PROJECT COMPLETION REPORT

PROJECT TITLE

Text detection, text matching and image matching within GUI

application screen images (DIMG)

submitted by

Moushumi Medhi : 16AT92P07

PI : Rajiv Ranjan Sahay

Department of Electrical Engineering Indian Institute of Technology, Kharagpur 721302 West Bengal, India

Project code: DIMG

Project Duration: 20-08-2015 to 19-08-2018

Table of Contents

A. Project code: DIMG
B. Project Title: Text detection, text matching and image matching within GUI application screen images (DIMG).
C. Duration of Project: 20-08-2015 to 19-08-2018
D. Principal Investigator: Dr. Rajiv Ranjan Sahay, Assistant Professor, Department of Electrical Engineering,
E. Organization: IIT Kharagpur
F. Sponsoring Agency: Altair Engineering India Pvt. Ltd.,Mercury 2B Block, 5th Floor, Prestige Tech Park, Sarjapur Marathahalli Outer Ring Road, Bangalore - 560 103

Report

1. Project Objectives

The primary goal is to extract text from cluttered, shaded, textured, complex, low contrast background where there are so many possible sources of variation. Text can be interpreted as "structured edges", "a group of strokes", "connected components", "stable extremal regions", "high frequency components", "a kind of texture", or a combination of all. However, there are so many stable, structured edge objects and high frequency textures bearing similar characteristics, strokes or texture properties as that of text, making it difficult to design effective feature representation to discriminate text, which might again vary widely in color, size, fonts, style, appearance, layout, aspect ratio, orientation, alignment, clutter background, distortion, language context, resolution. Text recognition confronts challenges beyond those in general object recognition and is yet an unsolved problem.

2. Fully achieved/ Partially achieved

Fully achieved.

3. Product/Process developed/Technology transferred

We have developed and delivered a standalone application to identify texts and read the detected texts in images.

4. Manpower trained

One Research Consultant employed in the project.

5. Suggestions for Utilization of Project Outcome

The text detection algorithm proposed in the project can be utilized by the research community to extract text from cluttered, shaded, textured, complex, low contrast background where there are so many possible sources of variation. The proposed work on text detection can also be easily extended to address the several other interconnected or related

computer vision problems such as object recognition tasks which demands detection and suppression of text first, such as, images containing logos.

6. Product/Process as an outcome of the Project

We have developed and delivered a standalone application to identify texts and read the detected texts in images. We have explored various text detection algorithms, including, hand-crafted and deep learning based methods for the task of text detection and identification.

7. Methodology for text detection

7.1 MSERs (Maximally Stable Extremal Regions) based text detection.

MSER [1] are stable connected components of some gray-level sets of the image and stay nearly the same through a wide range of thresholds. MSER-based text detection can be expected to be a reliable approach as it can be tuned to perform full text detection in still frames and is also effective in terms of computation cost and speed. It works well for text because the consistent color and high contrast of text leads to stable intensity profiles.

Implementation techniques: OpenCV [2] Text regions are detected in cropped images but takes too long to execute.

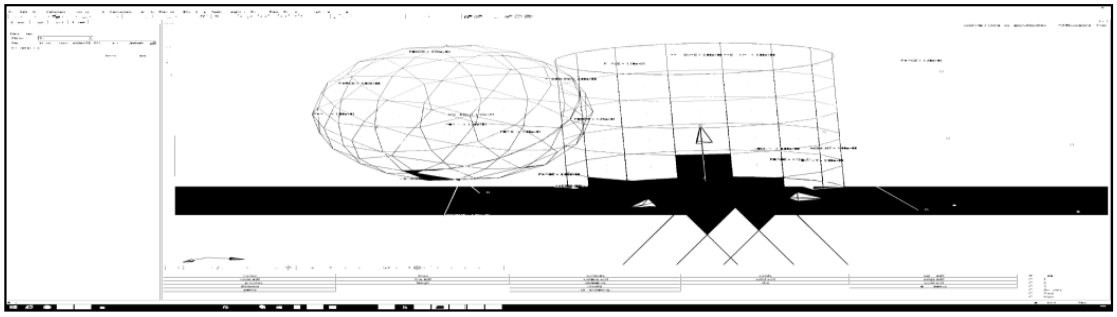

Figure 1: Segmented regions displayed after color thresholding.

Text regions are detected in cropped images in Figures 1 and 2 but

takes too long to execute.

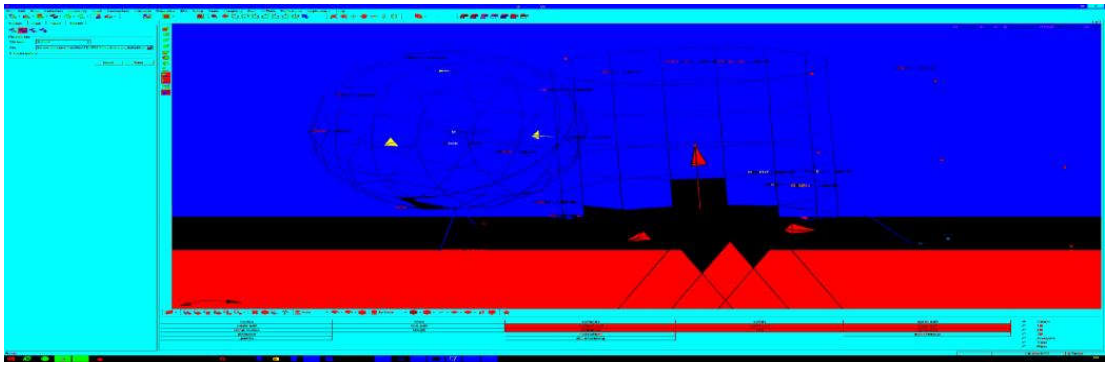

Figure 2: Color segmented regions.

Implementation: Matlab

All the mser regions in Figures 3, 4 are partitioned and are plotted using arbitrary random colors from jet colormap for visualization. Filtering out non text msers based on geometric properties.

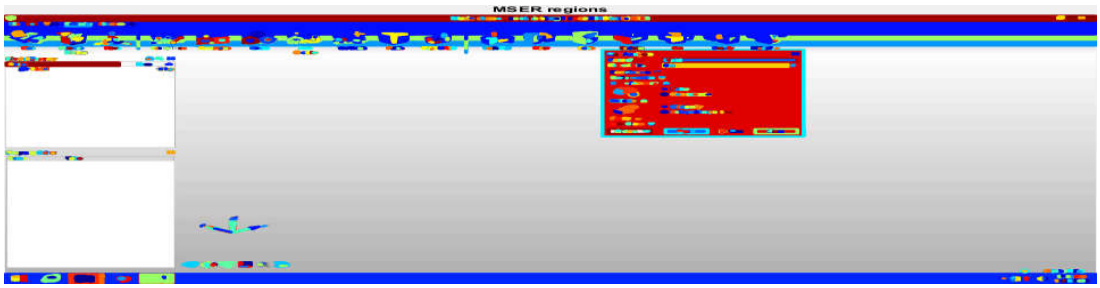

Figure 3: Detection of MSER using Matlab.

