Multilingual Text Inversion Detection using Shape Context

Anand P V[1] **Karthik K**[2]
Innovation Incubator Advisory Pvt. Ltd.
Technopark, Thiruvananthapuram, India
VP – AI & Data Science[1] AI Engineer[2]
anand.pandithar@gmail.com[1] k.karthik@ieee.org[2]

Abstract

It is common to have aberrations in manually scanned textual documents. Document inversion is among the most frequent but a harder anomaly to detect efficiently. Moreover, an algorithm that may detect text inversion in one language may not work for another language. Deep Learning can be a language-agnostic solution, but they are not the most efficient. In this paper, we present an inversion detection algorithm based on shape context, which is a mathematical descriptor that uses log-polar histograms to encode relative shape information. Furthermore, to localize text blocks inside images, an efficient text bounding box algorithm has been proposed. The end-to-end algorithmic pipeline can localize text and detect inversion in multi-lingual text documents. The experiments demonstrate the method to have around 17.5x speed improvement vis-a-vis a standard Deep Learning model, with near 100% accuracy on the test dataset.

Keywords: Multi-Lingual Documents; Inversion Detection; Text Detection; Text Bounding Box

I. INTRODUCTION

More efficient and deterministic ways to solve a problem are often preferred in the field of computer science. Deep Learning, being so powerful, can be employed to solve many complex tasks. However, it is prudent to use efficient traditional methods, if comparable accuracy can be obtained. Many of the algorithms "running on the Edge", use computer vision and traditional mathematics, rather than compute and memory-intensive neural nets. Even otherwise, it is beneficial to minimize the running time and resource usage, on traditional workstations and cloud instances.

In this paper, scanned images of textual documents are taken as input. The aim is to detect whether the documents are inadvertently scanned bottom-up. Seemingly simple, as it may sound, this is a daunting task for a computer, which represents images and alphabets, as a 2-D or 3-D matrix of numbers. Standard Deep Learning models can be used to do a binary classification of upright vs. inverted images. Instead, we have used pure arithmetic operations to represent the shape of characters, so that inversion of shapes can be detected by matching shape histograms. To complete the pipeline, the Text Bounding Box Detection algorithm is proposed in Section III-B to localize the text from images and feed it into the Inversion Detection Algorithm proposed in Section III-D.

II. MANUAL SCANNED DOCUMENTS

A. Anomalies in Scanned Images

Image can become inverted when a document is scanned upside-down. During the manual scan, other discrepancies like skew or rotation also can naturally creep in. We can detect the rotation angle by drawing a minimum area bounded rectangle around all identified pixels so that it considers the rotation of the document inherently. If lines are present in the document, then they may become linear point clouds in the scanned image. It is possible to identify the document skew angle by analyzing the point cloud to find the best fit line, using Hough Transformation [1] and Extended Hough Transformation [2].

While it is feasible to do skew correction and rotation correction, the same idea may not work in the case of inverted images as document angle, θ becomes zero. Moreover, the document can unintentionally become inverted, after rotation or skew correction, as a rotation of $90 + \theta$ can be detected as $-\theta$, or $-90 - \theta$ as $+\theta$. Thus, it is much harder to detect the inversion of documents or images containing text, than other anomalies.

B. Related Work

The filling capacity of an upright and downside character can be visually deciphered to be different. If we can find a metric to measure the filling capacity of characters, then it becomes possible to understand whether the document is inverted or not. Such a metric, known as 'Water Fill Technique' is given in this paper [3]. The sum of water fill capacity of the upright and inverted characters in the scanned page is computed and the image with lower capacity is classified to be inverted. However, this method may not work in most non-English languages.

Another method is known as 'Double Peaks', projects all the pixels onto the y-axis. Thus, each line in the document would result in 2 peaks, due to the shape of small English characters. If more sub-peaks are found on the right side of each peak corresponding to a line, then the document is upright. But the "double peak logic" would falter when the text is in capital letters or is in a non-English language. In this paper, we discuss an end-to-end pipeline to automatically localize the alphabets and words from multi-lingual document images and analyze them using a mathematical shape descriptor to detect document inversion in scanned images.

