.2 Review of Preceding Work

While there has been a significant amount of work on the extraction of text in images and video frames, very little text detection work is found in the literature that considers the temporal nature of video. This section surveys the few approaches that do include temporal analysis.

Shim et al [1] uses a simple inter-frame analysis technique to reduce false alarms. Individual frames are first processed by finding regions with homogeneous strength, forming positive and negative images by double thresholding, and applying heuristics to remove non-text regions. Then, the candidate text regions in groups of five adjacent frames are considered. Text is assumed to be stationary. A candidate text region is discarded if regions of similar position, intensity, and shape do not appear in the other four frames. Note that this approach would incorrectly discard moving text regions.

Lienhart [2] takes a similar approach, but allows text motion. Individual frames are segmented using properties of local color histograms and choosing text candidate regions using heuristics. Temporal analysis is used to refine detection results. For each potential text region detected in a frame, the text candidate regions in the next frame are searched for one of identical size, color, and shape. If such an area is not found, the region is discarded as non-text. This approach assumes text remains rigid. It also requires that text detection is applied to each frame, so it is not applicable to the operator-assisted.

Li and Doermann [3] describe a simple algorithm for tracking rigid, moving text in video. A simple pixel-level template matching scheme is used. It is
assumed that the text is moving with constant velocity. A record is kept of this
velocity. Given the known location of a text region in a frame, its location in
the next frame is predicted using this velocity. A simple least-squared-error
search is performed around a neighborhood of the predicted location to find the
precise location [22]. Note that the pattern matching is performed on both text
pixels and background pixels alike. This can be problematic when text occurs
on complex backgrounds, or when text moves over backgrounds of different
gray level intensity. This approach also fails for text exhibiting a non-linear
velocity.
Qian et al. [5] present a text tracking approach in compressed video. For rolling
text tracking, the horizontal and vertical texture intensity projection profiles
are matched between two frames to identify text objects. For static text line
tracking, the starting and ending frames of text lines are determined by
computing Mean Absolute Difference (MAD) that represents the difference
between the text block regions in the direct current images of two consecutive
frames.
Jiang et al. [6] propose a Line against Line Matching (LALM) to track text
objects based on MAD. Given two horizontal bounded text objects in the
current frame and next frame, LALM is measured by MAD between the
horizontal midline of the text area in current frame and all horizontal lines of
the text area in next frame. If the minimum MAD is smaller than a pre-defined
threshold, the two text objects are labeled as an identical text object.
Antani et al. [6] and Gargi et al. [7] utilize motion vectors in an MPEG-1 bit stream in the compressed domain for text tracking, which is based on the methods of Nakajama et al. [9] and Pilu [10]. This method is implemented on the P and I frames in MPEG-1 video streams. Some preprocessing, such as checking any spatial-inconsistency and checking the number of significant edges in the motion vector, is first performed. The original bounding box is then moved by the sum of the motion vectors of all the macroblocks that correspond to the current bounding box. The matching results are refined using a correlation operation over a small neighborhood of the predicted text box region.

Li et al. [11] presented a text tracking approach that is suitable for several circumstances, including scrolling, captions, text printed on an athlete’s jersey, etc. They used the sum of the squared difference for a pure translational motion model, based on multi-resolution matching, to reduce the computational complexity. For more complex motions, text contours are used to stabilize the tracking process. Edge maps are generated using the Canny operator for a slightly larger text block. After a horizontal smearing process to group the text blocks, the new text position is extracted [25]. However, since a pure translational model is used, this method is not appropriate to handle scale, rotation and perspective variations.

Text tracking is performed to reduce the processing time for text localization and to maintain the integrity of position across adjacent frames. Although the precise location of text in an image can be indicated by bounding boxes, the text still needs to be segmented from the background to facilitate its recognition.
This means that the extracted text image has to be converted to a binary image and enhanced before it is fed into an OCR engine [26].

4.3 Text Inflowing or Leaving the Video

Text in video sometimes scrolls on or off the screen, such that in some frames only a portion of the text event is visible. I include special cases in the algorithm for handling this type of motion. Text exiting the frame is the easier case. The motion determination steps discussed above are applied only on the portion of the text event that is visible. If the computed motion indicates that the text event is exiting the frame, the tracked text box is clipped at the video frame boundary. Text scrolling into the video is more difficult, because the spatial extent of the text event is not known. For example, in the operator-assisted indexing application described above, the human may mark the visible portion of a text event occurring on the edge of the frame. In subsequent frames, the tracker determines that the text event is moving towards the center of the frame. We would like the tracker to be able to automatically resize the tracking box as more text enters the frame [27]. This case is handled in the following way. The number of edge pixels occurring in the known text region is counted and used as a texture measure. When the tracker detects that the tracking box is moving from the edge of the frame towards the center, the number of edge pixels in the region near the edge is also counted. If the density of edges between the two regions is comparable, the tracking box is expanded to accommodate the incoming text [4][17].
References


