iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** VI **Month of publication:** June 2020

DOI: http://doi.org/10.22214/ijraset.2020.6048

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VI June 2020- Available at www.ijraset.com

Auto Jotting: A Notes Maker

Venkata Vamsi Medidi[1], Srinivas Gouryraj[2] ,Trupthi M[3]
[1,2,3]*Department of Information Technology, Chaitanya Bharathi Institute of Technology*

Abstract: Online learning is the trend of the day and is the future of the education. During the online lectures, the tutors sometimes tend to write important points on the board. Though transcripts of the lectures are provided, everything that the tutor writes on the board might not be present in them. It is also difficult for the students to note down all the points that are delivered. In order to solve this problem and for efficient usage of time, a solution was proposed to build an automatic reliable system which will process a lecture video using image processing techniques, extracts text using OCR and converts into a text file which will be used as a précised notes of the lecture.This file can also be used for further revision of the lecture. The proposed system is planned into four phases. Firstly, the video is converted into a set of frames by using OpenCV in python. Secondly, extracting the text from the set of frames which we got from the first phase by using OCR. Then, NLP techniques will be applied to the extracted text to make it as a readable text. Finally, converting the extracted text into a text file.

Keywords: OCR - Optical Character Recognition, SWT - Stroke Width Transform, NLP – Natural Language Processing, MSER – Maximally Stable Extremal Regions, EAST – An Efficient and Accurate Scene Text Detection.

I. INTRODUCTION

Nowadays online videos became an important resource for students and other learners. Making notes from the videos will be useful for the learners for revision. But, making notes manually from the videos is a very difficult part. With the growth of online videos as a resource for students and learners to acquire knowledge, it is very difficult to make notes manually for every video. So, the proposed system is to build an automatic reliable system which will process a video using image processing techniques, extracts text using OCR and converts into a text file which will be used as a précised notes of the lecture. The proposed application is useful for making the notes from the videos without humane power. The proposed application will take a video as input and it will produce a text file containing the text written on the board. The text file which is produced by the application is also useful for revision purpose. The main objective of the proposed system is to make précised notes for the students/learners who are gaining knowledge through video lectures.

II. PROPOSED SYSTEM

A. Architecture

The proposed one usually contains six major stages. The first stage is converting of given video into frames by using read() and set() methods present in the OpenCV package of python. In the second stage, the duplicates from the extracted frames will be detected and those will be removed by using Hamming distance. In the third stage text regions in the frames will be detected and bounding boxes will be made around each sentence by using the predefined neural network model called EAST [4]. In the fourth step, the text present in the bounding boxes will be extracted by using the Optical Character Recognition technique. In the fifth step, the extracted text will be passed to the grammar correction API. In the final step, the text will be made in the form of a text file. The architecture for the proposed system is as shown below in Fig. 1.

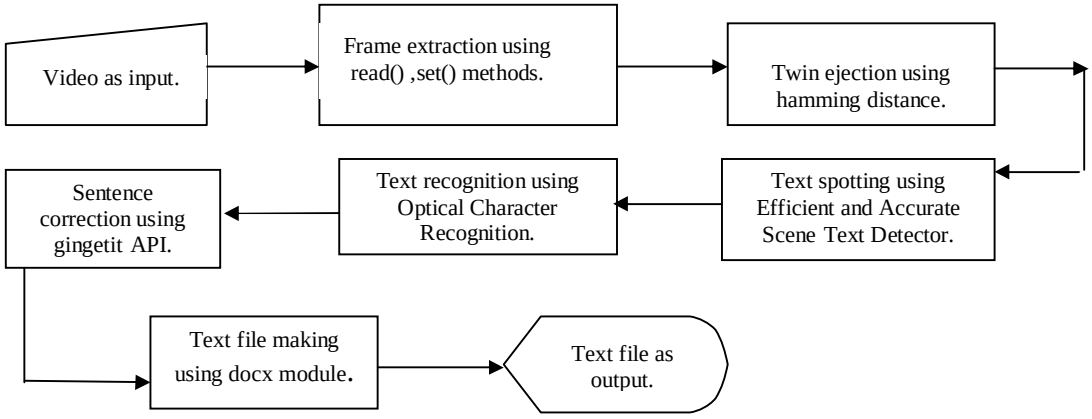


Fig. 1 Proposed Architecture

III. METHODOLOGY

A. Input

The proposed system will take video as input here, the user can provide input video by giving a you tube link or he/she can upload the video which is present in his local system. The length of the video that the proposed system can take will depend upon the speed of the processor and the speed of the internet in which the application is using. The proposed system will take video in MP3 format only if the user is going to upload it from his local system.

B. Frame Extraction

After taking video as input the proposed system will convert the video into frames. The extracted frames are stored in a folder in the local system. It depends on the time interval at which the frames have to be extracted. As per the experiments conducted, for every 5 seconds a frame will be extracted from the video input.

C. Twin Ejection

After getting the frames from the input video, the extracted frames may contain duplicates that are, the frames having the same text. It will increase the redundancy and it will consume time and memory by processing the duplicate frames. To reduce those issues the duplicate frames will be removed from the extracted frames by using Hamming distance measure which will be used for calculating the similarity between images. It will return a value that will explain the similarity between the images. If the value is close to zero then the images are similar and if the value is equal to or closer to 10 then those images are more dissimilar.

