Detection of Text in Video

Dave Snyder

February 26, 2010

Abstract

The detection of artificial text in video has been approached in many different ways, but each method shares a common thread. Text must first be segmented, then classified, and finally passed on to another algorithm depending on the overall task. Each method has shown varying degrees of success, however with no common data set and no standard metric for measuring results, comparison is difficult. Some researchers make assumptions about the problem to make it simpler to solve, while others attempt perhaps overly complex methods of accounting for more possibilities. It seems that this task is not impossible to do, but doing it in a way that is robust and elegant does not appear to have been done.

1 Introduction

Text that is displayed in a video often provides a useful context regarding the content of the video, which can then be used for understanding, retrieval, and search. The nature of readable text may make it more easily segmented and correctly identifiable than other objects, perhaps even correctly identifiable as specific characters or words. Temporal information provided by video is also useful and can be exploited for this task, assuming text is present in more than one frame of video.

Two types of text may be present in a frame of video, such as that shown in Fig 1.1. Artificial text is regarded by most researchers of video text detection techniques as the most common. This type is superimposed on the video during the editing process and is usually used for captions or titles. Scene text, on the other hand, is text which is present during the filming of the video, such as sponsor banners placed around the rink at a hockey game. Some algorithms are able to pick out scene text to an extent, but the majority focus on artificial text. Unfortunately in the reviewed algorithms, testing was not done on artificial text and scene text independently so the extent to which any given algorithm performed on one task in comparison to the other is unknown quantitatively.

Video provides redundant information that can be exploited for enhanced segmentation, noise reduction, and motion tracking. Unfortunately it seems many of the papers which claim to extract text from video do not take full advantage of this extra temporal information.

While some authors present a method for use with video, the authors do not make use of temporal information, instead choosing to treat each video frame separately. This approach

Figure 1.1: Video clip of a hockey game taken from YouTube. Notice both scene text and artificial text are present. Over the course of the video both artificial text and scene text move in and out of the frame.

is still useful for video frames as well as still images, however, because it takes into account complex backgrounds and text which is not of non-uniform a gray level.

It is common to segment text using edge detection [1, 2] for simplicity and speed. Other more exotic methods have also been proposed which use multiscale wavelet features [3] or discrete cosine transform coefficients [4]. To avoid as many false positives as possible, classification of detected regions can also be performed. Ye et al. use only a support vector machine (SVM) [3], Chen et al. use a neural network and an SVM [1], while Lienhart and Wernicke use a neural network for their approach [2]. Each technique suggests different features to use for classification, some more intuitive than others. In contrast, Qian et al. and Lienhart and Effelsberg do not perform a classification to improve segmentation, instead focusing on approaches to segmentation which produce results that standard OCR software can use to perform the final text recognition task [5, 4]. Likewise, Crandall et al. also rely only on image processing information and do not perform any classification techniques [6].

Table 1.1 provides an overview of several techniques reviewed. While it is difficult to generalize an algorithm into a few words, the table provides a guide for the segmentation, features chosen, classifier used with the chosen features, post processing run after classification, and if the algorithm is designed specifically for motion video.

Segmentation refers to the means by which each frame of video was divided into regions of interest which are likely to contain text. Edge detection, wavelet coefficients, discrete cosine transform coefficients, projection profiles, and other information is often combined with morphological image processing and some type of heuristics for this task. Morphological processing allows for the growing or shrinking of regions, many times with the goal of closing selected pixels into a solid shape, expanding the selected region to include a buffer zone, to simply to quantify connected components.

Features used are specific to each paper and include gradient maps, DCT coefficients,

Table 1.1: Overview of reviewed algorithms

Ref	Segmentation	Features	Classifier	Post processing	Video
[1]	Edge, Morphological	Gradients, DCT Coefficients	SVM, MLP	OCR	No
[2]	Edge, Projection Profiles	Raw Data	MLP	Temporal Heuristics	Yes
[3]	Wavelet, Morphological, Projection Profiles	Wavelet, Crossing Count Histogram	SVM	Multiscale Merging	No
[4]	DCT, Contrast, Morphological, Projection Profiles	*Segmentation**	*LDC**	Text Tracking	Yes
[5]	Color, Contrast, Motion, Heuristics	*Segmentation**	*LDC**	OCR	Yes
[6]	DCT, Morphological, Heuristics	*Segmentation**	*LDC**	Text Tracking, Temporal Heuristics	Yes

wavelet coefficients, using the raw data itself, and using other customized features. Note that [4,5,6] did not explicitly use a feature set other than the binary result of segmentation. For their methods if a region of interest was segmented, it was classified as text. Also note that these papers did not explicitly train classifiers for the same reason, and the region was labeled text if it was segmented. Other authors used more conventional classifiers including neural networks and support vector machines to decide if a segmented region of interest was text or not based on the features provided.

