# EXTRACTING TEXT FROM VIDEO

Jayshree Ghorpade[1], Raviraj Palvankar[2], Ajinkya Patankar[3] and Snehal Rathi [4]

[1]Department of Computer Engineering, MIT COE, Pune, India
jayshree.aj@gmail.com
[2]Department of Computer Engineering, MIT COE, Pune, India
ravirajp7@gmail.com
[3]Department of Computer Engineering, MIT COE, Pune, India
ajinkya1412@gmail.com
[4]Department of Computer Engineering, MIT COE, Pune, India
snehalrathi_done@gmail.com

***ABSTRACT***

*The text data present in images and video contain certain useful information for automatic annotation, indexing, and structuring of images. However variations of the text due to differences in text style, font, size, orientation, alignment as well as low image contrast and complex background make the problem of automatic text extraction extremely difficult and challenging job. A large number of techniques have been proposed to address this problem and the purpose of this paper is to design algorithms for each phase of extracting text from a video using java libraries and classes. Here first we frame the input video into stream of images using the Java Media Framework (JMF) with the input being a real time or a video from the database. Then we apply pre processing algorithms to convert the image to gray scale and remove the disturbances like superimposed lines over the text, discontinuity removal, and dot removal. Then we continue with the algorithms for localization, segmentation and recognition for which we use the neural network pattern matching technique. The performance of our approach is demonstrated by presenting experimental results for a set of static images.*

***KEYWORDS***

*JMF, Image Processing, Text Extraction, Text Localisation, Segmentation, Text in Video*

## 1. INTRODUCTION

Text appearing in images and videos is usually categorized into two main groups according to the source where it comes from: Artificial text (referred also as caption text or superimposed text) and Scene text (referred also as graphics text). Artificial text is laid over the image at a later stage (e.g. the name of someone during an interview, the name of the speaker, etc.). In contrast to artificial text, scene text is part of the scene (e.g. the name of a product during a commercial break, etc.) and is usually more difficult to extract due to different alignment, size, font type, light condition, complex background and distortion. Moreover, it is often affected by variations in scene and camera parameters, such as illumination, focus, motion, etc. The video contents are available from TV broadcasting, internet and surveillance cameras. These videos contain texts, including scrolling texts or caption text made by artificial overlaying after recording and scene text embedded in backgrounds. Text embedded in images contains large qualities of useful information. Since words have well-defined and unambiguous meanings, text extracted from video clips can provide meaningful keywords which can reflect the rough content of video. These keywords can be used for indexing and summarizing the content of video clip [1].

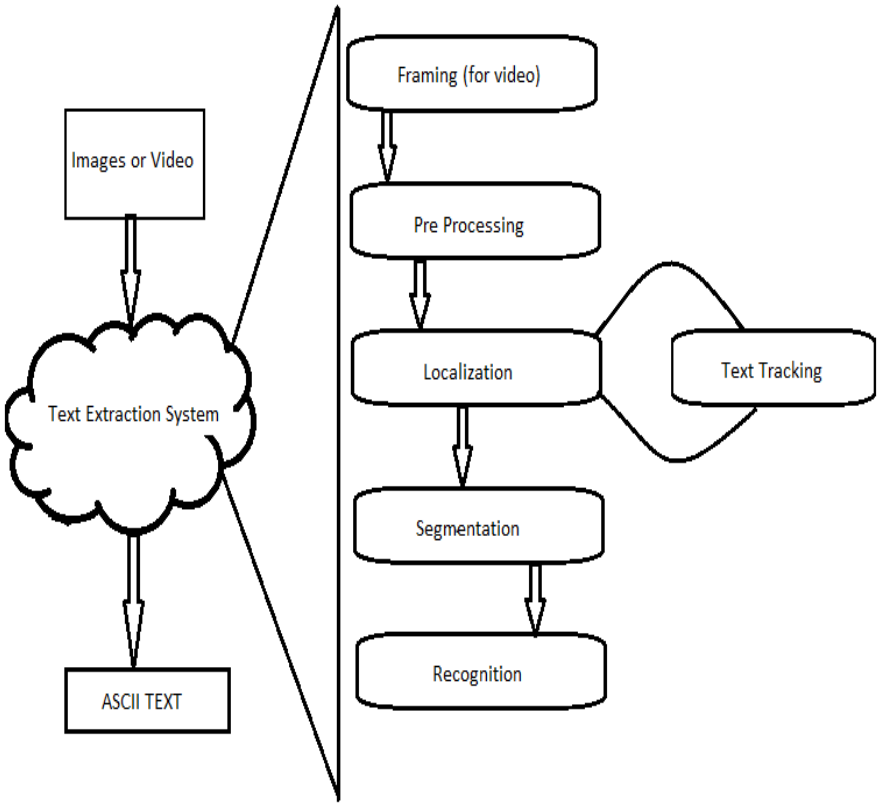


Figure 1: Our proposed model

The process of extracting text from video consists of various stages as seen in the figure 1. The system can take a real time video or a video from stored database as input. Each stage is explained by presenting experimental results for a set of static images [1], [2].