1) *Hamming Distance*: In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In other words, it measures the minimum number of substitutions required to change one string into the other or the minimum number of errors that could have transformed one string into the other. In a more general context, the Hamming distance is one of several string metrics for measuring the edit distance between two sequences. It is named after the American mathematician Richard Hamming. Here, Hamming distance is used for finding the similarity between two images. The X, Y resolutions of an image are converted into a binary 1-D array. The resolution arrays of two images are taken as an input to the hamming distance and it will return a value based on the below formula

$$\text{Hamming } (r_1, r2) = \text{no.of similar elements/no.of total elements.}$$

If the value is close to or equal to 1 then, images are more similar and if the value is equal and greater than 10 then the images are most dissimilar. By conducting several experiments that have value less than or equal to 2 is giving good results for the proposed application.

D. Text Spotting

The next step in the proposed system is to detect the text present in the frames after removing duplicates from the frames. For text spotting, three different techniques were applied those are –
Maximally Stable Extremal Regions (MSER) [5].
Stroke Width Transformation (SWT) [6].
An Efficient and Accurate Scene Text Detector (EAST) [4].

1) *Maximally Stable Extremal Regions (MSER)[5]:* MSER is a method for blob detection in images. The MSER algorithm extracts from an image several co-variant regions, called MSERs: an MSER is a stable connected component of some gray-level sets of the image.

Key idea: Consider a movie generated by thresholding a gray level image (intensity values [0, 1,…, 255]) at all possible thresholds starting from 0 and ending at 255. Initially, it's an empty image and then some dots (local minima) appear and start growing New dots appear and start growing and so on From time to time two disconnected regions get merged Finally, all regions get merged into a single component. But, the important observation here is that – starting from a tiny seed area (one or a few pixels), a region continues growing 24 till it fills the object containing the initial seed area and then remains (almost) unchanged for quite some time in the threshold movie until it gets merged with the bigger (generally, parent) object to which it belongs MSER intends to capture these stable regions.

Algorithm (MSER enumeration)
Input: Image I and the Δ parameter
Output: List of nested extremal regions.
Step 1: For all pixels shorted by intensity
i. Place a pixel in the image as its tern come
ii. Update the connected component structure
iii. Update the area for the effected connected components
Step 2: For all connected components Detect regions with local minima w.r.t. rate of change of connected component area with a threshold; define each such region as an MSER.

2) *Stroke Width Transform (SWT) [6]:* The algorithm receives an RGB image and returns an image of the same size, where the regions of suspected text are marked. It has 3 major steps: the stroke width transform, grouping the pixels into letter candidates based on their stroke width, and finally, grouping letter candidates into regions of text. The Stroke Width Transform - A stroke in the image is a continuous band of nearly constant width. An example of a stroke is shown in Fig. 2(a). The Stroke Width Transform (SWT) is a local operator that calculates for each pixel the width of the most likely stroke containing the pixel.

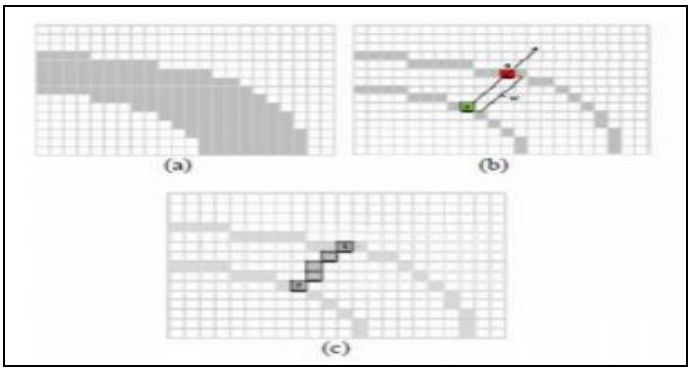


Fig. 2 Stroke width [6]

First, all pixels are initialized with ∞ as their stroke width. Then, calculate the edge map of the image by using the canny edge detector. Consider the edges as possible stroke boundaries, and wish to find the width of such stroke. If p is an edge pixel, the direction of the gradient is roughly perpendicular to the orientation of the stroke boundary. Therefore, the next step is to calculate the gradient direction g_p of the edge pixels and follow the ray $r=p+n*g_p$ ($n>0$) until finding another edge pixel q. If the gradient direction g_q at q is roughly opposite to g_p, then each pixel in the ray is assigned the distance between p and q as their stroke width, unless it already has a lower value. If, however, an edge pixel q is not found, or g_q is not opposite to g_p, the ray is discarded. To accommodate both the bright text on a dark background and dark text on a bright background, need to apply the algorithm twice: once with the ray direction g_p and once with $-g_p$.
After the first pass described above, pixels in complex locations might not hold the true stroke width value (Fig. 3(b)). For that reason, pass along each non-discarded ray, where each pixel in the ray will receive the minimal value between its current value, and the median value along that ray.