In the post processing phase, one of three options were chosen for rejecting any false alarms that may have made it past classification. A third party OCR too was used, the redundancy of temporal information was exploited, or in the special case of [3], multiple scales produced by the wavelet transform were merged. Finally, for those algorithms which explicitly used the extra information provided by video, "yes" was recorded, and for those which only looked at individual frames "no" was recorded.

2 Background

The discrete cosine transform, the wavelet transform, and the MPEG encoding process are often leveraged for text detection in video. To better understand the popularity of these topics, brief background information is provided. This material is referenced from [7].

2.1 Discrete Cosine Transform

The discrete cosine transform (DCT) expresses a function as the sum of cosine functions of varying frequency. It can be thought of as the real part of the Fourier Transform, which uses sine to represent the imaginary part and cosine to represent the real part of a frequency decomposition.

The most common use of the DCT is in lossy JPEG compression. An image is transformed into cosine space and truncated at some threshold frequency. The inverse transform reproduced much of the original image, however since data has been removed perfect reconstruction is not possible.

Unlike the Fourier Transform, when computing the DCT only a finite size window is used almost exclusively. JPEG uses an 8×8 window size, however any $N \times N$ window is valid. The goal of using a window is to find regions in the image which are uniform and represent those with less data than regions contain more information. The DCT of a window with little variation contains many low values which can be set to zero and thrown away, while the DCT of a window with a significant amount of variation would not contain many low values and would likely be kept. Since edges are high frequency information, this concept can be extended to locating edges in a scene based on the value of the DCT coefficients.

Fig 2.1 illustrates the effect of throwing away higher and higher values after taking the DCT with an 8×8 window. At first there is very little difference, however as more values are removed each of the entire 8×8 region becomes represented by their largest value, introducing strong blocking artifacts.

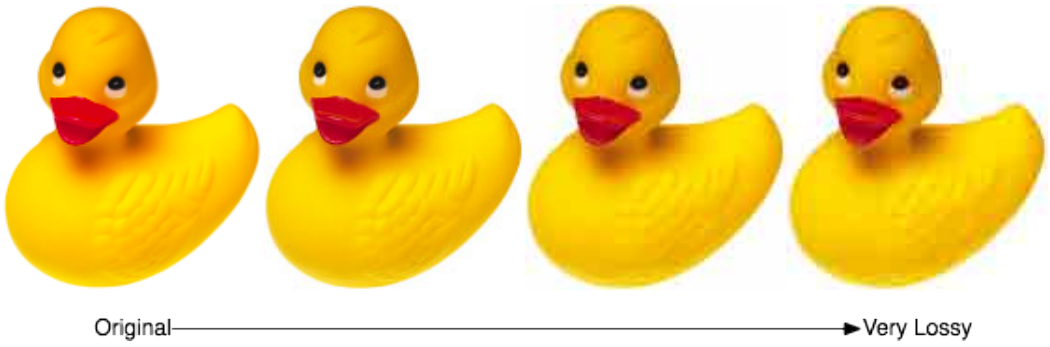

Figure 2.1: Visibility of the underlying 8×8 DCT increases as more coefficients are removed from each block.

2.2 Discrete Wavelet Transform

Similar to the Fourier transform which leverages sinusoids as basis functions, the wavelet transform makes use of an alternate set of basis functions which may be more suited for a given problem. Both the continuous and the discrete wavelet transform (DWT) exist, however only the discrete transform will be discussed here due to its direct application to digital image processing.