The source code and dataset are available at:
https://github.com/AdroitAnandAI/Multilingual-Text-Inversion-Detection-of-Scanned-Images

III. Text Inversion Detection

To detect inversion, the location of text in the image needs to be localized. The cropped image patch inside the localized bounding box is fed into the 'Inversion Detection Algorithm' in Section III-D.

A. *Text Bounding Box Detection*

The textual content inside a natural scenic image can be localized using the EAST (An Efficient and Accurate Scene Text Detector) or CRAFT (Character Region Awareness for Text Detection) method [4, 5]. EAST is a fully-convolutional neural network model adapted for text detection. CRAFT is a deep learning based character-level text detection technique, that explores the affinity between characters to detect text area. However, it is possible to detect characters, words and text area more efficiently, as we deal with only textual documents here. Experiments demonstrate logarithmic improvement in efficiency when an arithmetic method is used to detect text [Fig.1].

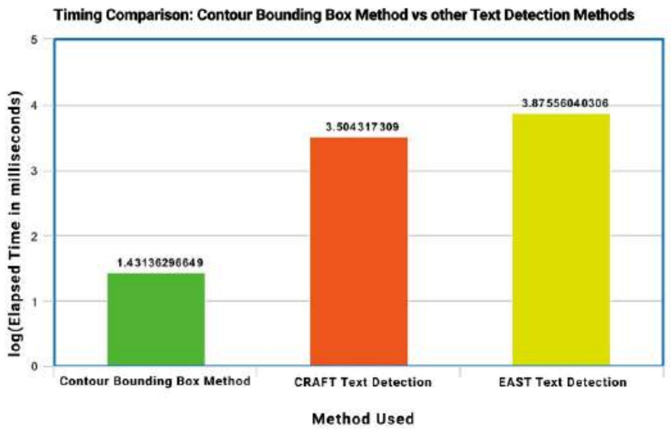


Fig. 1. Time: EAST (7.51s), CRAFT (3.19s),Contour Bounding Box (0.027s)

To identify the vertical boundaries of lines in an image, you can project the pixels in the image along the y-axis and find consecutive troughs. The consecutive troughs would correspond to line separation, because of an absence of pixels along the vertical gap between lines. However, this method may not work when the image contains noise, as it may result in unexpected spikes along the curve troughs. More efficient and stable results are found possible by using the algorithm in Section III-B, to localize text from images, based on dilation and contour analysis.

B. *Bounding Box Detection Algorithm*

1. *Invert the image and threshold to get white alphabets in black background*
2. *Remove the lines using Hough Transformation* [1], *to minimize interference in subsequent steps*
3. *Find 'C' = number of contours*
4. *Iterate Kernel Size, k from 3 to N, where N > 3*
   - *Grow the size of foreground object using Morphological Dilation* [6] *and Structuring Element with Kernel Size, kxk*
   - *Plot Contour Graph, i.e. the Number of Contours Vs Kernel Size. The frequency of contours will take a sudden dip when characters get merged into words*
   - *Fit an inverse sigmoid curve at the tail-end of the Contour Graph, obtained above* [7]
   - *If successful fit is found, then declare 'word-merge' detection and break the iteration*
5. *Find the contours around the image, dilated with kernel size identified to merge words*
6. *Draw Bounding Rectangle around the identified contours to get "word" bounding boxes*
7. *Merge overlapped bounding boxes identified with IOU (Intersection over Union) metric, using Non-Maximal Suppression* [8] *used in the object detection pipeline.*

In a scanned image, the alphabets of every word would be spatially nearby. Hence, dilation with bigger kernels has a higher chance to merge the alphabets in a word, as a single object. The number of contours before dilation would represent the number of characters in the image. Hence, the number of contours will suddenly decrease, when the characters get merged into words.

Imagine an 'S' shaped curve, mathematically parameterized by a sigmoid function [9]. Such a curve signifies a sudden hike in the y-axis value [Fig.2].