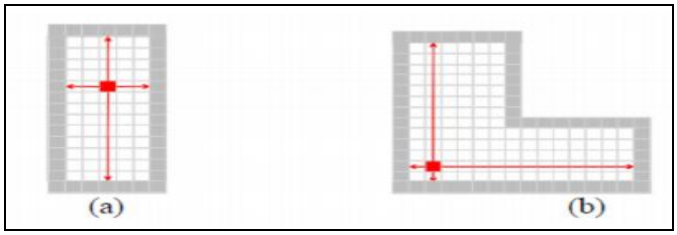


Fig. 3 Stroke width values [6]

Finding Letter Candidates: Now have a map of the most likely stroke-widths for each pixel in the original image. The next step is to group these pixels into the letter candidate. This will be done by first grouping pixels with similar stroke width and then

applying several rules to distinguish the letter candidates. The grouping of the image will be done by using a Connected Component algorithm. To allow smoothly varying stroke widths in a letter, let two pixels be grouped if their SWT ratio is less than 3.0. Now must detect the connected components which can pass as letter candidates, by applying a set of fairly flexible rules. These rules are as follows:

The variance of the stroke-width within a component must not be too big. This helps with rejecting foliage in natural images, which are commonly mistaken for text.

The aspect ratio of a component must be within a small range of values, to reject long and narrow components.

The ratio between the diameter of the component and its median stroke width to be less than a learned threshold. This also helps to reject long and narrow components.

Components whose size is too large or too small will also be ignored. This is done by limiting the length, width, and pixel count of the component.

The remaining connected components are considered letter candidates and are now to be aggregated into regions of text.

Grouping Letter Candidates to Text Regions: Since single letters are not expected to appear in images, now attempt to group closely positioned letter candidates into regions of text. This filters out many falsely-identified letter candidates and improves the reliability of the algorithm results. Again, use a small set of rules to group letters together into regions of text. These rules will consider pairs of letters, and are as follows:

Two-letter candidates should have similar stroke width. For this reason, limit the ratio between the median stroke-widths to be less than some threshold.

The ratio between the heights of the letters and between the widths of the letters must not exceed 2.5. This is due to capital letters next to lower case letters.

The distance between letters must not exceed three times the width of the wider one.

Characters of the same word are expected to have a similar colour; therefore compare the average colour of the candidates for pairing.

3) *An Efficient and Accurate Scene Text Detector (EAST) [4]*: OpenCV's EAST text detector is a deep learning model, based on a novel architecture and training pattern. It is capable of running at near real-time at 13 FPS on 720p images and obtains state-of-the-art text detection accuracy. With the release of OpenCV 3.4.2 and OpenCV 4, now use a deep learning-based text detector called EAST, which is based on Zhou et al.'s 2017 paper, ***EAST: An Efficient and Accurate Scene Text Detector [4]***. Call the algorithm "EAST" because it's an: **E**fficient and **A**ccurate **S**cene **T**ext detection pipeline. The EAST pipeline is capable of predicting words and lines of text at arbitrary orientations on 720p images and can run at 13 FPS, according to the authors. Perhaps most importantly, since the deep learning model is end-to-end, it is possible to sidestep computationally expensive sub-algorithms that other text detectors typically apply, including candidate aggregation and word partitioning. To build and train such a deep learning model, the EAST method utilizes a novel, carefully designed loss function.

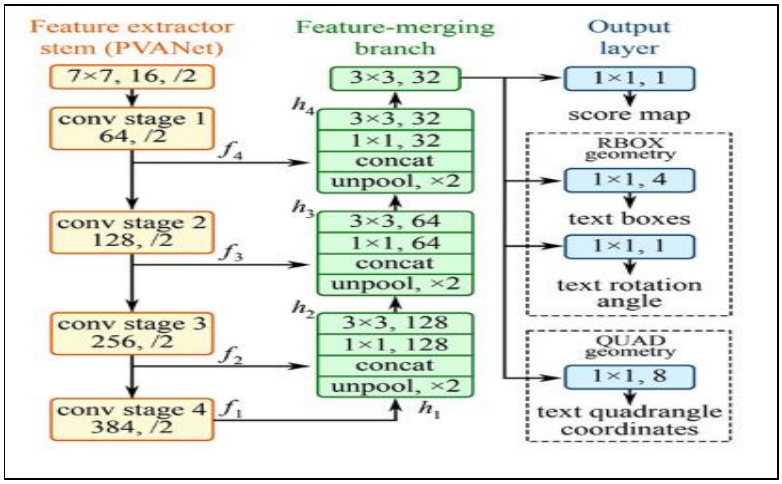


Fig. 4 EAST architecture[4]

E. Text Extraction

After text spotting the text has to be extracted to make a text file for that reason a good text extraction technique PyTeserract – OCR is used. Tesseract has **Unicode (UTF-8) support**, and can **recognize more than 100 languages** "out of the box".

1) Working: The first step is a connected component analysis in which outlines of the components are stored. This is a computationally expensive design decision at the time but has a significant advantage: by inspection of the nesting of outlines, and the number of child and grandchild outlines, it is simple to detect inverse text and recognize it as easily as black-on-white text. Tesseract is probably the first OCR engine able to handle white-on-black text so trivially. At this stage, outlines are gathered together, purely by nesting, into Blobs. Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces. Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each satisfactory word is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize the text lower down the page. Since the adaptive classifier may have learned something useful too late to contribute to the top of the page, a second pass is run over the page, in which words that were not recognized well enough are recognized again. A final phase resolves fuzzy spaces and checks alternative hypotheses for the x-height to locate small-cap text.