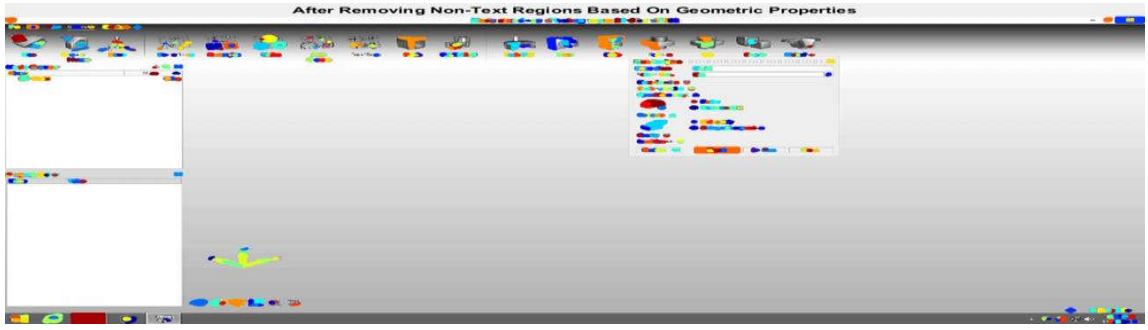

Figure 4: Filtering of the MSERs.

To see closely and clearly the working of mser algorithm and the intermediate steps (bounding boxes around msers and filtering of BBs), it is tested on cropped and magnified images. Magnification increases the contour area and helps better visualization, provided magnification does not lead to blurred or pixelated image which is why it failed on magnifying by 25 times. As can be seen from previous examples, mser algorithm is working on the full images and hence need not be cropped. However if need arises, then both can be automated. But in the

deployment version it needn't be cropped as they can be magnified using a sliding window rule.

Figure 5: Detection of MSERs in cropped regions

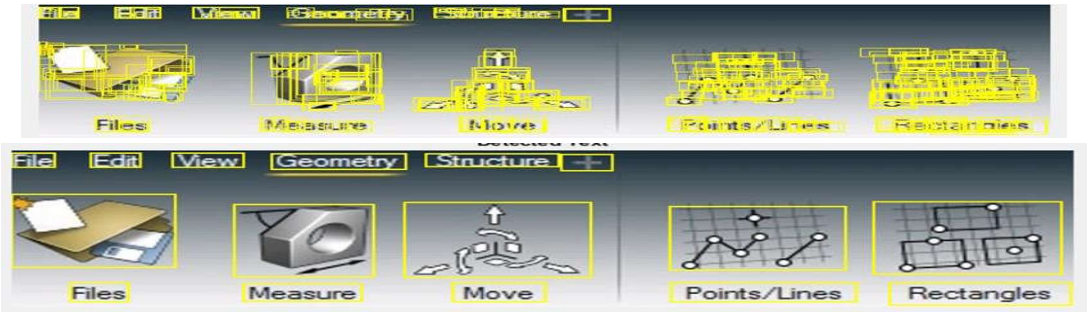

Figure 6: Proposal filtering and final detected proposals.

Better candidates are generated as shown in Figure 6 with a reasonably moderate magnification factor. Highly magnified image turns out to be grainy, pixelated or blurred where MSER fails.

7.2 SWT (Stroke Width Transform)

SWT (Stroke Width Transform) [3] is an operator which calculates for each image pixel the width of the most likely stroke containing the pixel. It is based on the idea of detecting text based on small variance of stroke widths.

Implementation: OpenCV and Boost

Boost [4] is a set of libraries for the C++ programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, and unit testing .Here **Boost Graph Libraries (BGL)** [5] are used to create graph, and add vertices and edges which are mathematical abstractions created for connected component analysis [6].

The coding for swt computation is such that the coordinates of the pixels with constant stroke width of each of the components are stored as set of vectors which are considered as edges and are stored in add_edge() function provided by boost library.

Console window

```
Running textDetection with dark_on_light 1
Before filtering, 320 components and 161935 vertices
```

Figure 7: Console window displaying the number of components and vertices.

Figure 8: Detection of Canny edges.

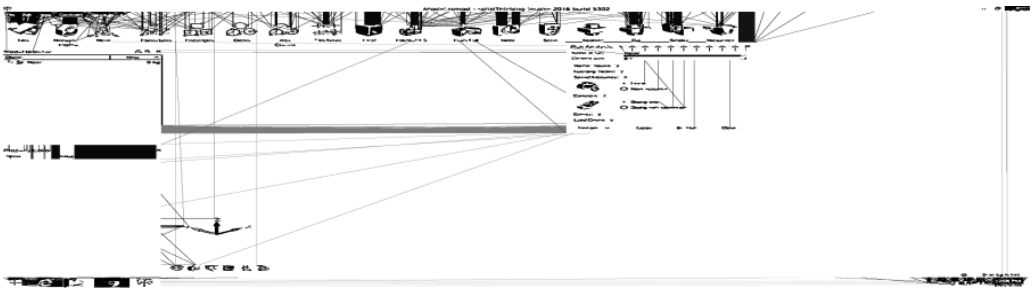

Figure 9: Generation of Stroke Widths.

The pixels in each of the lines in Figure 9 form a component with similar stroke width. As stroke widths are computed for a set of pixels with edge pixels at both ends, certain pair of pixels are wrongly considered as edge pixels due to opposite gradient directions of each of the two edge pixels and hence appear as lines in the output image. Due to high dimension of the image and too many components, the final output could not be obtained as the code enters into an infinite loop, so we cropped parts of the image and the results were examined.

Implementation : Matlab + CPP MEX file

A C++ MEX file is built in matlab using Microsoft Visual C++ from a C++ source file that computes the SWT image. The input image is pre-processed in matlab and then the mex file is called to obtain the resultant image containing constant stroke width components which is finally displayed in matlab figure window.

Figure 10: Stroke widths generated using Matlab + CPP MEX file.

The resultant images in Figure 11 retain both text as well as many non-text components which need to be further processed at latter stages to eliminate the false detections. At this stage it is rather making foreground objects appear clearer.

Implementation: OpenCV

Step1: Morphological transformations

Top hat: Top-hat transform [7]-[10] is the difference between the input image and its opening by some structuring element. Opening is defined as erosion of an image followed by dilation. Erosion and dilation are two basic operators in mathematical morphology.