The basis functions used are not specific to the transform, but rather are selected based on the problem. Originally the Haar basis functions were used, as they are the simplest and oldest, however currently it is more common to use the Daubechies basis functions, the CDF 5/3, or the CDF 9/7 basis functions. As an example, the JPEG 200 image compression standard uses the CDF 5/3 based wavelet for lossless compression and the CDF 9/7 based wavelet for lossy compression. It is not common to use the Daubechies basis functions for image processing as they are not symmetric, but the CDF functions are. Regardless of the specific set of basis functions used, two properties must be true. First, there are only two basis functions for any given set, and second, one acts as a low pass filter while the other acts as a high pass filter. The Haar and Daubechies basis functions are orthogonal, however this is not necessarily a requirement.

Given an input function and a set of basis functions, the discrete wavelet transform can be computed. The basis functions are applied as high and low pass filters to the original signal resulting in twice as much data out as data in. These outputs are downsampled by two, resulting in the same amount of data in as data out. The output are the coefficients which map the input function into wavelet space. This process is shown in Fig 2.2.

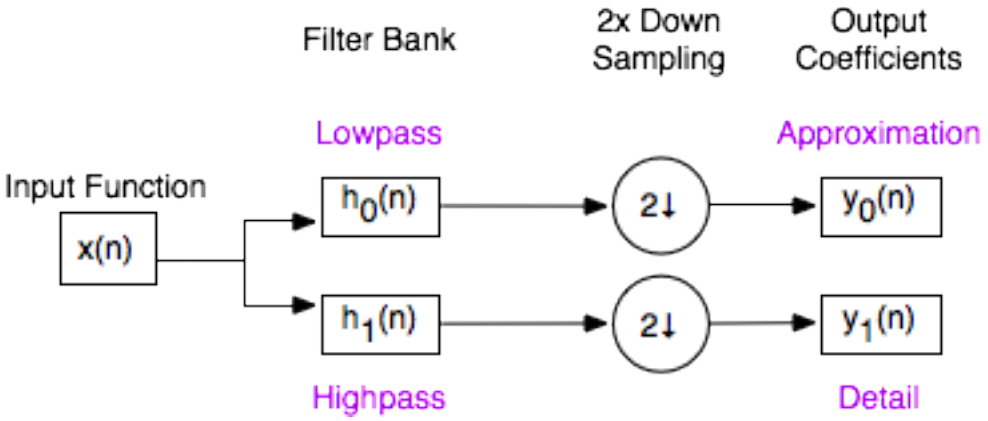

Figure 2.2: Block Diagram of the DWT

While a single filtering pass is interesting and useful, wavelets become far more useful through iteration. Given a lowpass, downsampled image, the same highpass and lowpass filters can again be applied and the results downsampled by two. Since in the practical case, the input data is finite, this iteration can occur until only a single element of data remains or until a threshold is reached. Thus by recursively applying the discrete wavelet transform, we produce multiresolution imagery, or imagery which contains several different scales. This is useful when incorporating scale invariance into an algorithm, or when an application may require several different scales of data. The recursive discrete wavelet transform is diagramed in Fig. 2.3.

Since images are discrete, finite, two-dimensional functions, extending the general discrete wavelet transform to work with images is trivial. Since the multidimensional wavelet transform is separable, the two-dimensional case is simply the product of the wavelet transform taken in the horizontal and vertical directions. For convention, the upper left most

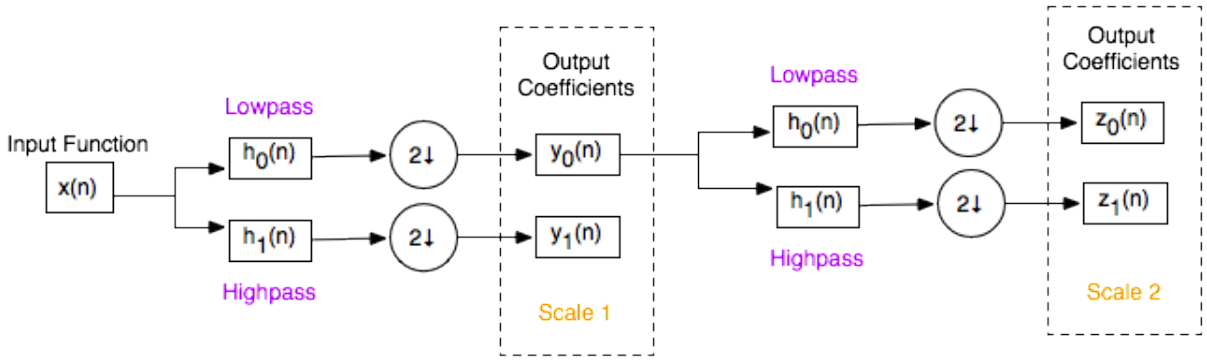

Figure 2.3: Block Diagram of two scales of the DWT

quadrant of an image is where recursion is visualized. The wavelet transform of an image is shown in Fig. 2.4[1].