F. Sentence Correction and Text File Making

After extracting the text present in the frames sentence correction is one of the post-processing techniques for OCR to get good results. For sentence spelling and grammar corrections language check, textbolb, spell checker, grammar bot, gingerit and concluded that gingerit is giving good results compared to the other APIs. After getting correct sentences those are made into a doc file by using python's docx module.

IV. TESTING AND RESULTS

Firstly a video is taken as input and frames are extracted from the input video as explained in the methodology section. For a particular video Fig. 5 is the output after extraction of frames.

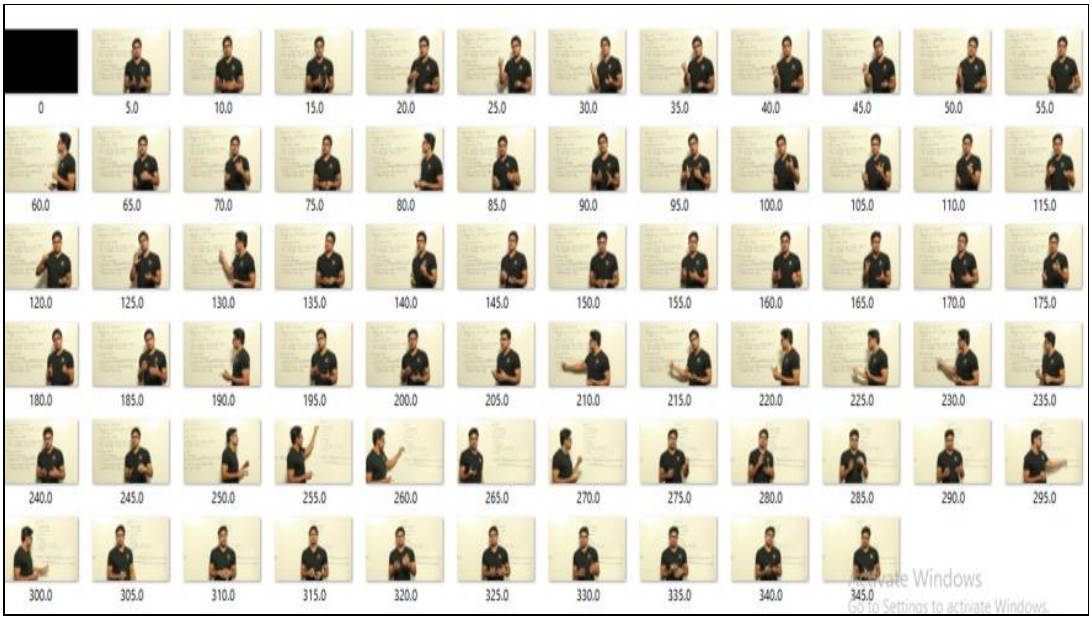


Fig. 5 Extracted frames for a video.

Then, the extracted frames may contain duplicates which will cause redundacy. In order to decrease the rdundency twins in the frames are ejected by using hamming distance s expained in the methodology section. The output after removing duplicates from the above shown frames is as shown in Fig. 6.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VI June 2020- Available at www.ijraset.com

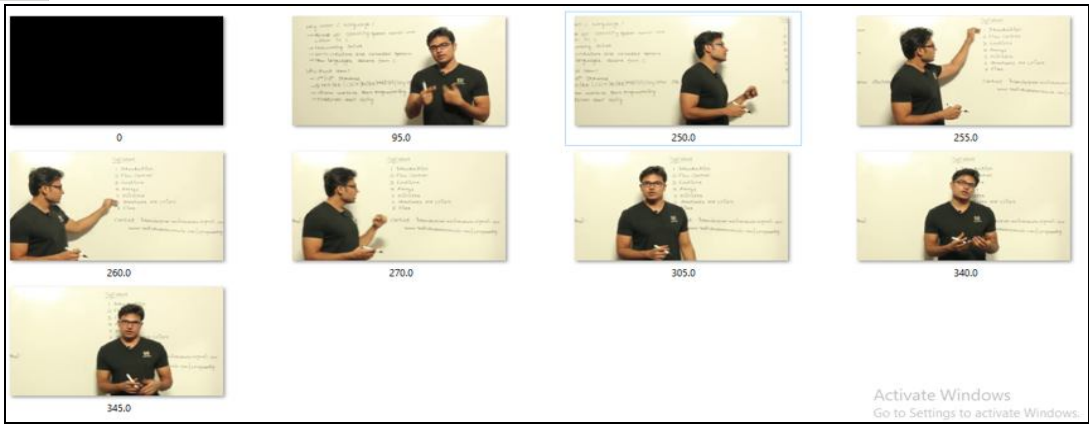


Fig. 6 Frames after Twin Ejection.

After removing of duplicates from the extracted frames, text has to be detected which is present in the frames. For spotting of the text three different techniques are used as explained in the methodology secction and the results for each technique is shown in Fig. 7, Fig. 8 and Fig.9.