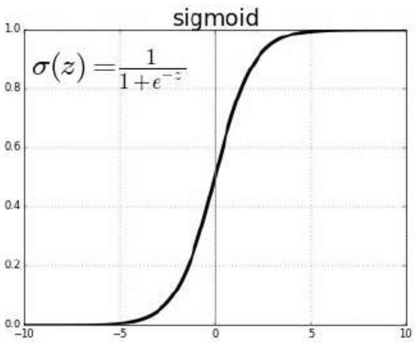


Fig. 2. Sigmoid-shaped curve signifies gradual increase and plateau of values

Consider "Kernel Size" along the x-axis and 'Number of Contours' along the y-axis to plot the 'Contour Graph'. When the number of contours goes down, the shape of the curve will take the form of an inverse sigmoid function. Find f (-x) to flip the sigmoid function about the x-axis.

$$f(-x) = \frac{1}{1+e^{z}} \tag{1}$$

Try to fit an inverse sigmoid function at the tail end of the curve to detect the fall in the number of contours [10]. To detect word merge, the parametric curve fit algorithm tries to solve a nonlinear least-squares problem to fit shape of Equation (1) to the curve. For the curve in [Fig.3], the kernel size to merge characters is found as 11x11.

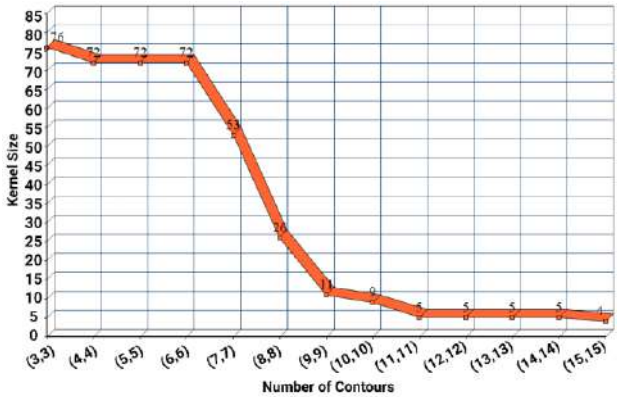


Fig. 3. Contours vs Kernel Size: Inverse Sigmoid Curve for English Text. Kernel Size is found 6~12 for English Text, and 25~30 for Malayalam Text.

Empirically, the kernel size varies significantly across different languages and fonts. Once kernel size, kxk is found, take it as a constant for images from the same document or book. Even if more words get merged due to variable proximity of words, the 'Inversion Detection Algorithm' in Section D will work.

C. Shape Context using Log-Bin Histogram

Textual characters in an image can be considered as objects of different shapes, formed by pixels. We can represent the shape of each character with the help of a shape descriptor known as 'Shape Context' [11]. Log-polar histogram bins are used to compute and compare shape contexts [Fig.4].

The log-polar histogram captures the angle and distance to randomly sampled (n-1) points of a shape, measured from the reference point. Thus, the log-bin histogram of similar shapes tends to be near. As we increase the sampling density of the edge points of a shape, the representation becomes increasingly accurate [11].

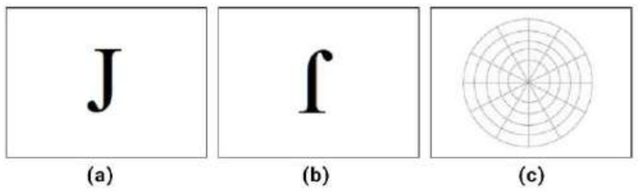


Fig. 4. **(a)** Upright 'J' Shape **(b)** Inverted 'J' **(c)** Log-Polar histogram bins

To identify an alphabet, find the pointwise correspondences between edges of an alphabet shape and stored base images alphabets. The base image needs to contain all the alpha-numerals of a language or even special characters, that may be present in the document. As the shape of each alphabet, numeral, or special character is different, we can identify the character from image patches [Fig.5].

To measure shape similarity, match the corresponding log-bin histograms using Pearson's chi-squared test [12]. To identify the alphabet in an image patch, measure the similarity of each alphabet with all the shapes found in the base image and minimize the cost. The algorithm to detect inversion using the sum of minimum match cost is given in Section D.