## 1.1. Framing of Video

We have used the Java Media Framework (JMF) to capture the media content and frame the video. JMF is a framework for handling streaming media in Java programs. JMF is an optional package of Java 2 standard platform. JMF provides a unified architecture and messaging protocol for managing the acquisition, processing and delivery of time-based media. JMF enables Java programs to:

- Present (playback) multimedia contents.
- Capture audio through microphone and video through camera.
- Do real-time streaming of media over the Internet.
- Process media (such as changing media format, adding special effects).
- Store media into a file.
- JMF provides a platform-neutral framework for handling multimedia.

The input to this stage was a video containing text. The video was then framed into images using JMF at the rate of 1 frame per second. This rate could be increased or decreased depending on the speed of the video i.e. on the basis of fps (frames per second). The images

were scaled to a resolution of *280x90* and were saved at the specified location on the hard disk drive.

*For example*: If the video was of 30 seconds, then 30 frames (images) of the video would be scaled to a size of 28x90 and saved which were then given as an input to the next stage [7].

## 1.2. Pre Processing

A scaled image was the input which was then converted into a gray scaled image. This image formed the *first stage of the pre-processing* part. This was carried out by considering the RGB color contents(R: 11%, G: 56%, B: 33%) of each pixel of the image and converting them to grayscale. The conversion of a colored image to a gray scaled image was done for easier recognition of the text appearing in the images as after gray scaling, the image was converted to a black and white image containing black text with a higher contrast on white background. As the input video was framed into images at the rate of 1 frame per second in the first stage, the input to this gray scaling stage was a scaled (280x90) colored image i.e. the frames of the video [3], [6].

The *second stage of pre-processing* is lines removal. A video can contain noise which is either a horizontal fluctuation (a horizontal line throughout the screen) or a vertical fluctuation (a vertical line throughout the screen). Thus, for successful recognition of the text appearing in the frames, there is a necessity that these horizontal and vertical fluctuations should be removed. This was carried out by clearing all pixels (changing pixel color from black to white) located on all lines which appeared horizontally and vertically across the screen because of the fluctuations which may have occurred in the video. This stage did not make any changes to the image if the video frame did not contain any horizontal and vertical fluctuations [6].

The *third stage of pre-processing* is discontinuities removals that were created in the second stage of pre-processing. As explained above, if the video contained any fluctuations, then those fluctuations were removed in the lines removal stage. If the horizontal and vertical fluctuations occurred exactly where the text was present, then it created discontinuities between the text appearing in the video frame which made the recognition of the text very difficult. This was carried out by scanning each pixel from top left to bottom right and taking into consideration each pixel and all its neighbouring pixels. If a pixel under consideration was white, and all the neighbouring pixels were black, then that corresponding pixel was set as black because all the black neighbouring pixels indicated that the pixel under consideration was cleared at the lines removal stage because of the fluctuations [3].

The *final output of pre-processing* stage is wherein the remaining disturbances like noise are eliminated. This was carried out again by scanning each pixel from top left to bottom right and taking into consideration each pixel and all its neighbouring pixels. If a pixel under consideration was black, and all the neighbouring pixels were white, then that corresponding pixel was set as black because all the black neighbouring pixels indicated that the pixel under consideration was some unwanted dot (noise) [2].

## 1.3. Detection and Localization

In the text detection stage, since there was no prior information on whether or not the input image contains any text, the existence or non existence of text in the image must be determined. However, in the case of video, the number of frames containing text is much smaller than the number of frames without text. The text detection stage seeks to detect the presence of text in a given image. Selected a frame containing text from shots elected by video framing, very low threshold values were needed for scene change detection because the portion occupied by a text region relative to the whole image was usually small. This approach is very sensitive to scene

change detection. This can be a simple and efficient solution for video indexing applications that only need key words from video clips, rather than the entire text. The localization stage included localizing the text in the image after detection. In other words, the text present in the frame was tracked by identifying boxes or regions of similar pixel intensity values and returning them to the next stage for further processing. This stage used Region Based Methods for text localization. Region based methods use the properties of the color or gray scale in a text region or their differences with the corresponding properties of the background. [2], [5].