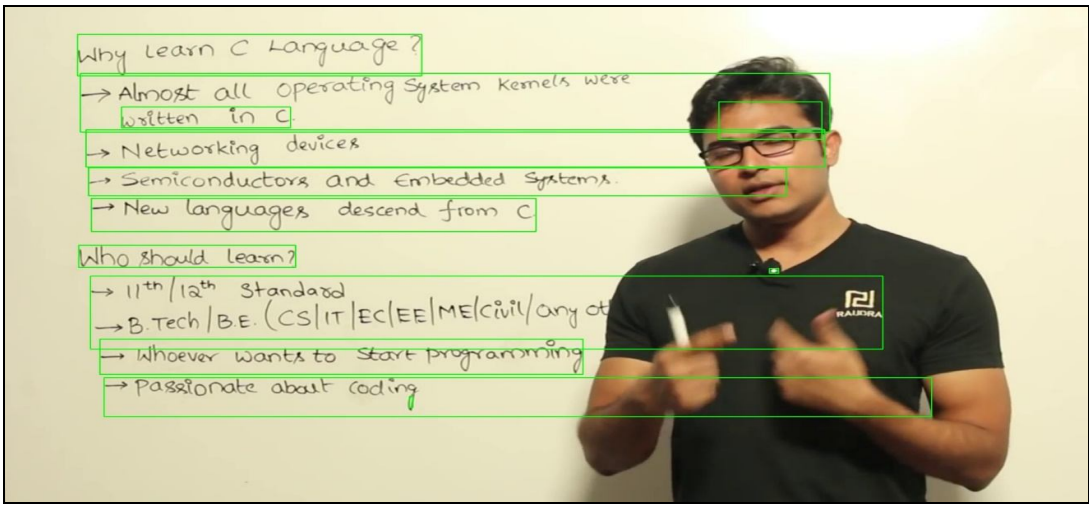


Fig. 7 Text spotting by MSER.

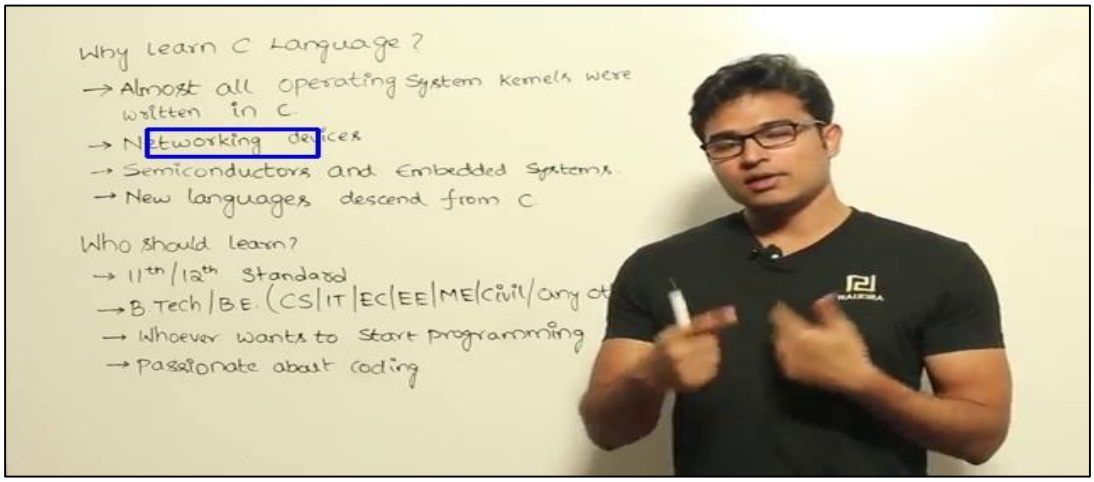


Fig. 8 Text spotting by SWT.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VI June 2020- Available at www.ijraset.com

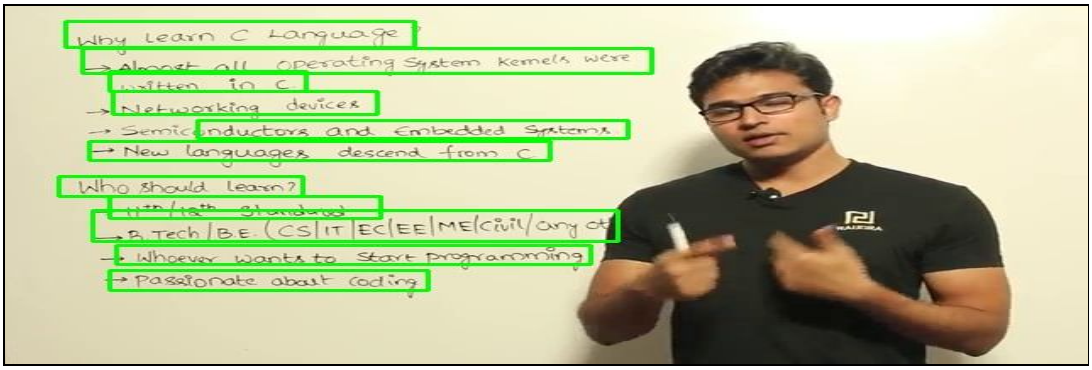


Fig. 9 Text spotting by EAST

As seen in the above figures that MSER is detecting non-textual regions too. So, it is moved to SWT but, SWT is unable to detect all textual regions then, shifter to EAST which is based on CNN able to detect 98% of the text present in the frames. So, EAST is giving good results when compared to other techniques.