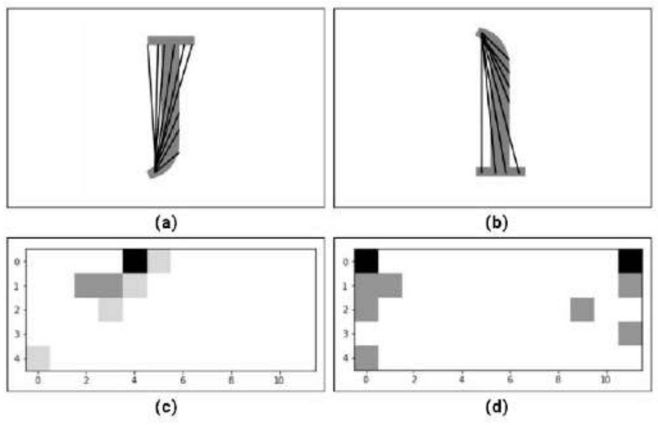


Fig. 5. **(a)** and **(b)** (n-1) vectors from a reference point from shape 'J' and inverted 'J' **(c)** Log-Polar histogram visualization of Upright & Inverted 'J'

To find the total match cost of the text inside the bounding box, sum up the minimum histogram match cost of each shape inside the bounding box with each alpha-numeral in the base image. The idea can be mathematically represented by,

$$\sum_{i=1}^{n} \min_{j=1\,to\,m} C(p_i, p_j) \tag{2}$$

where m = Total number of characters in base image,

n = Number of characters in Text Bounding Box,

p_i & p_j are corresponding points on the first and second shape respectively. The cost of matching log-bin histograms of corresponding points, C (p_i, p_j) denoted by $C_{i,j}$ is computed by Pearson's χ^2 test statistic [12].

$$C(p_i, p_j) = \frac{1}{2}\sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)} \tag{3}$$

where h_i (k) & h_j (k) represent K-bin normalized histogram at p_i and p_j respectively. Alternatively, cosine distance can be used to compare histograms efficiently. To find point-to-point correspondence between shapes, we can solve the linear sum assignment problem using the Hungarian algorithm [13].

Formally, let matrix C be the cost matrix, where C [i, j] is the cost of matching point p_i with p_j and let X be a Boolean matrix where X [i, j] = 1 when row 'i' is assigned to column 'j'. The optimal one-to-one assignment cost can be computed by,

$$\min \sum_i \sum_j C_{i,j} X_{i,j} \tag{4}$$

The method will work for multiple languages also, as the idea hinges on the concept of shape. To identify a document in another language, store and compare with the base alphabets of the corresponding language. The applicability of the method for multi-lingual documents is demonstrated in Section E.

D. Inversion Detection Algorithm

The base image needs to contain all the possible characters in the document. It will ideally contain small and upper case alphabets of the language, along with numerals 0-9.

a) Use Bounding Box Detection Algorithm to find the bounding box with 'maximum width' on one-page.

b) Crop the image inside the bounding box and apply Canny edge detection [14].

c) Find bounding boxes around each character in the base image and image from step (b)

d) Pick N points at random, from the edge points of each character shape.

e) Construct the mathematical descriptor — shape context — for each character.

f) Compare the log-polar histograms using Pearson's chi-squared test. Instead, use cosine distance to improve the comparison speed.

g) Find the alphabet with minimum matching cost, among all characters inside the base image.

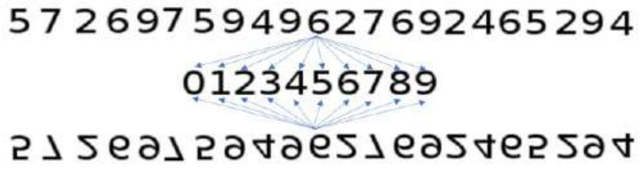


Fig. 6. Upright & Inverted numbers against an all-numeral base image.

h) Compute β = Sum of minimum matching cost of each character bounding box in input image containing text.

i) Flip the cropped from step (b) and iterate Step (d)-(h) to compute β'