Erosion: A 3x3 matrix of logical 1's is taken for the structuring element [11], with the middle point as the structuring element which is superimposed on each input pixel and the input pixel is set to foreground if the input pixel along with all its 8 neighbours are also set to foreground, otherwise it is set to background.

Dilation: Reverse of erosion, where the pixel is set to background only if the input pixel along with all its 8 neighbours are set to background.

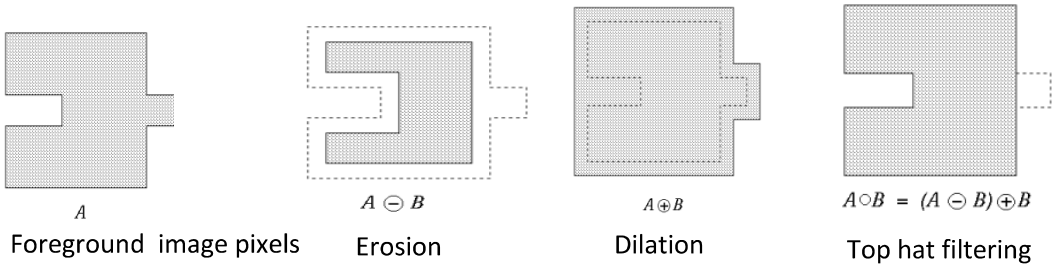

Figure 11: Morphological operations.

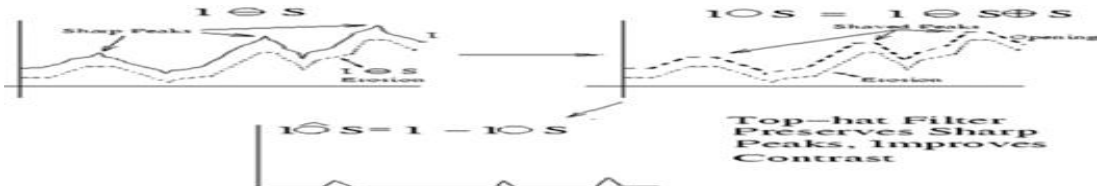

Figure 12: Demonstration of top hat filter preserving sharp peaks.

Top-hat filter preserves sharp peaks and eliminates other features, including the poor contrast features as shown in Figure 16.

Figure 13: Output after applying morphological operations.

Step2: Thresholding
Bi-level (binary) image is obtained out of a grayscale image to remove noise, that is, to filter out pixels with too small or too large values.

Figure 14: Output after thresholding.

Step3: Gaussian filtering
Gaussian filtering [12] is applied to smoothen the image.

Figure 15: Output after applying Gaussian filtering.

Step4: Image dilation

Bright regions within an image are expanded.

Figure 16: Output after applying image dilation.

Step5: Canny edge image

Figure 17: Output on applying Canny edge detection algorithm.

Step6: Detected proposals

Figure 18: Detected text proposals are shown in green bounding boxes.

The results in Figure 22 show acceptable results as the parameters can be tuned to detect all the foreground regions and very small or large blobs can be later filtered out.

Imlementation: Python + OpenCV

Method 1: Dark text detection

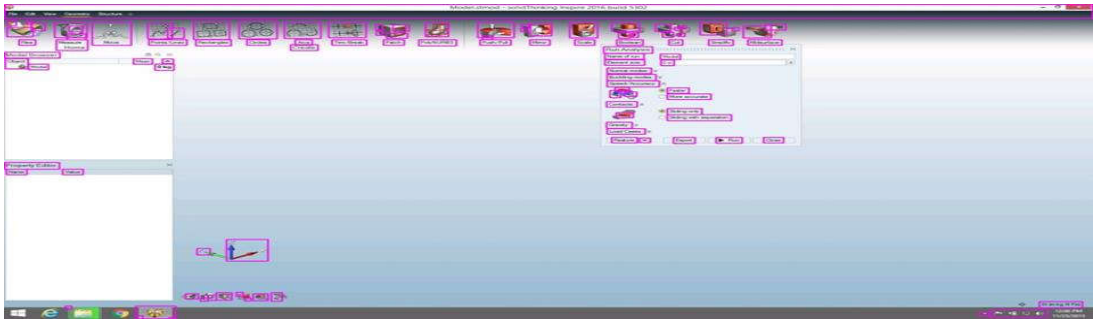

Figure 19: Results for dark text detection.

If furthering processing is planned to be carried out in python, then this method can be useful for text candidates extraction but it has be run twice for detection of both dark and light text. Dark text is detected in Figure 23 and therefore the date and time is missed. It is not actually the light text at the top which is detected but the dark strip at the background which seems like a foreground object. Again the light icons at the bottom right has a thick dark boundary. After fine tuning the parameters of previous method, we have the following results:

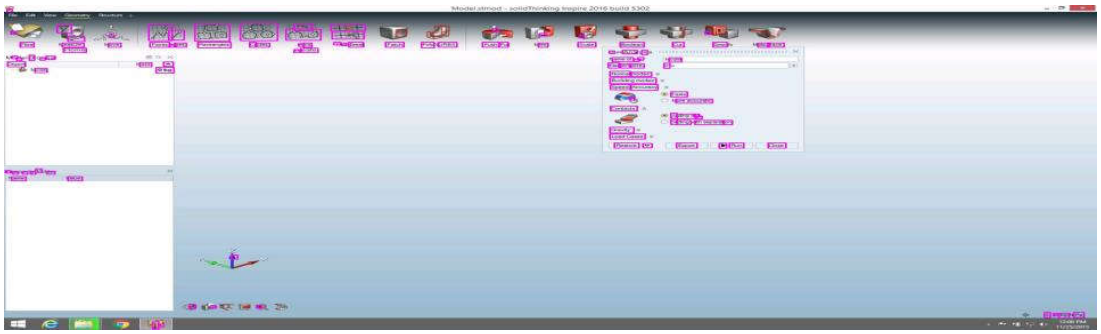

Figure 20: Fine-tuned results.

Method 2: Light text detection

Figure 21: Results for light text detection.

Methods 1 detects all the foreground objects at a time while method 2 detects them at two runs but with lesser false positives than method 1.

Also method2 fails for light colored text detection. Method 1 would still be more preferable as it detects all the objects at one go which can subsequently be filtered.

Implementation: Python + OpenCV +NumPy

NumPy [13] is an extension to Python that adds support for large, multi-dimensional data, along with a large library of mathematical functions to operate on these data.

Figure 22: Dark pixel detection.

Black pixels form the foreground detected objects which are generated based on pixel color thresholding and hence would not be an effective method for multicolored text proposal generation.

Implementation: Matlab

Method1: Character level detection

Step 1: Conversion to binary image

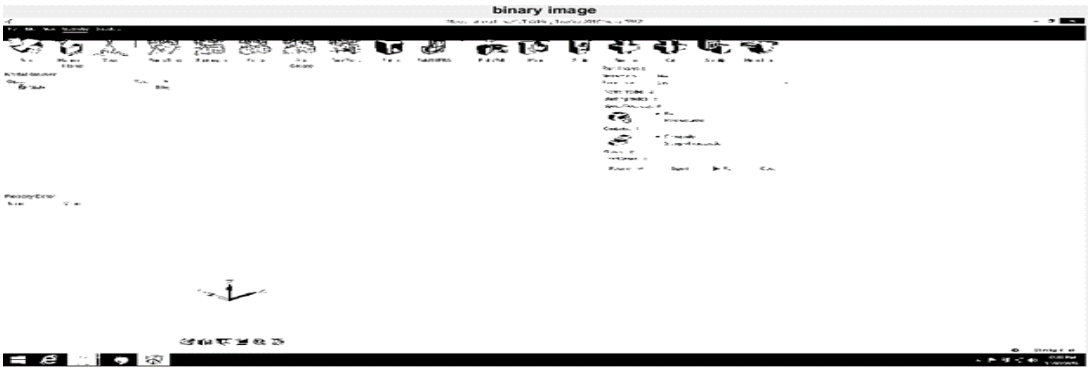

Figure 23: Binary image formation.

Step 2: Filling up of holes in the text in the inverted binary mage

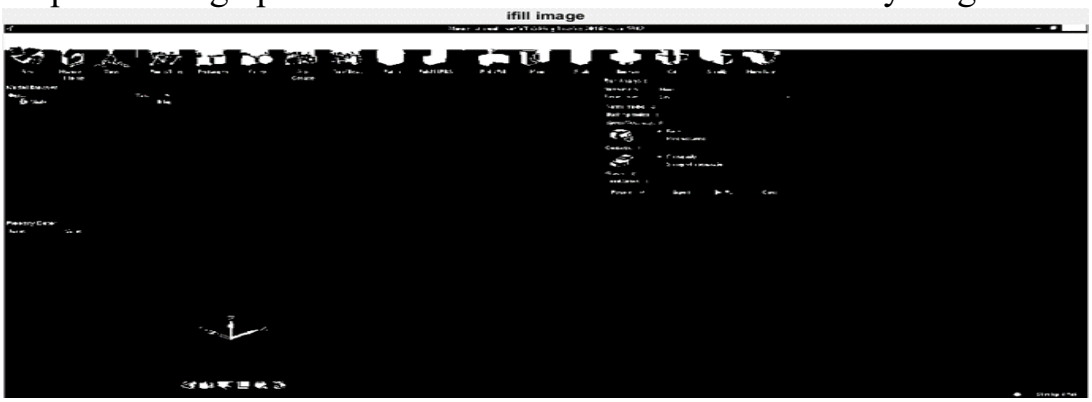

Figure 24: Hole filling in inverted binary image.

Step 3: Detected Objects

Figure 25: Detection of objects.

Method 2: Word level detection

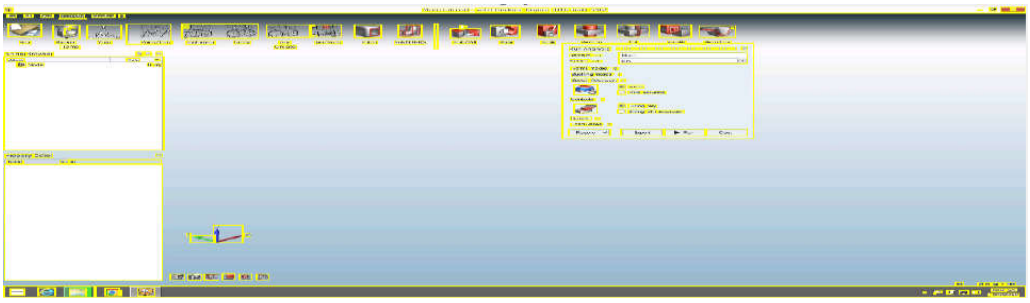

Figure 26: Word level detection.

All the above detection techniques has been employed to extract objects and observe the results in different platforms and a decision of selection of any of the techniques would require a clear idea of what and how the task would be carried out in post processing stage.

7.3 Optical Character Recognition (OCR)

Implementation: Matlab

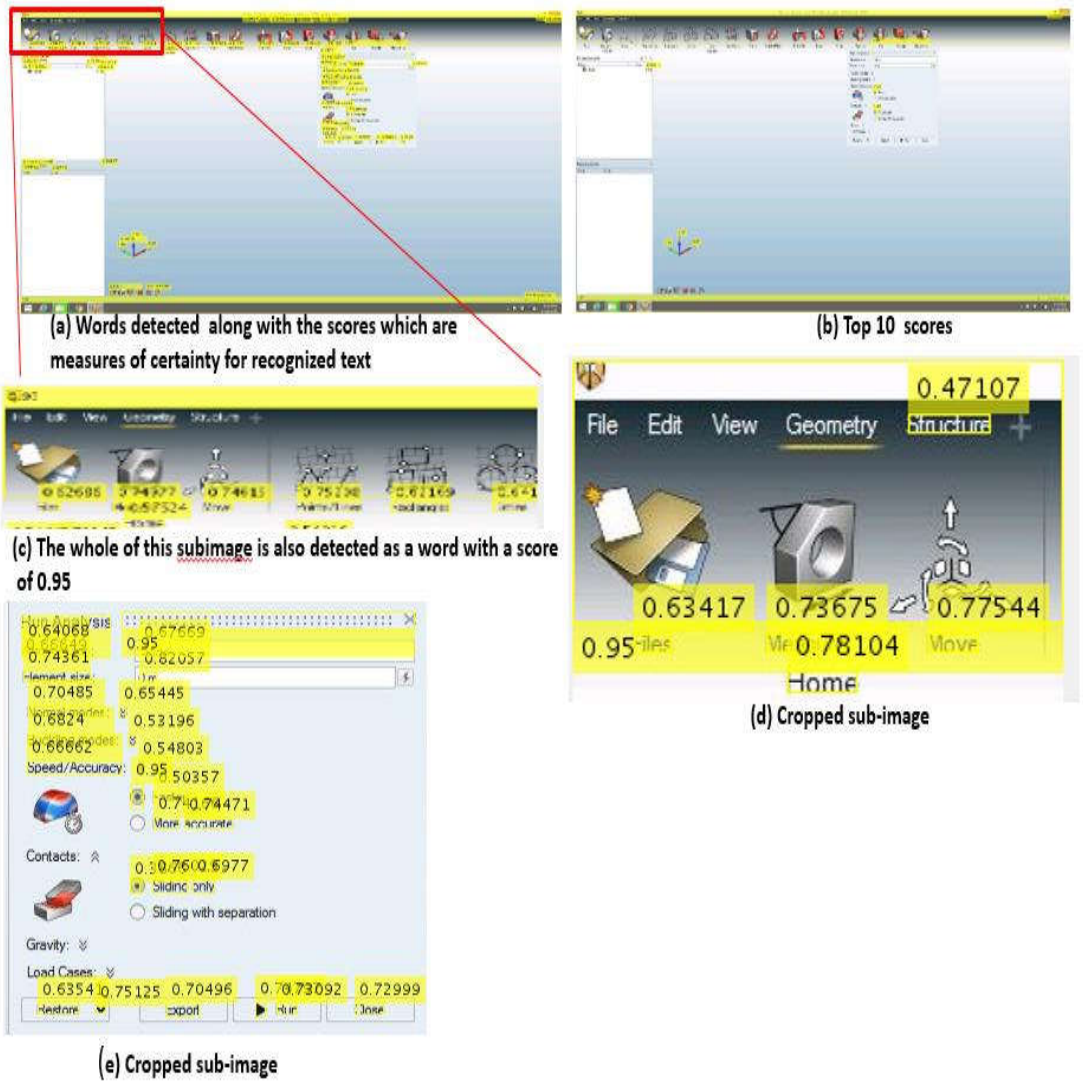

Figure 27: Optical Character Recognition performed on the detected texts.

OCR [14], just like our other experimented methods, has certain flaws as it outputs false positives with indeed higher scores as evident in Figures 31 (a)-(e) and true positives are also sometimes missed out (Figure 31 (d)).

Implementation: ASP .NET
Method 1: Using Microsoft Office Document Imaging model

Figure 28: Microsoft Office Document Imaging model

In examples 2 and 3 in Figure 32, OCR fails on the full images and hence tested on patches of the full images

8. Machine Learning Techniques for Text Detection

- **ADA**ptive **BOOST**ing technique(ADABOOST) [15]
- SVM [16]
- LIBSVM [17]
- LUKAS NEUMANN CODE [18]
- CNN-SVM [19]

8.1 Training dataset for text detection

Chars74K dataset

The Chars74k dataset [20] consists of:

- 64 classes (0-9, A-Z, a-z)

- 7705 characters obtained from natural images
- 3410 hand drawn characters using a tablet PC
- 62992 synthesised characters from computer fonts

This gives a total of over 74K images (which explains the name of the dataset).

8.2 Preprocessing Training/Testing Dataset

Almost 95% of the character images of the Chars74k character database [20] is pre-processed to tightly fit into a rectangular contour, so as to best resemble the generated character region proposals. The rest 5% of the character images displays adverse effects on outline fitting and hence were excluded from the pre-processing task. Few instances of the prepared text character images from Chars74k database are presented in Figure 33.

Figure 29: Preprocessed training Chars74k dataset.

8.3 ADABOOST

Adaboost [15] relies on a ensemble of weak classifiers to form a strong classifier where each successive classifier is trained only on those selected samples which pass through the preceding classifiers.

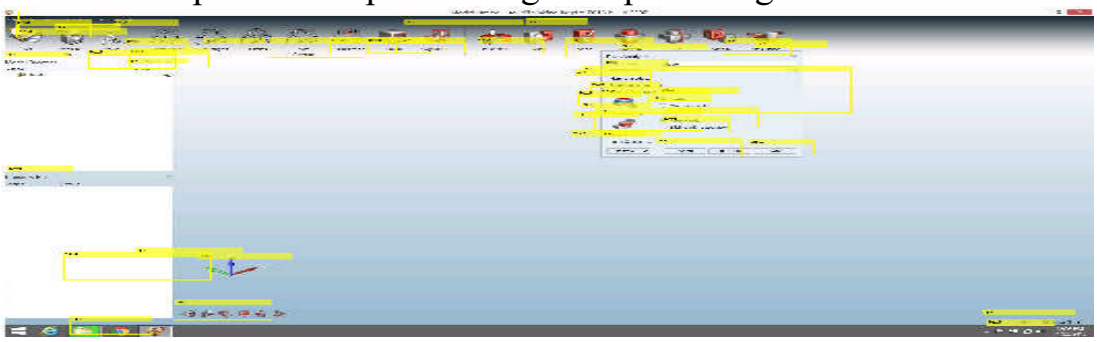

Figure 30: Text detection results using ADABOOST.

Adaboost [15] did not yield much satisfactory results with text though it works well with faces, fences, etc.

8.4 SVM

"Support Vector Machine" (SVM) [16] is a supervised machine learning algorithm where each data item is plotted as a point in n-

dimensional space (where n is number of features) with the value of each feature being the value of a particular coordinate. Then a hyper-plane is obtained that separates the two classes by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Color Correlogram as features

Figure 31: Test images for feature extraction.

Color autocorrelogram plots: ci (i=1 to 8) denotes color number and the lines are plotted using the same color as in color map of the image. Color autocorrelogram corresponding to the 4 images are plotted to see how the spatial correlation of pairs of same colors changes with distance and whether they can be used as a discriminating criteria. Yaxis denotes the probability of finding a same color pixel at the specified distance.

Distance vector= [1 3 5 7] pixels.
Image colors are quantized to 8 colors.

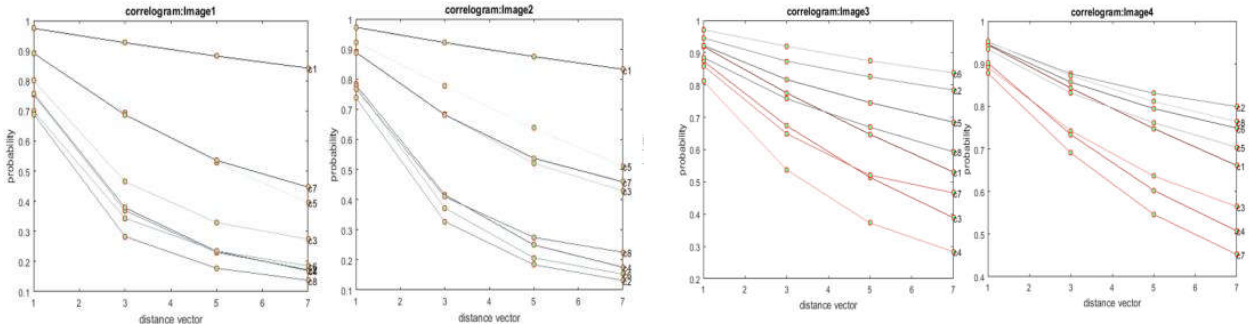

Figure 32: Correlogram of the four test images.

Though it looks like a possible feature extractor in these cases as it had given desirable results, but it could not be expected to work very well in all the cases.

Histogram of images as features

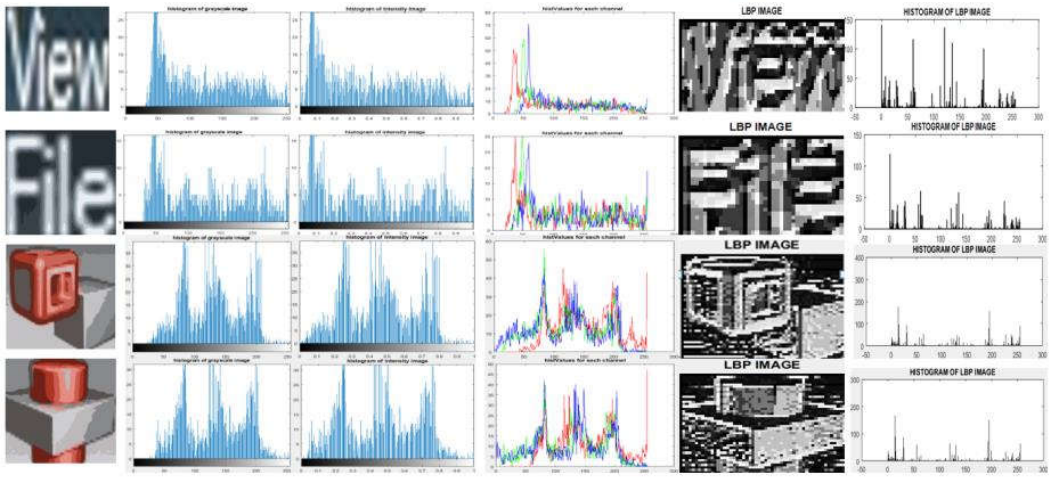

Figure 33: Study of patterns of histograms of text and non text images

Local Binary Pattern (LBP) is an efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

Histograms comparison
Method 0: Correlation
Method 1: Chi-Square
Method 2: Intersection
Method 3: Bhattacharya

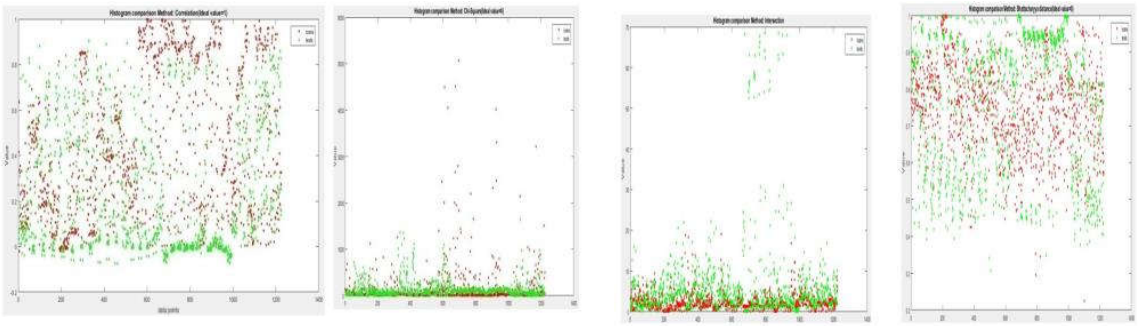

Figure 34: Histogram comparison using four methods: Correlation, Chi-Square, Intersection, Bhattacharyya distance.

No threshold could be determined from any of the plots as data and icons seem to be inseparable.

MATRIX PLOT

Histogram of 50 icon images are compared among one another.

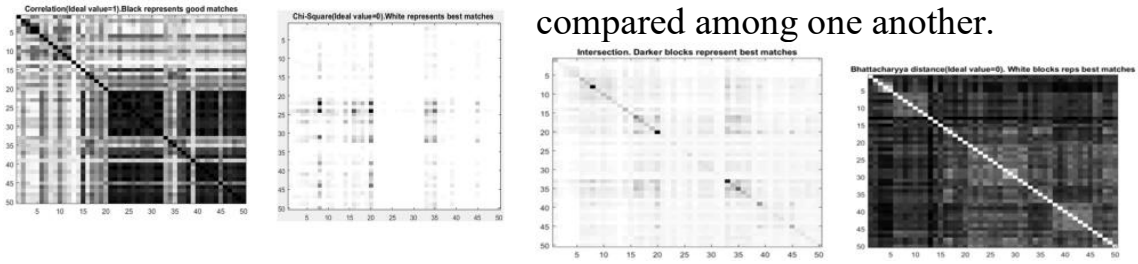

Figure 35: Histogram of 50 icon images.

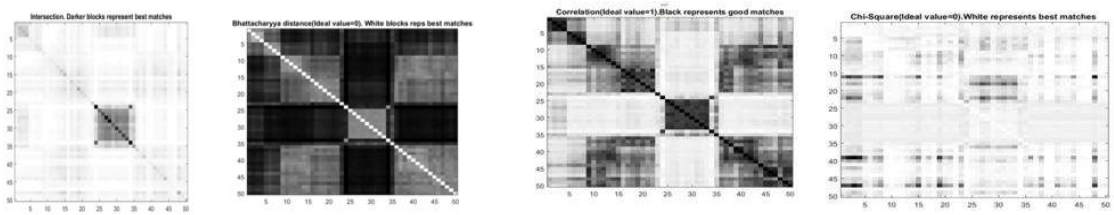

Figure 36:: Histogram of 50 icon images.

Thus Bhattacharyya methods for histograms comparison yields better results.

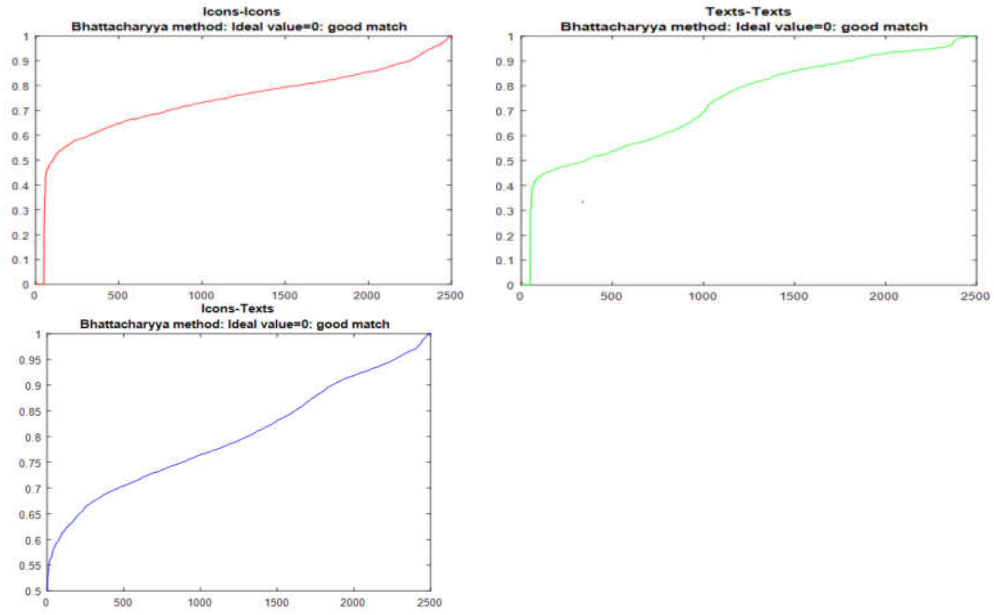

Figure 37: Histogram comparison using Bhattacharya method

More than 80% of pair of text and non-text images are having error values of >.7(close to 1, which was as expected). So a threshold of 0.7 could be set but it accounts to approximately 64% and 52% of comparison values of icons-icons and text-text cases which is not desirable. Hence histogram as a feature vector would be efficient only if the texts are of similar type and so are the icons.
We still went on to test the SVM with histograms.

SVM results: LOSS PERCENTAGE=66%
Due to high loss percentage, we then considered experimenting the same dataset and same histogram features using libsvm.

8.5 LIBSVM

Features: Histogram

Training:
Total Number of positive images (Text)=83
Total Number of negative images (Non-Text/icons)=83

Testing:
Total Number of positive images (Text)=176
Total Number of negative images (Non-Text/icons)=120

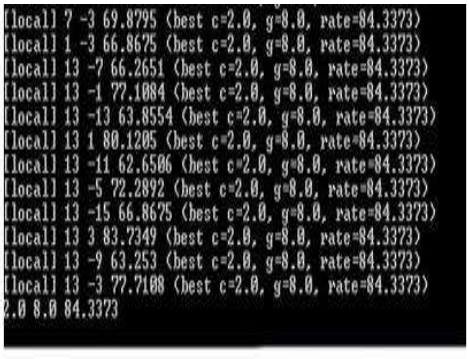

Figure 38: Selection of Best Tuning parameters.

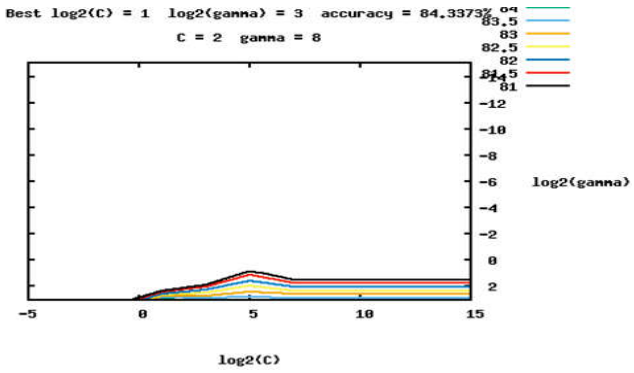

Figure 39: Libsvm testing Accuracy

Libsvm results

Table 1: Table shows the Libsvm results obtained using a RBF kernel.

kernel	N fold cross validation (N) (input)	C (output)	G(output)	Parameter source	RATE(output)	Testing accuracy
RBF	3	8	8	Grid.py	84.3373	87.3378
RBF	5(default)	2	8	Grid.py	84.3373	89.8649
RBF	10	8	8	Grid.py	84.9398	
RBF	15	8	8	Grid.py	85.5422	
RBF	20	8	8	Grid.py	84.9398	
RBF	25	8	8	Grid.py	83.7349	
RBF	unknown	2	2	Easy.py	84.3373	83.7838
LINEAR	5(default)	0.5	2	Grid.py	66.2651	65.2027

8.6 IMPLEMENTATION OF LUKAS NEUMANN CODE

Detection algorithm fails for the full image. Hence the text detection on cropped patches from the original image.

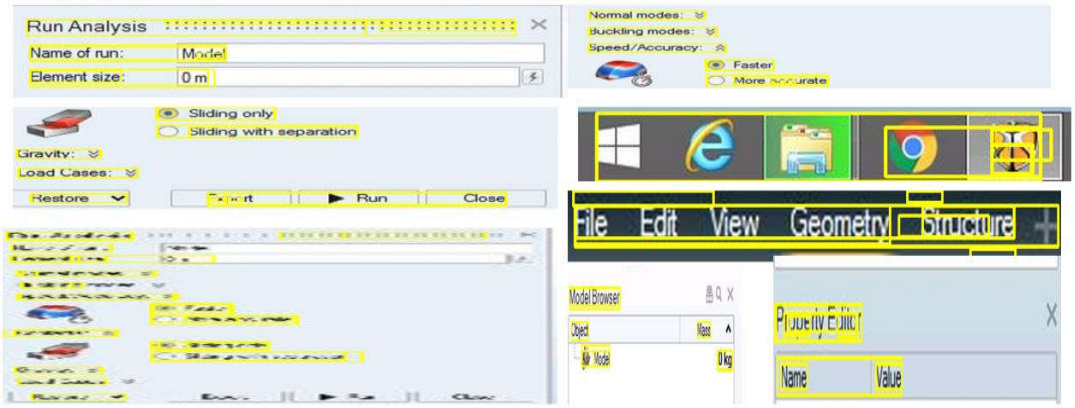

Figure 40: Text detection results obtained using Lukas Neumann Code

Lukas Neumann text detection code [18] implemented using C++ and OpenCV, is the state-of-the-art and its implementation itself needs certain pre-requisites and building of extra OpenCV libraries. Again the trained model had been published as an xml file without clearly specifying the method used for training and the contents of those xml files are also difficult to decipher. So tempering with the code would not be an easy task and might change the accuracy of the model (for better or for worst). However at this stage we would not completely rule out the feature extraction algorithm (mser detection) that had been used.

8.7 CNN-SVM

A convolutional neural network is a type of feed-forward artificial neural network comprising of various combinations of convolutional and sub-sampling layers optionally followed by fully connected layers. CNNs are efficient at learning invariant features, but do not always produce optimal classification results. Conversely, SVMs with their fixed kernel function cannot learn complicated invariances, but produce good decision surfaces. Hence CNN and SVM are combined by replacing the fully connected network of CNN by SVM that produces better classification. CNN-SVM [19] has produced higher accuracy than SVM or LIBSVM with the same dataset.

Dataset: 203 (icons) + 203 (texts) = 406

Table 2: Accuracies obtained for the original and the resized images.

	ACCURACY	
TESTING IMAGE S %	ORIGINAL IMAGES	RESIZED IMAGE
90	96.01	95.38
80	97.04	96.26
70	97.32	96.12
60	97.54	97.2
50	97.821	97.72
40	97.45	97.33
30	97.36	98
20	98.78	98.42
10	97.5	99.5

To avoid over-training, we have decided to take only 30% of the data for training and the rest were used for testing.
False detections from Lukas Neumann code are given as inputs to CNN-SVM [19] text Detector and the results are as shown below.

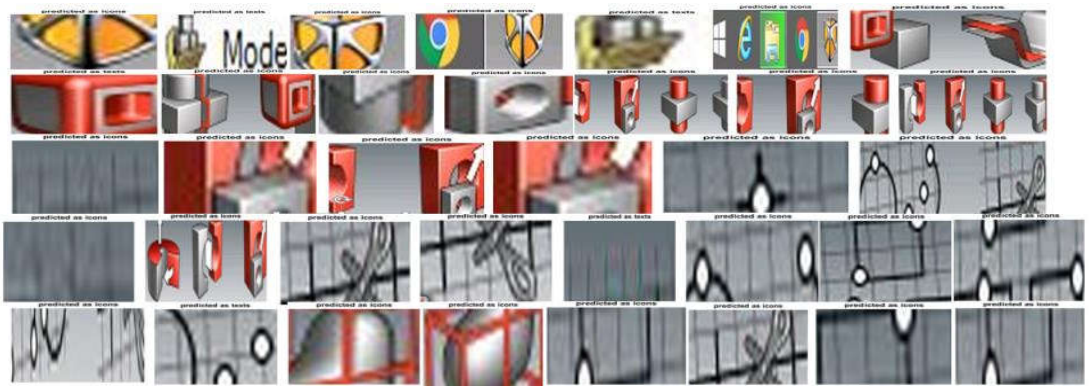

Figure 41: Results of CNN-SVM detector after giving false detections to Lukas Neumann code.

9. Research problems formulated and solutions

Text detection is a pattern recognition application which is very useful in advertising, indoor localization in malls, analyzing business and administrative documents, corporate website access, and related applications. However, it still continues to pose challenges because of the extensively varying types and patterns of text. The detection problem is further exacerbated when dealing with unstructured scenes replete with huge sources of variation and inherent complexities. To address this challenging problem, we propose to explore maximally stable extremal region (MSER) based text detection and recognition using convolutional neural networks (CNN). Prior segmentation of text regions in natural images is an objective in our proposed text detection method. It also facilitates the additional task of optical character recognition (OCR) which recognizes the text detected in the image. The work on text detection had also been extended to address the problem of logo recognition in natural scenes by first detecting and suppressing text of varying color, font size and orientation in the natural images containing logos. Consequently, it was followed by clustering of remaining stable extremal regions (ERs) forming logo region proposals based on spatial proximity and subsequent logo detection and recognition using deep learning.

Additionally, we had also explored Stroke Width Transform (SWT) and foreground object detections algorithms to tackle the problem of text detection.

10. Further work required, if any, to get full benefits or enhancement

Nil

11. Output

11.1 Technology Development

Technology developed at Lab scale/pilot scale/commercial scale

Lab scale.

Technology demonstration in field setup

No.

Technology Transfer to industry

Yes.

11.2 Participation in conference/ training workshops attended

Medhi, M., Sinha, S., Sahay, R.R. (2017). A Text Recognition Augmented Deep Learning Approach for Logo Identification. In: International Conference on Computer Vision, Graphics, and Image Processing. ICVGIP 2016. Lecture Notes in Computer Science(), vol 10481. Springer, Cham. https://doi.org/10.1007/978-3-319-68124-5_13.

11.3 Temporary manpower recruited

1 Research Consultant.

11.4 Achievements of the Project

We have explored both hand-crafted and deep learning based feature extraction techniques to detect and identify text in images. We have conducted research on several text detection algorithms to analyze the performances of the different algorithms. Herein, we have successfully addressed the problem of text detection and we have also delivered a standalone application for text detection.

Signature of PI
Name of PI: Dr. Rajiv Ranjan Sahay
Name of the participating Organization: IIT, Kharagpur

References

[1] Donoser, M., & Bischof, H. (2006, June). Efficient maximally stable extremal region (MSER) tracking. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)* (Vol. 1, pp. 553-560). Ieee.

[2] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[3] Werner, G. (2013). Text Detection in Natural Scenes with Stroke Width Transform. Ben Gurion University.

[4] Schäling, B. (2011). The boost C++ libraries. Boris Schäling.

[5] Siek, J. G., Lee, L. Q., & Lumsdaine, A. (2001). The Boost Graph Library: User Guide and Reference Manual, The. Pearson Education.

[6] He, L., Ren, X., Gao, Q., Zhao, X., Yao, B., & Chao, Y. (2017). The connected-component labeling problem: A review of state-of-the-art algorithms. Pattern Recognition, 70, 25-43.

[7] Mossi, J. M., & Albiol, A. (1999). Improving detection of clustered microcalcifications using morphological connected operators.

[8] Halkiotis, S., Botsis, T., & Rangoussi, M. (2007). Automatic detection of clustered microcalcifications in digital mammograms using mathematical morphology and neural networks. Signal Processing, 87(7), 1559-1568.

[9] Quintanilla-Domínguez, J., Ojeda-Magaña, B., Cortina-Januchs, M. G., Ruelas, R., Vega-Corona, A., & Andina, D. (2011). Image segmentation by fuzzy and possibilistic clustering algorithms for the identification of microcalcifications. Scientia Iranica, 18(3), 580-589.

[10] Anand, S., & Rathna, R. A. V. (2013, March). Detection of architectural distortion in mammogram images using contourlet transform. In 2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN) (pp. 177-180). IEEE.

[11] Shih, F. Y., & Gaddipati, V. (2003). General sweep mathematical morphology. Pattern recognition, 36(7), 1489-1500.

[12] Van de Weijer, J., & Van den Boomgaard, R. (2001, December). Local mode filtering. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001 (Vol. 2, pp. II-II). IEEE.

[13] Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

[14] Eikvil, L. (1993). Optical character recognition. citeseer. ist. psu. edu/142042. html, 26.

[15] Schapire, R. E. (2013). Explaining adaboost. Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik, 37-52.

[16] Xue, H., Yang, Q., & Chen, S. (2009). SVM: Support vector machines. In The top ten algorithms in data mining (pp. 51-74). Chapman and Hall/CRC.

[17] Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2(3), 1-27.

[18] Neumann, L., & Matas, J. (2013). Scene text localization and recognition with oriented stroke detection. In Proceedings of the ieee international conference on computer vision (pp. 97-104).

[19] Niu, X. X., & Suen, C. Y. (2012). A novel hybrid CNN–SVM classifier for recognizing handwritten digits. Pattern Recognition, 45(4), 1318-1325.

[20] De Campos, T. E., Babu, B. R., & Varma, M. (2009). Character recognition in natural images. VISAPP (2), 7(2).