### 1.4. Segmentation

After the text was localized, the text segmentation step deals with the separation of the text pixels from the background pixels. The output of this step is a binary image where black text characters appear on a white background. This stage included extraction of actual text regions by dividing pixels with similar properties into contours or segments and discarding the redundant portions of frame [2].

### 1.5. Recognition

This stage included actual recognition of extracted characters by combining various features extracted in previous stages to give actual text with the help of a supervised neural network. In this stage, the output of the segmentation stage is considered and the characters contained in the image were compared with the pre-defined neural network training set and depending on the value of the character appearing in the image, the character representing the closest training set value was displayed as recognised character [2], [4].

## 2. LITERATURE SURVEY

The previous work which has been done on this concept has been basically done on images of documents that have been scanned which are typically binary images or they can be easily converted to binary images using simple binarization techniques i.e. converting the color image into a grayscale image and then thresholding the grayscale image [1], [6].

These techniques used for text detection were developed for binary images only and thus these techniques cannot be directly applied to color image or a video. The main problem is that colored images and videos have rich color content, complex colored backgrounds which are high in resolution and contains a lot of noise which makes it difficult to extract the text from them using the techniques which were used earlier for a scanned image document which is in binary form. The binarization is a conventional method used for extracting text from a image. The color of the text in an image is lighter or darker as compared to its background and this is used for text detection in binarization and so the text region in the image can be segmented by thresholding. These methods are based on the assumption that text pixels have a different color than background pixels, thus threshold is used to separate the text from the background. But the noise and the disturbances contained in an image can also share the same color as of the text and thus is becomes difficult to detect the region containing text [3], [4], [7].

Also, here are different language dependent characteristics such as: stroke density; font size; aspect ratio; and stroke statistics affect the results of text segmentation and binarization methods.

The previous text extraction algorithm for extracting text from a scanned image document involved 3 basic steps as follows:

- ***Step1: Text Detection-*** The procedure of detection determines the boundary of candidate text regions in a frame.

- ***Step 2: Text Extraction-*** Text extraction segments these regions and generates binary images for recognition.

- ***Step 3: Text Recognition-*** Recognition was conducted with commercial OCR engines. However, the traditional OCR technique can only handle binary text image with simple background, while the images colored images which have complex background will contain a lot of noise and disturbances. These disturbances will therefore influence recognition accuracy heavily.

The analysis of the above mentioned technique is as follows:

## 2.1. Text Detection:

It was unpredictable for the text color to be darker or lighter than its background. Therefore, simple binarization was not adaptive enough in text extraction.

## 2.2. Text Extraction:

There often exist many disturbances from background in a text region. They share similar intensity with the text and consequently the binary image of the text region is unfit for recognition directly.

## 2.3. Text Recognition:

The result of recognition was a ratio between the number of correctly extracted characters and that of total characters and evaluates what percentage of a character were extracted correctly from its background. For each extraction result of characters, if it did not miss the main strokes, it was taken as a correct character. The extraction results were then sent to OCR engine directly. A commercial OCR engine was utilized for recognition.

Another method was proposed for text extraction from a colored image with complex background in which the main idea was to first identify potential text line segments from horizontal scan lines. Text line segments were then expanded or merged with text line segments from adjacent scan lines to form text blocks. False text blocks were filtered based on their geometrical properties. The boundaries of the text blocks were then adjusted so that text pixels lying outside the initial text region were included. Text pixels within text blocks were then detected by using bi-color clustering and connected components analysis. Finally, non-text artifacts within a text region were filtered based on heuristics [3], [5].

This method consisted of basically 6 steps:

- ***Step1: Identify Potential Text Line Segments***

  The grayscale luminance value of each pixel on the scanline was first computed from the pixel's R, G, and B color values. Each horizontal scanline was divided into several segments based on the value of the maximum gradient difference computed at each pixel location. The maximum gradient difference was computed as follows: first the horizontal luminance gradient G!X of the scanline was computed by using the mask [-I, I], then the difference between the maximum and minimum gradient values was computed within a local window of size 1 x 21; we call this the maximum gradient difference. Usually, text areas have larger maximum gradient difference value than background areas. For each scanline segment with large maximum gradient difference values, the following statistics were computed from the horizontal gradient waveform: first the number of crossings was counted at a horizontal gradient level equal to a negative threshold value (negativeEdgeCount), then the number of crossings was counted at a horizontal gradient level equals to a positive threshold (positiveEdgeCount), several consecutive negative crossings (without positive crossings

in between) were counted as one crossing; similarly for positive crossings. This was because each text stroke required a pair of positive and negative crossings (in either order) as end points. If negativeEdgeCount or positiveEdgeCount was less than some threshold, the scanline segment was discarded. The mean and variance of the horizontal distance between the crossings were then computed. For scanline segments containing text, the mean and variance were considered within a certain range. If not, the scanline segment was discarded. All remaining line segments after the filtering process were potential line segments for text [3].