Regarding computational complexity, the discrete wavelet transform can be implemented as the fast wavelet transform, which has linear complexity. Regarding the inverse transform, assuming the coefficients have not been altered, the original function can be perfectly reconstructed.

3 Segmentation

Given a sequence of video frames, potential regions of interest must be segmented. Methods of segmentation include edge detection, color and contrast segmentation, thresholding discrete cosine transform coefficients, and thresholding wavelet transform coefficients.

3.1 Edge Detection

For their technique, Chen et al. extract regions of interest from each frame of video based on vertical and horizontal edges produced by a Canny edge filter. Dilation of the extracted edges is performed using empirically derived rectangle kernels of size 1×5 for the vertical direction and 6×3 for the horizontal direction. When comparing the regions selected to ground truth text regions, assuming a region is correctly located if there is an 80% overlap between the two, this method extracted 9,369 text lines but produced 7,537 false alarms from the test data. This results in a precision of 55.4%, which is quite poor. Nevertheless, regions of interest were selected and can be fed into a classifier for further refinement.

Chen et al. further refine their text selection by using a baseline detection algorithm previously developed by Chen, Bourlard, and Thiran [8]. Additionally, final selections are constrained to be between 75 and 9,000 pixels in area, with a horizontal-vertical aspect ratio of 1.2 and a height between 8 and 35 pixels. No reason is given for these criteria, however the authors do note that text can vary greatly in size and a scaled image pyramid, or simply

[1]Source:
`http://en.wikipedia.org/wiki/File:Jpeg2000_2-level_wavelet_transform-lichtenstein.png`