After spotting of the text, text has to be extracted it is done by using PyTesseract- OCR technique as explained in the Methodology section. For extraction of the text efficiently the frame is cropped into spotted text regions as shown in the Fig. 10. Then for those cropped images PyTesseract-OCR is applied as shown in Fig. 11.

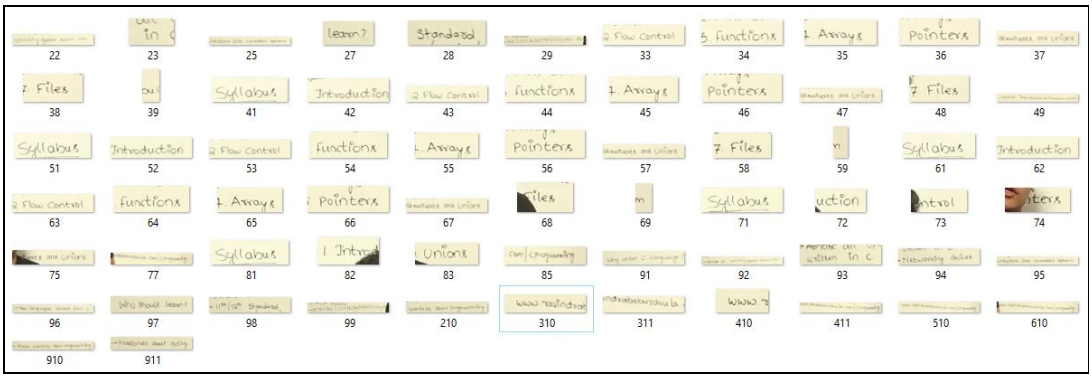
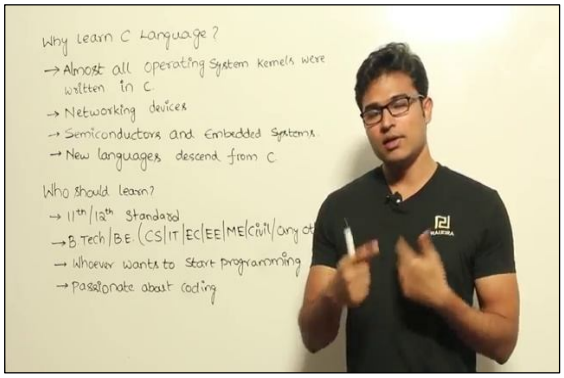
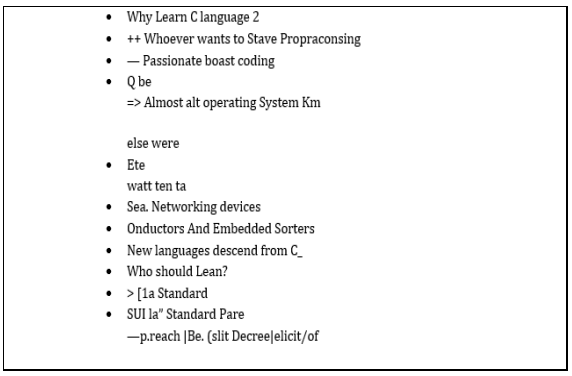


Fig. 9 Cropped images.