- ***Step 2: Potential Text Blocks Detection***

  In the second step, every potential text line segment was expanded or merged with potential text line segments from adjacent scanlines to form a potential text block. For each potential text line segment, the mean and variance of its grayscale values were computed from the luminance image. This step of the algorithm was run in two passes. In the first pass, the line segment (with the same endpoints) immediately below a potential text line segment detected in step 1 was considered. If the mean and variance of the luminance values of the current line segment was close to those of the potential text line segment above it, it was merged with the line segment above it to form a potential text block. This process was repeated for the line segment immediately below the newly formed text block. In the second pass, the process was repeated in a bottom up fashion. The reason for having both top-down and bottom-up passes was to ensure that we do not miss any text blocks [3].

- ***Step 3: Text Blocks Filtering***

  The potential text blocks were then subject to a filtering process based on their area, and their height and width ratio. If the computed values fell outside a pre-specified range, they were discarded [3].

- ***Step 4: Boundary Adjustment***

  For each text block, it was necessary to adjust the boundaries to include text pixels that were lying outside the boundaries. In doing so, first the average maximum gradient difference of the text block was computed. Then the boundaries of the text block were adjusted to include outside adjacent pixels that had maximum gradient difference that were close to that of the computed average [3].

- ***Step 5: Bi-Color Clustering***

  The bi-color clustering was then applied to each detected text block. The color histogram of the pixels within the text block was used to guide the selection of initial foreground and background colors. Two peak values were picked that were of a certain minimum distance apart in the color space as initial foreground and background colors [3].

- ***Step 6: Filtering***

  In the filtering step, false text blocks and noise artifacts within a text block were eliminated. First, the connected components of text pixels within a text block were determined. Then the following filtering procedures were performed:

  - If text-region-height > some threshold T1, and area of any connected component > text-region-areal2, the entire text block was discarded.
  - For every connected component, if its area < text-region-heighU2, then the connected component was regarded as noise and discarded.

- If a connected component touched one of the boundaries of the text block, and if its size was larger than a certain threshold, it was discarded.

The output from this approach consisted of connected components of text character pixels [3].

## 3. PLATFORM/TECHNOLOGY

### 3.1. Operating System

Windows XP/Vista/7

### 3.2. Language

Java J2SE:

Java Platform, Standard Edition or Java SE is a widely used platform for programming in the Java language. It is the Java Platform used to deploy portable applications for general use. In practical terms, Java SE consists of a virtual machine, which must be used to run Java programs, together with a set of libraries (or "packages") needed to allow the use of file systems, networks, graphical interfaces, and so on, from within those programs.

## 4. APPLICATIONS AND FUTURE SCOPE

### 4.1. Applications

4.1.1. In the modern TV programs, there are more and more scrolling videotexts which can provide important information (i.e. latest news occurred) in the TV programs. Thus this text can be extracted.

4.1.2. Content based image retrieval involves searching for the required keywords in the images. Based on the results the images are retrieved.

4.1.3. Extracting the number of a vehicle in textual form:

- It can be used for tracking of a vehicle.
- It can extract and recognize a number which is written in any format or font.

4.1.4. Extracting text from a presentation video:

- The time consumed for reading the text from a video can be reduced as the text contained in the video can be retrieved without going through the complete video again and again.

The memory space required will also be reduced.

### 4.2. Future Scope

We have implemented the project considering English language; it can be further extended to other languages. If enlarged in future implementations, it will largely improve the efficiency of the algorithm.

# 5. EXPERIMENTS AND ANALYSIS

## 5.1 Input Image

Figure 2 shows the original image, which is given as the input to the system. The system's job is to extract the text from this image.


Figure 2: Original Image

## 5.2 Output Images

Figure 3 shows the processing of the original image at every phase and final extraction of the text. For the input image as shown in Figure 2, the extracting text is "***canvas***". Different phases of operations include:

- Gray Scaled Image (Figure 3).
- Lines Removal (Figure 4).
- Dot Removal (Figure 5).
- Discontinuity Removal (Figure 6).
- Recognized word (Final extracted text) (Figure 7).