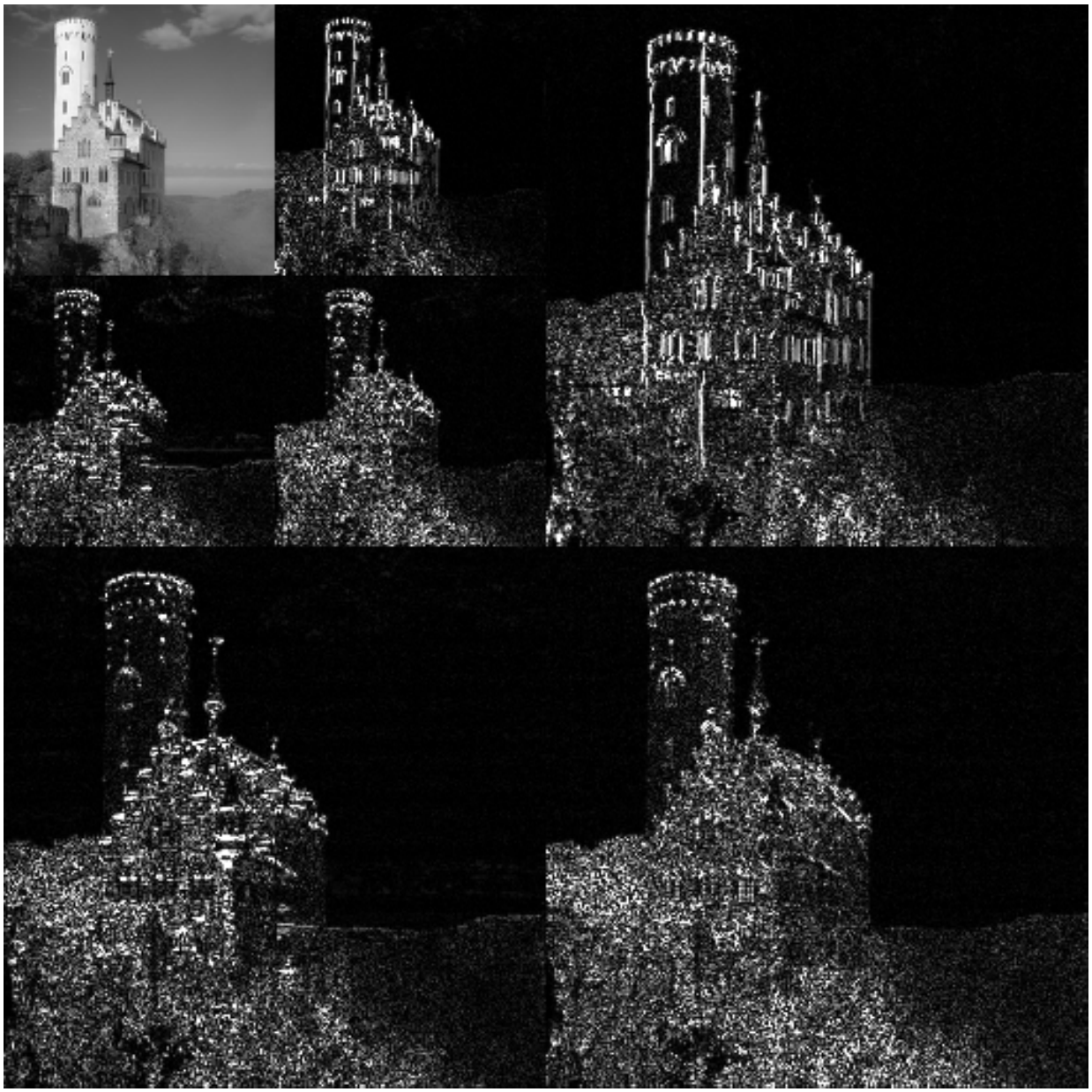

Figure 2.4: Two level 2D DWT[1].

a series of images down sampled by a factor of two, could allow this algorithm to work on larger text regions.

3.2 Color and Contrast Segmentation

Lienhart and Wernicke use region growing to create bounding boxes for the selected text regions. Profile projections in both horizontal and vertical directions were used in an iterative process to further refine bounding boxes by merging overlapping regions or deleting some selections which were under threshold intensity values. The number of iterations as well as the thresholds used for bounding boxes were determined empirically. Color information was used to group text and remove as many background pixels as possible.

To segment individual characters and remove the background, text selections are first scaled to have a height of 100 pixels by using a bilinear interpolation. Next pixels of the same color as those just past the bounding box are removed from within the bounding box, as they are assumed to be background. For video, temporal information is used to assist in the background removal process as it is assumed the text changes very little from frame to frame, while the background can change significantly. The authors make no attempt to validate this assumption, however.

3.3 Discrete Cosine Transform Coefficients

Crandall et al. perform connected component analysis is on detected text blocks followed by a procedure similar to the morphological dilation. Blocks which are larger than 8 pixels in width and length are kept as localized regions of text.

Crandall et al. perform text detection by analyzing texture features provided by the Discrete Cosine Transform (DCT). An 8×8 block-wise DCT is performed on each video frame, producing a set of DCT coefficients. A specific, empirically determined subset of these coefficients are used to characterize "text energy" in the horizontal and vertical directions. A dynamically chosen threshold, driven by an empirically derived formula based on the contrast of the entire video, is used to determine whether or not each block contains text. Since an 8×8 block size is used, subsampling is performed to detect text of different sizes. The author cites [9] for describing an efficient way of subsampling in the DCT domain.

3.4 Wavelets for Segmentation

Ye et al. compute the wavelet transform of each video frame using the Daubichie basis functions. Wavelet coefficients are used with an energy function as a feature to locate potential text regions in the coarse detection step of this algorithm. Text is assumed to produce large wavelet coefficients and therefore have a higher energy than non-text regions. A dynamic threshold based on the energy histogram of each image is used to select which energy value is sufficient for text detection. The morphological close operation is used to combine nearby pixels of text into clusters. Projected profiles are then used to separate individual lines. Finally, any clusters smaller than eight pixels in height, or with a width-to-height ratio of less than one, are discarded.

4 Features

4.1 Wavelet Features

For Ye et al. Fine detection is accomplished using four collections of wavelet features and a support vector machine classifier. Features used include wavelet moment; histogram; co-occurrence features including energy, entropy, inertia, local homogeneity, and correlation; and crossing count histogram. A total of 225 features from each line of text are generated using this information. Since a large number of features may hinder the overall performance of a classifier, a forward search algorithm was used to reduce the number of features used to just 32. Of the original features, wavelet moment and the crossing count histogram appear to be the most important based on the results of the forward search.

Large text produces a relatively low frequency signal, while small text produces a relatively high frequency signal. In the wavelet domain, low frequency information produces a stronger response in a deep scale, after only a few iterations of the wavelet transform, while high frequency information produces a stronger response in a shallow scale, after many iteration. These two responses will be similar, thus multiscale text detection is possible by choosing different scales from which to extract features.

4.2 Other Features

Other researchers who used the DCT coefficients for classification or the raw image intensity values did no further processing of this information before training a classifier. Half of the works reviewed did not explicitly use a classifier either, instead focusing on the segmentation task and assuming that after a good segmentation text will be present in regions of interest selected.

5 Classification of Regions

5.1 Support Vector Machine

Chen et al. chose to verify text regions by making use of an SVM trained on four feature types. Grayscale spatial derivatives, a distance map, constant gradient variance, and discrete cosine transform coefficients are the features used. Constant gradient variance was developed by the authors but is fairly straightforward. Given a nine by nine region of pixels, a local mean and variance are computed. This local information is then compared with a global gradient variance for the whole image, and the gradient magnitude for a given pixel. In this way, changes in contrast can be exploited as a feature.

An MLP was also trained, but the authors found an SVM performed better and selected it as the classifier of choice for this algorithm. A confidence threshold was set for determining what was text and what was not based upon the four features used. On the training data, the SVM using the constant gradient feature had an error rate of 1.07%. For the grayscale spatial derivatives the error was 3.99%, for the distance map, 2.56%, and for the discrete cosine transform, 2.00%. Using the same testing data previously used by Chen et al. in section 3.1, 7255 of the 7537 false alarms were removed by classification, and 23 true text

lines were removed. Overall the authors report that this produces a 97% precision rate and a 0.24% rejection rate.

Ye et al. also trained an SVM, using their selected wavelet features with boosting, to improve performance. Specific details about the kernel used are not provided by the authors.

5.2 Neural Network

Lienhart and Wernicke employ gradient maps as features which are fed into a neural network. Image gradients, or gradient maps are the two-dimensional derivative of the image. Details of the network architecture are included in the paper. Training was conducted on a set of data containing 6,000 text patterns and 5,000 non-text patterns. A validation set was used to further tune the network before testing. Little other detail was provided about the parameters used for this network.

6 Post Processing

6.1 Optical Character Recognition

Before handing results off to a stock algorithm for OCR, binarization must first take place.

The binarization performed by Crandall et al. is composed of three steps: preprocessing, logical level thresholding, and character candidate filtering. Preprocessing rotates any non-horizontal localized text regions into the horizontal orientation, and doubles the resolution via a linear interpolation. Contrast is enhanced by histogram equalization, and ten frames are used with a rigid text tracker for temporal averaging. Logical level thresholding is applied to the image after a conversion to L*a*b* color space to the luminance channel, using an empirically determined threshold value. This color space consists of light-dark, red-green, and yellow-blue opponent channels and seeks to be perceptually accurate to the average human observer. Standards have been published which allow for the conversion from the more familiar RGB (red, green, blue) color space into L*a*b* space. This step is applied to the original image, and its inverse, to account for the text being lighter or darker than the background. Once again, connected component analysis is performed and any characters that do not meet certain criteria set by the authors are discarded. A voting strategy is used to determine if the original or inverse image should be kept. Taken into account for each character are height similarity, width similarity, horizontal alignment, aspect ratio, clean spacing, and periodicity of vertical projection.

Lienhart and Wernicke perform binarization by using a simple threshold which is halfway between the average background and foreground color.