j) If $\beta < \beta'$,

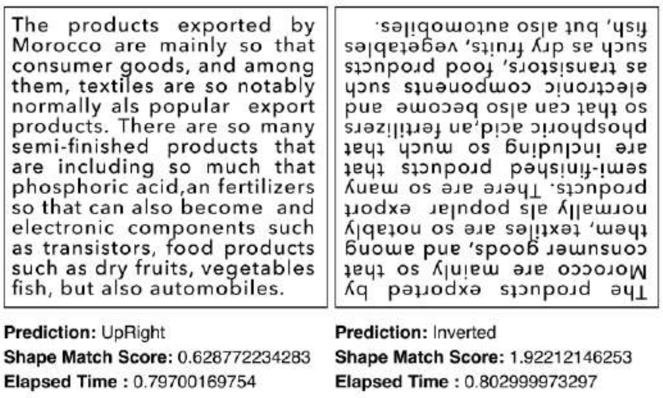
then Input Document is Upright

else Input Document is Inverted

E. Multi-Lingual Documents

The 'shape context' (SC) as an idea, describes the concept of shape mathematically. The algorithm gives distinguishable 'Shape Match Score' on upright and inverted documents in various languages. Hence, the matching based on SC would work on multiple languages as demonstrated below [Fig.7-9].

The text bounding box detection is also a language-agnostic algorithm, as dilation-based character merge depends only on pixel proximity. However, the threshold value to be used to classify inversion in various languages may be different. Interestingly, we can extend the same idea to identify the language from images.

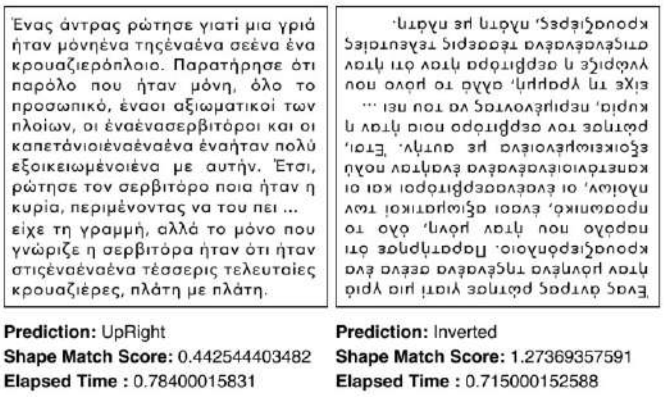


The products exported by Morocco are mainly so that consumer goods, and among them, textiles are so notably normally als popular export products. There are so many semi-finished products that are including so much that phosphoric acid,an fertilizers so that can also become and electronic components such as transistors, food products such as dry fruits, vegetables fish, but also automobiles.

Prediction: UpRight
Shape Match Score: 0.628772234283
Elapsed Time : 0.79700169754

Prediction: Inverted
Shape Match Score: 1.92212146253
Elapsed Time : 0.802999973297

Fig. 7. Inversion Detection on English Language text images.

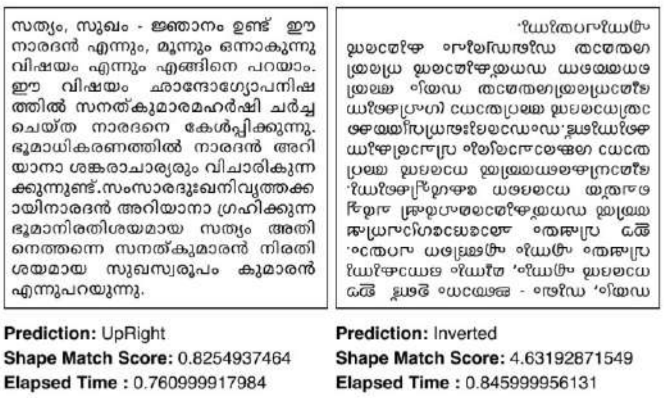


Ένας άντρας ρώτησε γιατί μια γριά ήταν μόνηένα τηςέναένα σεένα ένα κρουαζιερόπλοιο. Παρατήρησε ότι παρόλο που ήταν μόνη, όλο το προσωπικό, έναοι αξιωματικοί των πλοίων, οι ένaένασερβιτόροι και οι καπετάνιοιέναένaένα έναήταν πολύ εξοικειωμένοιένα με αυτήν. Έτσι, ρώτησε τον σερβιτόρο ποια ήταν η κυρία, περιμένοντας να του πει ... είχε τη γραμμή, αλλά το μόνο που γνώριζε η σερβιτόρα ήταν ότι ήταν στιςέναένaένα τέσσερις τελευταίες κρουαζιέρες, πλάτη με πλάτη.

Prediction: UpRight
Shape Match Score: 0.442544403482
Elapsed Time : 0.78400015831

Prediction: Inverted
Shape Match Score: 1.27369357591
Elapsed Time : 0.715000152588

Fig. 8. Inversion Detection on Greek Language text images.