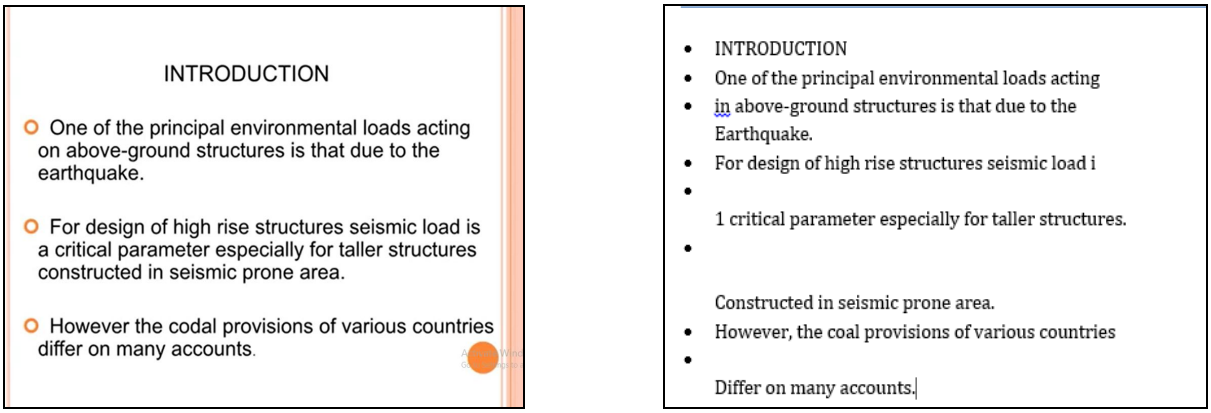




Input Output

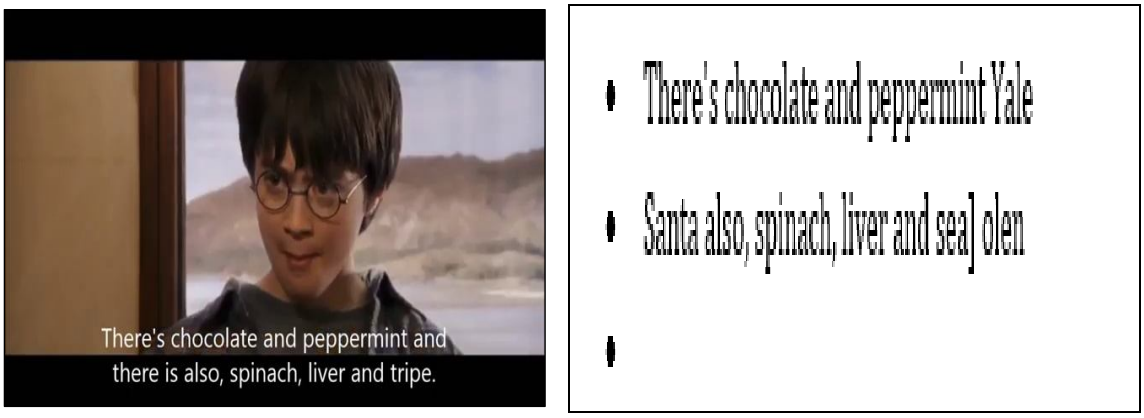
Fig. 10 Result of a handwritten text..

Now the extracted text is converted into a doc file by using techniques explained in the sentence correction and text file making under methodology section.
The proposed system not only works for handwritten text but, it is giving more accuracy for typed text and for making transcript of subtitles in a video as shown in the below Fig. 11 and Fig. 12.



Input Output

Fig. 11 Result of a PPT slide.



Input Output

Fig. 12 Result of a subtitle frame.

V. CONCLUSIONS AND FUTURE SCOPE

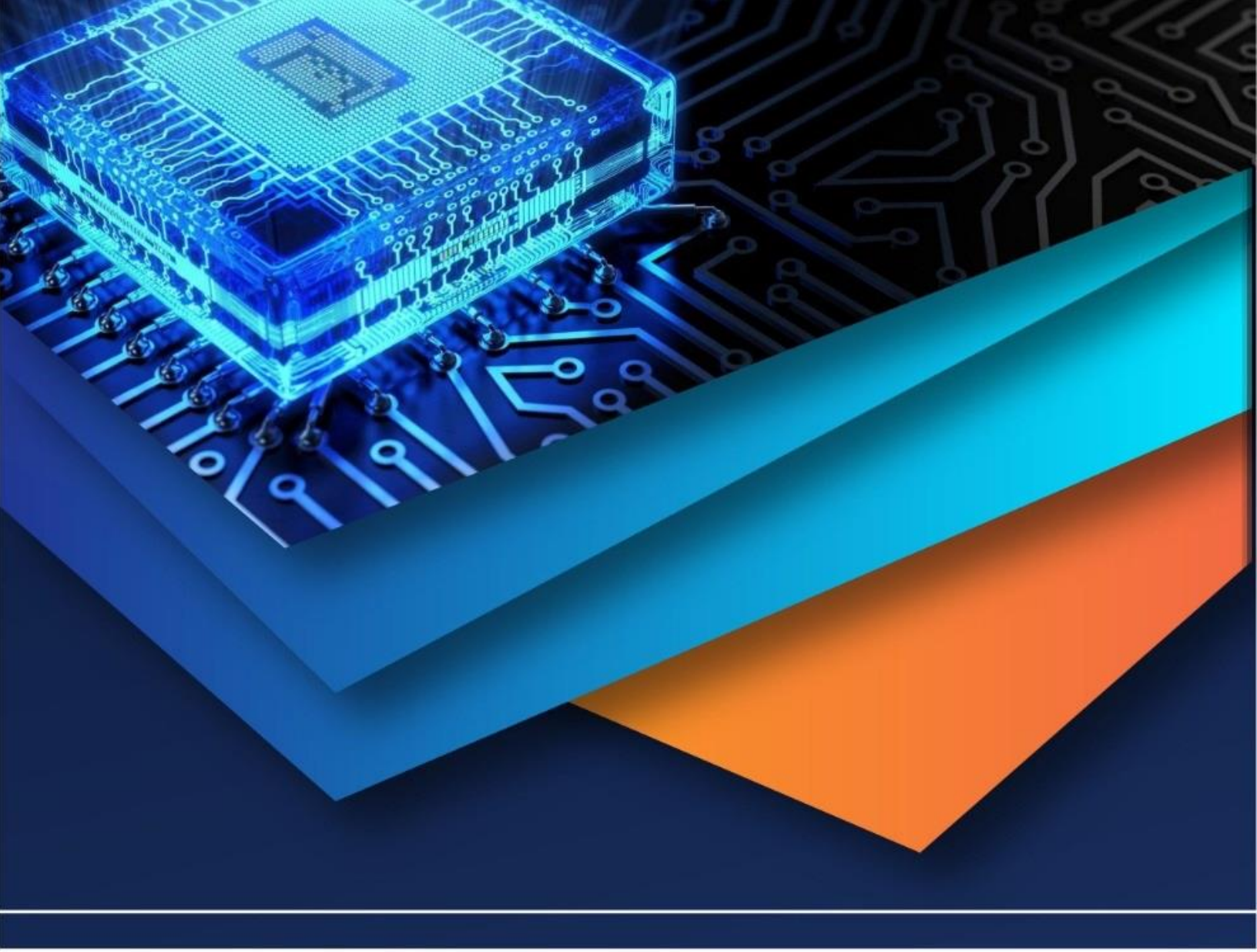
The proposed model, Auto jotting tends to solve the problem of making précised notes of the video lectures effectively. Different algorithms like MSER, SWT, and EAST were tried to build the model. MSER had the problem of overfitting wherein it tried to detect the non-text regions as well and SWT had the problem of under fitting wherein it detected only limited text regions. EAST, which is based on neural networks, outperformed all the other algorithms and gave the accurate results as shown in the results section. The built model has the greater accuracy for the typed text like PPT and subtitle text whereas the results do not seem promising for the handwritten text when compared to typed text and we propose to improve our model further for the handwritten text. In order to increase the accuracy for handwritten text we will implement our own CNN model for text extraction. To reduce the redundancy of text while making text file need to remove duplicate images for that we are going to compare the image similarity with respect to content present in them.