6.2 Temporal Heuristics

For video, Lienhart and Wernicke exploit temporal redundancy to help remove false alarms, reduce noise, and improve text segmentation accuracy. In order to reduce complexity, video was sampled every second, and if text was found each frame available from one second before and one second after was then analyzed for text content. Profile projections were used as

signatures for text, allowing text to be tracked from frame to frame assuming very little variation in the text from frame to frame. Due to noise, compression artifacts, and other issues it is difficult to track text from frame to frame perfectly. Thus a dropout threshold is used to allow tracking to continue even if a few frames are skipped. Text which occurs for less than a second or is present in less than 25% of the frames in a sequence is discarded.

6.3 Text Tracking

Crandall et al assume text tracking is rigid; that is, text moves with a constant velocity and in a linear path. Motion vectors provided by the MPEG compression process are used to predict text motion [10, 11]. By using only macroblocks with more than four edge pixels, as computed by the Sobel edge detector, motion vectors can be more useful over short video sequences. The authors note that MPEG motion vectors are computationally efficient as the work has already been done, however they are often too noisy to use directly as a tracker, depending on the quality of MPEG encoding used. A comparative technique is used to augment motion vectors for the text tracking task. Frame-by-frame comparisons of connected components are used with a threshold to determine if a localized text region exists in multiple frames.

7 Published Results

There is no standard method for evaluating the performance of a object detection algorithm, however there are similarities. For example, all researchers work with ground truth data where the correct answer is known. This data poses another problem, however, because it was generated by a human and it is likely to contain errors. If the task of a graduate student was to draw a bounding box around the text instances in several thousand frames of video, it would not be unreasonable to expect variation in where the box is placed around the text and other similar errors. Likewise the algorithm may have successfully found the text but the box it draws is too large or shifted or skewed in some way compared with the ground truth. To account for these issues, a threshold for percent overlap can be set to decided when two regions are "close enough" to be called the same. What this value is depends on a researcher's personal preference. Alternately, boxes could only be counted when they exactly overlap, as done by [6], however it seems that exact overlap will push scores lower than they need to be.

Other issues in comparing results include a lack of common data, and different sizes of the sets of data used. Some groups chose thousands of frames of video over which to score their techniques, while others chose only a small number of frames. To properly compare these methods, each would need to run on the same data using the same percent overlap of bounding box when reporting correct results.

Three metrics are commonly used to report results: recall, precision, and false alarm rate. Recall, shown in Eq. 7.1, is simply the number of correct detects over the total number of targets in the ground truth. A perfect score for Recall is 100%. Precision, shown in Eq. 7.2, is the number of correct detects over the total number of detects reported by the algorithm. Ideally this would also be 100%. Finally false alarm rate is the number of false alarms over

the number of correct detections, shown in Eq. 7.3. Ideally this would be zero.

$$\text{Recall} = \frac{\text{correct detects}}{\text{correct detects + missed detects}} \tag{7.1}$$

$$\text{Precision} = \frac{\text{correct detects}}{\text{correct detects + false alarms}} \tag{7.2}$$

Crandall et al. tested their algorithm on two datasets of MPEG videos totaling over 11,000 frames at a 320×240 pixel resolution. Precision and recall were computed.

On the first dataset, the proposed algorithm only achieved a recall of 46% and a precision of 48% for detection and localization of caption text. On the second dataset, for the same task, a precision of 74% and a recall of 74% were achieved. The authors note these low numbers may be due to the requirement for data to exactly match the ground truth, which may not be necessary.

Compared with other techniques, I find the use of MPEG motion vectors interesting, given that the majority of videos encoded are done so using the MPEG standard. Since no slack was given in the selection of text compared with ground truth, the results may be better than they seem for this algorithm. Many other authors include a buffer region in their ground truth regions. Additionally, this algorithm does not make use of pattern classification techniques, which could also significantly improve results given the demonstrated usefulness of the discrete cosine transform features selected. This method was implemented by Kasturi et al. [12] and is available for download.

Ye et al. performed experimental testing on a notably small dataset of only 221 images, each of size 400×328 pixels. Ground truth was marked by hand, and a detection was marked successful if more than 95% of the ground truth box was contained in more than 75% of the detected rectangle. Based on this definition of a correct detection, recall (Equation 7.1) and false alarm rate (Equation 7.3) were computed.

$$\text{False Alarm Rate} = \frac{\text{False Alarms}}{\text{Correct Detections}} \tag{7.3}$$

Based on these metrics, the authors report a recall rate of 94.2% and a false alarm rate of 2.4% for their testing dataset.