സത്യം, സുഖം - ജ്ഞാനം ഉണ്ട് ഈ നാരദൻ എന്നും, മൂന്നും ഒന്നാകുന്നു വിഷയം എന്നും എങ്ങിനെ പറയാം. ഈ വിഷയം ഛാന്ദോഗ്യോപനിഷ ത്തിൽ സനത്കുമാരമഹർഷി ചർച്ച ചെയ്ത നാരദനെ കേൾപ്പിക്കുന്നു. ഭൂമാധികരണത്തിൽ നാരദൻ അറി യാനാ ശങ്കരാചാര്യരും വിചാരികുന്ന ക്കുന്നുണ്ട്.സംസാരദുഃഖനിവൃത്തക്ക ായിനാരദൻ അറിയാനാ ഗ്രഹിക്കുന്ന ഭൂമാനിരതിശയമായ സത്യം അതി നെത്തന്നെ സനത്കുമാരൻ നിരതി ശയമായ സുഖസ്വരൂപം കുമാരൻ എന്നുപറയുന്നു.

Prediction: UpRight
Shape Match Score: 0.8254937464
Elapsed Time : 0.760999917984

Prediction: Inverted
Shape Match Score: 4.63192871549
Elapsed Time : 0.845999956131

Fig. 9. Inversion Detection on Malayalam (Indian Language) text images.

F. Language Identification

Each language needs to have a separate base image, that contains all the potential characters of the corresponding language. The minimum matching cost of all characters inside the detected bounding box (using the algorithm in Section B) is added up to find the base image that minimizes the total cost.

The idea can be mathematically represented as,

$$L^* = \underset{l=1 \text{ to } L}{\operatorname{argmin}} \sum_{i=1}^{n} \min_{j=1 \text{ to } m} C(p_i, p_{l,j}) \quad (5)$$

where L = Number of languages possible,

m, n = characters in base & bounding box image,

$p_{l,j}$ = point in jth character of lth language,

L* = Detected Language.

G. *Limitations and Boundary Cases*

The text bounding box with maximum width in the image is fed into the 'Inversion Detection Algorithm'. If the selected bounding box contains only inverted text, then the entire image would be classified as inverted.

If the selected bounding box contains only those characters which look the same after inversion, then the algorithm may not work. To explicate, the capital letters such as "I", "C", "O", "D", "K" etc. are visually similar, post-inversion.

When such documents are expected, more bounding boxes need to be analyzed for inversion detection. Further, the text bounding box algorithm is designed to work only on textual scanned images and not on natural scenes.

IV. Result & Timing Analysis

The confusion matrix for the 'Shape Context' method and a trained VGG-16 Neural Network model is shown in Fig.10, 11. The predicted label is on x-axis and the true label on y-axis.

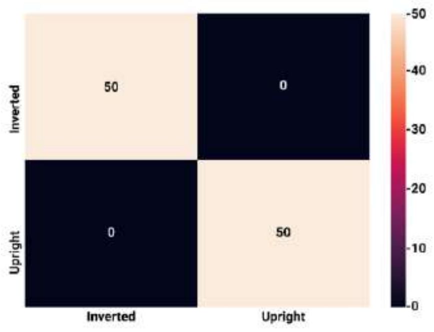


Fig. 10. Shape Context Method on a test dataset of 100 images.

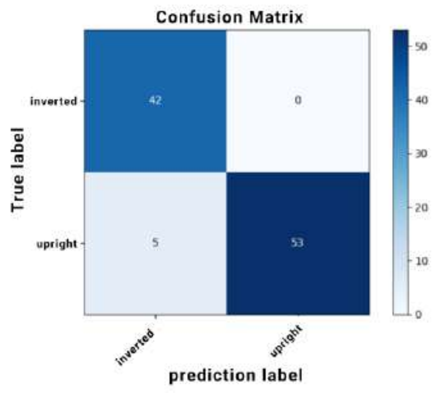


Fig. 11. Trained VGG-16 CNN Model on a test dataset of 100 images.

A. *Timing Analysis*

To enable comparison with Deep Learning model, a VGG-16 CNN network is trained with labeled upright and inverted images. The total time taken to detect text bounding boxes (with a constant kxk size) and pattern matching based on shape context is compared against VGG-16 inference time [Fig.12].

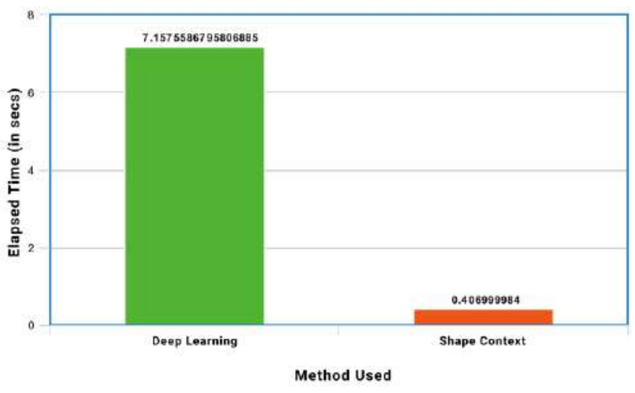


Fig. 12. VGG-16 vs Bounding Box Detection and SC Pattern Matching.

Given an image, the bounding box detection algorithm in Section III B takes 0.017 seconds to estimate kernel size, k. All the experiments were done on i7/ 16 GB/ 6GB 1660i machine.

V. Conclusion

We have proposed an end-to-end algorithmic pipeline to identify occurrences of text from a scanned image and to detect text inversion from image patches, irrespective of language. The appealing features of our approach are simplicity, speed, and language-agnosticism. Experiments demonstrate, both the algorithms, i.e. Text Bounding Box and Inversion Detection, perform comparably robust and much faster than their Deep Learning counterparts.

Acknowledgment

The authors would like to thank the organization, 'Innovation Incubator Advisory Pvt. Ltd' for the extended support to make this work possible.

References

[1] Atiquzzaman, Mohammed, and Mohammed W. Akhtar. "Complete line segment description using the Hough transform." Image and Vision computing 12.5 (1994): 267-273.

[2] Kamat, Varsha, and Subramaniam Ganesan. "A robust Hough transform technique for description of multiple line segments in an image." Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269). Vol. 1. IEEE, 1998.

[3] Pilevar, Hamid. "Inversion detection in text document images." 9th Joint International Conference on Information Sciences (JCIS-06). Atlantis Press, 2006.

[4] Zhou, Xinyu, et al. "East: an efficient and accurate scene text detector." Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2017.

[5] Baek, Youngmin, et al. "Character region awareness for text detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

[6] Ravi, S., and A. M. Khan. "Morphological operations for image processing: understanding and its applications." Proc. 2nd National Conference on VLSI, Signal processing & Communications NCVSComs-2013. 2013.

[7] Anand P.V., "The Power of Mathematical Ingenuity". Available from:
https://towardsdatascience.com/the-power-of-mathematical-ingenuity-49c7b6cfe05e [Published on Jan 1, 2020]

[8] Bodla, Navaneeth, et al. "Soft-NMS--improving object detection with one line of code." Proceedings of the IEEE international conference on computer vision. 2017.

[9] Han, Jun, and Claudio Moraga. "The influence of the sigmoid function parameters on the speed of backpropagation learning." International Workshop on Artificial Neural Networks. Springer, Berlin, Heidelberg, 1995.

[10] Anand P.V., "Touch-less Display Interfaces on Edge". Available from:
https://towardsdatascience.com/touch-less-display-interfaces-on-edge-be8dc277c5b8 [Published on Sep 21, 2020]

[11] Belongie, Serge, Jitendra Malik, and Jan Puzicha. "Shape context: A new descriptor for shape matching and object recognition." Advances in neural information processing systems 13 (2000): 831-837.

[12] Belongie, Serge, Jitendra Malik, and Jan Puzicha. "Shape matching and object recognition using shape contexts." IEEE transactions on pattern analysis and machine intelligence 24.4 (2002): 509-522.

[13] Kuhn, Harold W. "The Hungarian method for the assignment problem." Naval research logistics quarterly 2.1-2 (1955): 83-97.

[14] Canny, John. "A computational approach to edge detection." IEEE Transactions on pattern analysis and machine intelligence 6 (1986): 679-698