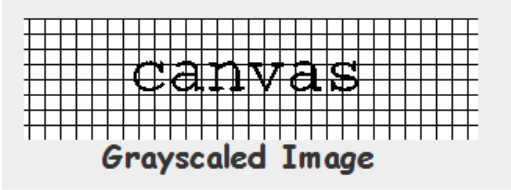


Figure 3: Gray Scaled Image

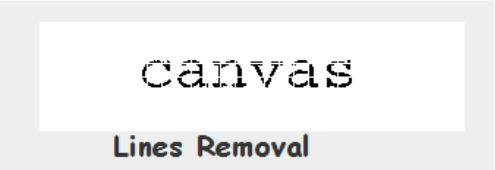


Figure 4: Lines Removal

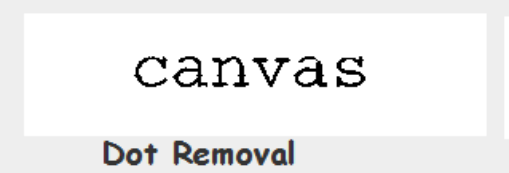


Figure 5: Dot Removal


Figure 6: Discontinuity Removal


Figure 7: Recognized Word

## ACKNOWLEDGEMENT

A single person cannot work directly or indirectly, somebody needs to help you. Before we get into thick of things we would like to add few heartfelt words for all the Teachers of Computer Department who took the pains to make our project successful.

We would like to express sincere thanks to Prof. Jayshree Ghorpade for her guidance, cooperation which helped us to successful completion of our project. We are also heartily thankful to our friends for their kind cooperation who have directly or indirectly helped us to complete the project successfully.

## CONCLUSIONS

Extraction of text information involves detection, localization, tracking, extraction, enhancement, and recognition of the text from a given image. A variety of approaches to text information extraction from images and video have been proposed for specific applications including page segmentation, address block location, license plate location, and content based image/video indexing. In spite of such extensive studies, it is still not easy to design a general purpose text information extraction system. This is because there are so many possible sources of variation when extracting text from a shaded or textured background, from low contrast or complex images, or from images having variations in font size, style, color, orientation, and alignment. These variations make the problem of automatic text information extraction extremely difficult. The problem of text information extraction needs to be defined more precisely before proceeding further. A text information extraction system receives an input in the form of a still image or a sequence of images. The images can be in gray scale or color, compressed or uncompressed, and the text in the images may or may not move. The text information extraction problem can be divided into the following sub problems:

- Detection
- Localization
- Tracking
- Extraction and Enhancement
- Recognition (NN)

Text detection, localization, and extraction are often used interchangeably in the literature. We can conclude that the text appearing in a video can be extracted using this software with the only restriction i.e. the video should not be very distorted. As, the characters are recognised on run-time basis, there may be a few cases in which one or two characters may get misrecognised i.e. a character may get recognised as some other character. But from the experiments performed on the software, most of the text gets recognized successfully.

## REFERENCES


[1] Extracting Textual Information from Images and Videos for Automatic Content Based Annotation and Retrieval by Julinda Gllavata.

[2] IEEE : An Edge-based Approach for Video Text Extraction,(This paper appears in : TENCON 2004. 2004 IEEE Region 10 Conference)

[3] IEEE: A Robust Algorithm for Text Extraction in Color Video'(This paper appears in: Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on Issue Date: 2000)

[4] K. Jung, K.I. Kim, and A.K. Jain. Text Information Extraction in Images and Videos: A Survey. Pattern Recognition Letters, 37:977–997, 2004.

[5] R. Lienhart and A. Wernicke. Localizing and Segmenting Text in Images and Videos. Transactions on Circuits and Systems for Video Technology, 12(4):256–268, 2002.

[6] N. Efford. Digital Image Processing: a Practical Introduction Using Java. Addison Wesley, 2000.

[7] http://java.sun.com/dev/evangcentral/totallytech/jmf.html


**Authors**


Prof. Jayshree Ghorpade, M.E. [Computer],

Assistant Professor, Computer Dept., MITCOE, Pune, India. 8+ years of experience in IT and Academics & Research. Area of interests are Image Processing, Data Mining & BioInformatics.

Raviraj P. Palvankar

Presently pursuing his Bachelor's degree in Computer Science in MIT College of Engineering, Pune, India. He is going to complete his engineering in May, 2011 and hopeful of career in Software Industry.

Ajinkya Patankar

A student of MIT College of Engineering, Pune, India. Presently he is final year student in a Department of Computer Engineering and going to complete his Bachelor's degree in May 2011.

Snehal Rathi

Presently pursuing his Bachelor's degree in Computer Science in MIT College of Engineering, Pune, India. He is going to complete his engineering in May, 2011 and hopeful of career in Software Industry.