This method seems interesting and powerful, however more extensive testing should be performed, perhaps with more restrictive conditions for correct detection. Temporal information in video is not taken into account either, which could be potentially beneficial.

Lienhart and Wernicke use a relatively small testing set of only 23 videos. Videos ranged in size from 352×240 to 1920×1280. If the detected bounding box overlapped the ground truth by 80% a detection was labeled correct. For individual images, a 69.5% hit rate, 76.5% false hit rate, and a 30.5% miss rate was found. For video, significant improvements were seen with a 94.7% hit rate, 18.0% false hit rate, and a 5.3% miss rate. Given the fairly liberal 80% overlapping region allowed, these numbers are perhaps not as good as the author makes them out to be. In particular, the large false alarm rate should be addressed.

8 Conclusion

Several methods for detecting text in video were presented. Comparison between the results of the methods is difficult due to the lack of a standard data set and differences in computing metrics, however qualitatively the algorithms can be compared. Each took a different approach to the problem and showed positive results. Methods which focused heavily on segmentation may benefit by using a more sophisticated classification technique. Likewise methods which relied heavily on the classifier to handle poorly segmented input may benefit from enhanced segmentation. Some methods made use of the temporal redundancy provided by video, but not all. Perhaps those methods which did not explicitly use this information would benefit by incorporating it.

From my own background, I favor the idea of using high frequency information to perform the initial segmentation via edge detection or wavelets. Heuristics about the size and proportion of text, color, contrast, or texture based models do not seem to be as reliable. Depending on the compression scheme of the video, it may also make sense to use the discrete cosine transform, as the calculation has already been done. Regarding features and classifiers to use, I feel I should do more research into newer papers to see what trends are currently emerging in the field. One such paper, [13], appears to be using similar techniques to those reviewed, including gradient maps and pixel intensity to train an SVM.

As work continues in this area and in the more general area of image understanding, it will most certainly be interesting and useful moving forward.

References

[1] D. Chen, J. Odobez, and H. Bourlard, "Text detection and recognition in images and video frames," *Pattern Recognition* **37**(3), pp. 595–608, 2004.

[2] R. Lienhart and A. Wernicke, "Localizing and segmenting text in images and videos," *IEEE Transactions on circuits and systems for video technology* **12**(4), pp. 256–268, 2002.

[3] Q. Ye, Q. Huang, W. Gao, and D. Zhao, "Fast and robust text detection in images and video frames," *Image and Vision Computing* **23**(6), pp. 565–576, 2005.

[4] X. Qian, G. Liu, H. Wang, and R. Su, "Text detection, localization, and tracking in compressed video," *Signal Processing: Image Communication* **22**(9), pp. 752–768, 2007.

[5] R. Lienhart and W. Effelsberg, "Automatic text segmentation and text recognition for video indexing," *Multimedia systems* **8**(1), pp. 69–81, 2000.

[6] D. Crandall, S. Antani, and R. Kasturi, "Extraction of special effects caption text events from digital video," *International Journal on Document Analysis and Recognition* **5**(2), pp. 138–157, 2003.

[7] R. C. Gonzalez and R. E. Woods, "Digital image processing," Pearson Education, Inc., (Upper Saddle River, NJ), 2008.

[8] D. Chen, H. Bourlard, and J. Thiran, "Text identification in complex background using SVM," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **2**, IEEE Computer Society, 2001.

[9] R. Dugad and N. Ahuja, "A fast scheme for altering resolution in the compressed domain," in *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on.*, **1**, 1999.

[10] J. Mitchell, C. Fogg, W. Pennebaker, and D. LeGall, *MPEG video compression standard*, Kluwer Academic Publishers, 1996.

[11] Y. Nakajima, A. Yoneyama, H. Yanagihara, and M. Sugano, "Moving-object detection from MPEG coded data," in *Proceedings of SPIE*, **3309**, p. 988, 1998.

[12] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**(2), pp. 319–336, 2009.

[13] C. Jung, Q. Liu, and J. Kim, "Accurate text localization in images based on svm output scores," *Image and Vision Computing* **27**(9), pp. 1295 – 1301, 2009.