VI. ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Mrs. M. Trupthi**, Asst. Professor, Dept. of IT, our project guide for her valuable guidance and constant support, along with her capable instructions and persistent encouragement. We are grateful to our Head of the Department, **Dr. Suresh pabboju**, for his steady support and provision of every resource required for the completion of this project. We would like to take this opportunity to thank our principal, **Dr. P. Ravinder Reddy**, as well as the management of the institute, for having designed an excellent learning atmosphere.

REFERENCES

[1] Wonjun Kim and Changick Kim, "A New Approach for Overlay Text Detection and Extraction From Complex Video Scene", IEEE Transactions on Image Processing, vol. 18, page no. 401-411, Feb 2009.
[2] Yuxiang Jiang, Haiwei Dong and Abdulmotaleb El Saddik, "Baidu Meizu Deep Learning Competition: Arithmetic Operation Recognition Using End-to-End Learning OCR Technologies", IEEE Access, vol. 6, page no. 60128-60136, Oct 2018.
[3] Daojian Zeng, Haoran Zhang, Lingyun Xiang, Jin Wang, and Guoliang Ji, "User-Oriented Paraphrase Generation With Keywords Controlled Network", IEEE Access, vol. 7, page no.80542-80551, Jun 2019.
[4] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang, "EAST: An Efficient and Accurate Scene Text Detector", 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Nov 2017.
[5] (2012) The MICC website. [Online]. Available: http://www.micc.unifi.it/delbimbo/wp-content/uploads/2011/03/slide_corso/A34%20MSER.pdf.
[6] (2013) The BGU website. [Online]. Available: https://www.cs.bgu.ac.il/~ben-shahar/Teaching/Computational-Vision/StudentProjects/ICBV131/ICBV-2013-1-GiliWerner/ICBV-2012-1-GiliWerner-report.pdf.
[7] Shu Tian, Xu-Cheng Yin, Ya Su, and Hong-Wei Hao, "A Unified Framework for Tracking Based Text Detection and Recognition from Web Videos", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, page no. 542-554, Mar 2018.
[8] Minghui Liao, Baoguang Shi and Xiang Bai, "TextBoxes++: A Single-Shot Oriented Scene Text Detector", IEEE Transactions on Image Processing, vol. 27, page no. 3676-3690, Apr 2018.
[9] Jie Lei, Qiao Luan, Xinhui Song, Xiao Liu, Dapeng Tao and Mingli Song, "Action Parsing-Driven Video Summarization Based on Reinforcement Learning", IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, page no. 2126-2137, Jul 2018.
[10] Huiping Li, D. Doermann and O. Kia, "Automatic text detection and tracking in the digital video", IEEE Transactions on Image Processing, vol. 9, page no. 147-156, Jan 2000.
[11] Peng Yang, Fanlong Zhang and Guowei Yang, "A Fast Scene Text Detector Using Knowledge Distillation", IEEE Access, vol. 7, page no. 22588-22598, Jan 2019.
[12] Kenny Davila and Richard Zanibbi, "Whiteboard Video Summarization via Spatio-Temporal Conflict Minimization", 14th IAPR International Conference on Document Analysis and Recognition, vol. 17, page no. 355-362, 2017.
[13] Greg C. Lee, Fu-Hao Yeh, Ying-Ju Chen, and Tao-Ku Chang, "Robust handwriting extraction and lecture video summarization", Multimed Tools Appl, vol. 76, page no. 7067-7085, Feb 2016.
[14] Bhargava Urala Kota, Kenny Davila, Alexander Stone, Srirangaraj Setlur and Venu Govindaraju, "Automated Detection of Handwritten Whiteboard Content in Lecture Videos for Summarization", 16th International Conference on Frontiers in Handwriting Recognition, page no. 18-24, 2018.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY