Ubiquitous Document Capturing with Deep Learning

Tayyaba Naz, Anam Ahmad Khan, Faisal Shafait
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology, Pakistan
Email: 14msittnaz@seecs.edu.pk

***Abstract*—Digital and paper based documents co-exist in our daily lives. Seamless integration of information from both sources is crucial for efficient knowledge management. This paper address the algorithm that can handle the detection of document so that it can be captured easily to convert it into a digital form for automatic integration of relevant information in electronic work flows. It uses the deep learning technique to provide a solution which is more generalized and flexible than other available solutions.**

***Index Terms*—Document capture, Document detection, Machine learning, Deep learning, CNN, LSD.**

I. Introduction

Modern world is heading towards digital generation, because digitization has several advantages like efficient searching, easy storage, sharing on cloud, remote access and easy backup. Thus due to these advantages, all the data being generated is in digital format. In order to cope up with this world's digitizing pace, everything currently available in non-digital form is now being converted into digital form. As documents has always been a very important means of saving and sharing data, same trend applies on them too. Documents exist in two forms: digital form and paper form. Both forms has their own advantages. Paper documents are easy to carry around and easy to read while digital documents gives us the benefit of efficient searching inside the document, easy backup, ease of management and less physical space as all of them resides in one system or on the cloud.

Integration of data is very critical nowadays due to generation of data at a rapid pace which needs to be processed. Big data is a very hot topic these days due to this explosive growth of data. If we don't have the data all together in one form, it won't be possible for others to process the data to extract out information of interest. Therefore, documents integration in one form is also very important for other research areas too.

Digitization of the paper documents by taking their image till now is being done via two ways; either by scanning the papers through scanners or by capturing them with camera. Scanning is usually done using flatbed scanners which reduces effort required to extract paper from scanned image because scanners not only automatically removes useless background part from output image but some advanced scanner also carries out the work of OCR on documents. But in the second case where documents are being captured with cameras which are mostly built into smartphones, the quality of document decreases which requires extra processing for enhancement. The extraction of document part from the captured image needs a separate algorithm. Scanning using flatbed scanners is the most used method in offices and universities but requires presence of a scanner, while the scanning of document using smartphones' camera is much more feasible for every one since smartphones are ubiquitous these days. To make this latter option much more feasible, algorithms for the image enhancement and extraction of document from the image are required. Using these techniques the usage of scanner for this digitization will reduce as not everyone owns or has easy access to a scanner and thus this scanning of documents will no more be a problem for anyone.

In order to fulfil this need of an algorithm for efficient extraction of document from the image a new methodology is being introduced in this paper which is mainly based on deep learning techniques along with a little image processing mainly LSD (Line segment detector) to extract the document from the image.

II. Literature Review

In response to the effort required to extract the document part from the image. The simplest method that has been used in the past to extract the document from the image is explained in [3] in which the background doesn't change, and the algorithm requires two pictures, one before placing the document in that environment and the other one after placing in that environment and then compares the difference between those two images to obtain document bounds. Another work has been done in this regard in the form of an ANDROID application [4] which is capable of capturing the document having any kind of background but while capturing it needs a little manual work from the user. It provides user with two rectangles a small and a large one on the screen, small one being on the inside of the larger one, and the user has to place the document boundaries in between those two rectangles. This application fails in three cases: rotated document image, very small text area and when all four margins are not visible properly. In this paper our proposed method is targeting all these three cases and with no need of manual work.

A lot of work has been done in this field and this problem has been addressed using three different methodologies: 1. Camera calibration [11], [12] 2. Document Content [13],

978-1-5386-2839-3/17/$31.00 ©2017 IEEE

[14], [15], [16], [17] 3. Geometric techniques [5], [6], [7], [8], [9], [10].

First methodology i.e. Camera Calibration, uses the information about camera's position and angle at which it is facing the document and then using this information, it extracts the boundary of document handling different perspective views of the document. On the other hand, the main drawback of this method is that the camera is dynamic in most of the scenarios and it is not possible to get the information of its position and angle at every moment whenever document is being captured.

Second methodology i.e. using Document Content, is based on the detection of document by looking out for the parts in the image which look likes text which is in a layout of a document. In [17] the background is pure black or is in the form of a very dark plain color so it is easy to just skip it of, and mostly it's focus is to extract the required document part of the image which is being captured from a book where some part of the other page is also visible. But the drawback of this methodology is that documents don't have any specific layout, their layout and text style vary a lot due to which the probability of failure of this method increases significantly.

The third methodology which has been used the most to solve this problem is based on the geometric techniques. In [5] the basic idea is to search for quadrilateral shaped object in the image which has document type properties to make sure it is a document. Another most related paper [6] works pretty good and extracts the boundary efficiently by detecting the white space first in it, then remove the text part of the document which reduces the number of edges detected, then it applies houghLines and extracts the boundary edges. But they have an assumption of paper always being white in color and rectangular in shape. With this assumption if there is something else more white in color other than paper in the image then it will fail.

This geometric methodology is the most common method used because it requires no extra information like position of camera or layout of text in the document which makes it the most generalized algorithm so far to detect the document. But as each method has some limitations, this method won't work in scenarios where the shape of document is not quadrilateral. For example, in the cases where all four edges of the documents are not visible or some part of the document is occluded or it is out of the image.

Nowadays machine learning is the new fashion as it is more adaptive and intelligent than other techniques. In order to incorporate these benefits of machine learning, deep learning is the main focus of the new method being introduced in this paper.

The reason why deep learning was elected for the solution to this problem is because it works best with unstructured media like images, sound, text to recognize patterns and classify them by extracting features on its own and we don't have to engineer features manually. Since we are dealing with images, Convolution Neural Networks is the best technique to apply on. Also as described in [18] using the MNIST example it has been shown that CNN is more suitable for classifying visual contents.

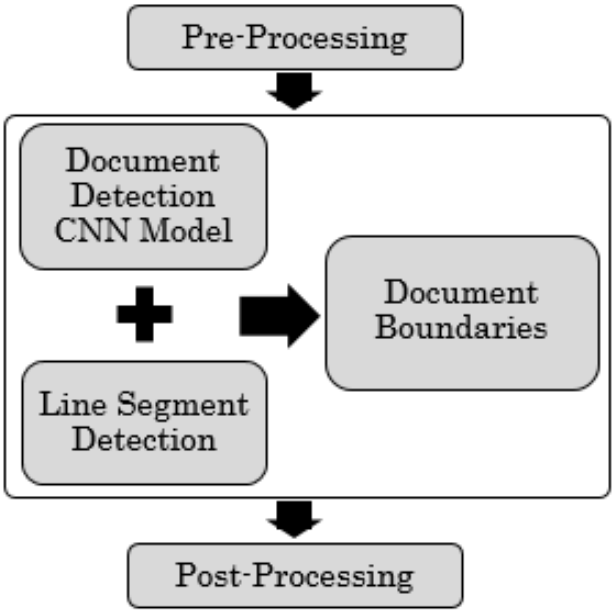

Fig. 1: Block Diagram

III. Methodology

The whole process is divided into three parts: pre-processing, main algorithm and post-processing. It has been shown in the form of a Block diagram in Figure 1.

Pre-processing and Post-processing mostly includes some basic image processing techniques for enhancement of image and refinement of some algorithms.

The main algorithm part as seen from the Figure 1 is mainly comprised of 3 blocks: Document detection with CNN model, the line segment detection, and the algorithm (based on RAST) to combine them in order to extract the document boundaries. CNN, LSD and Adaptive Subdivisions of Transformation Space (RAST) algorithm have been explained in detail in the next subsections.

A. Line Segment Detection (LSD)

LSD is one of many algorithms available for the detection of lines in an image, out of all those the most typical methods are: LSD, EDLines, Hough and Burns. In paper [1] they've compared these four line extraction methods and evaluated their performance based on the line repeatability criteria. This paper concludes that LSD algorithm performed better than all of the remaining algorithms. Out of those four algorithms we ourselves also have tested two of these algorithms; LSD and HougLines and found out that LSD gave us better results by detecting more number of lines in the given image than HoughLines as shown in Figure 2.

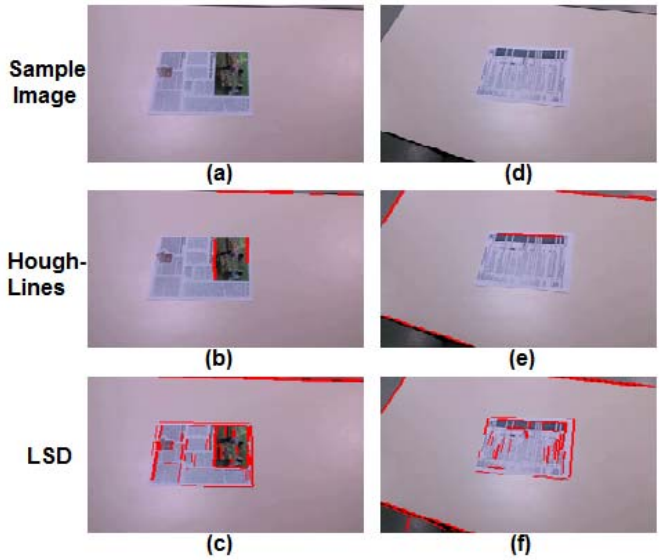

Fig. 2: Comparison of LSD and HoughLines: (a) and (d) are the sample images. (b), (e) (result of HoughLines) and (c), (f) (result of LSD) shows LSD detects more line segments than HoguhLines

Working of LSD algorithm has been explained in detail in [2].

B. Deep Learning Model Training using CNN

Deep learning is a branch of machine learning containing set of powerful techniques for learning in a multi-level artificial neural networks which helps in better understanding of data by extracting the right features from it. Thus due to it's property of extracting the right features the 'Feature Extraction' problem is till now being addressed the best by deep learning, as feature extraction is mainly the core in machine learning on which the models learn and produce results. Likewise as in our case we need to extract the features from the images. As we are dealing with images, therefore the type of neural network we used is Convolutional neural networks (CNN) because it's architecture is specifically designed for images.

Regular neural networks' architecture is composed of an input layer, output layer (classification layer) and many hidden layers, hidden layers are then comprised of many neurons and all of them are fully connected with the neurons of the layer next to them. So if we follow this same architecture for images then in case of a 200x200 pixel image, then if we take a single neuron from the first layer then it will be having 200x200x3 = 120,000 number of weights which is too large for each of the neuron to keep. And as the size of image increases more this amount will increase significantly which then will not be manageable. To solve this problem CNN architecture is designed in such a way that it's layers are not fully connected rather the size of the image keeps on decreasing as we move forward in the layers as shown in the

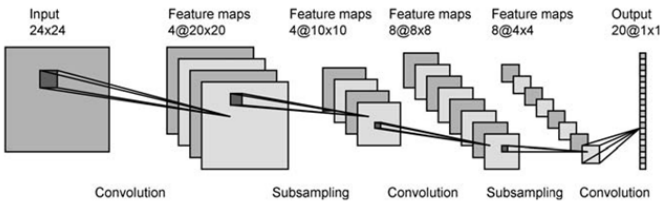

Fig. 3: CNN Architecture: Neurons in CNN are not fully connected and also the size of image reduces as it goes further deep into the layers [21]

Figure 3.

Training of a CNN model from scratch is very rare in practice because it is impractical to have such a huge amount of dataset to train the model, therefore the most common practice of training a CNN model is to use an already trained model on some huge dataset e.g. IMAGENET and then retrain it onto our own dataset to extract the features from the images. Dearth of data was also a problem in our case therefore we also went for this technique explained before which is known as 'Transfer learning'. We used pre-trained CNN model named **imagenet-caffee-alex** which is publicly available for transfer learning purposes. We used this model as a feature extractor by getting the output of second last layer as a feature vector and then applying SVM (Suppor Vector Machine) onto that feature vector to train a binary model for detection of document in the model. This model returns 1 if object is present in the input image and returns 2 if there is no document in the input image.

C. Adaptive Subdivisions of Transformation Space (RAST)

RAST is an efficient algorithm which uses no heuristic approach, it provides an optimal global solution through hierarchical searching. As the name suggests it is basically a divide and conquer algorithm which divides the main transformation space i.e. the input image and then further processes them [20].

The order in which each subdivision gets processed is done using the help of priority queues. The measure of priority in queues is calculated by the subdivision's quality, and this quality is evaluated according to the key elements of the problem. RAST's pseudo code has been published here in [20].

RAST algorithm has been explained in detail in [19] by solving a problem. In this paper they have solved the 'whitespace cover' problem, in which we have to find that whitespace rectangle in the given input image which has the maximum area and has no obstacles inside it.

RAST algorithm idea is based on voting and clustering like methods. This algorithm is adaptable to the problem's requirements and can be easily modified. RAST has already been proved very useful in visual matching and recognition purposes.

D. Proposed Algorithm

Pre-processing part includes the down-scaling of the image along with some basic image processing techniques for enhancing the image to help the line segment detection part. The enhancing of the image is done by converting the image from RGB into HSV image and then picking up it's third channel, because this way LSD algorithm gives the best results.

Training of the CNN model has been already explained in the previous section. Line segment detection is being done using the LSD.

The algorithm idea for combining LSD with CNN model is based on RAST, which is mainly a divide and conquer algorithm as already explained in the previous sub-section. It starts first with the detection of line segments, then for each line segment CNN model is applied on both of its sides resulting an answer about the presence or absence of document in that part of the image. As LSD gives out a lot of detected lines so we remove some line segments as they are not of any use to us on two basis: small in length, and having CNN result saying the presence of document on both sides. Then priority queue of line segments is created based on the confidence value of the CNN result, as the result having greater confidence value is more accurate than the one with less confidence value. Afterwards priority queue is traversed and the side of the line which says the absence of document in it is replaced with black pixels and the part which says the presence of document in it remains unchanged. Traversing the priority queue in this way resulted in an image having the document part in its original pixels and all the other parts in black pixels, and thus boundary of he document has been extracted.

The main challenging part in this was to input those parts of the image to the CNN model which were created through the division of the image along the detected line segments. Those parts of the image were not rectangular in shape and thus to make them rectangular their mirror image was concatenated along that division line as shown in the Figure 4.

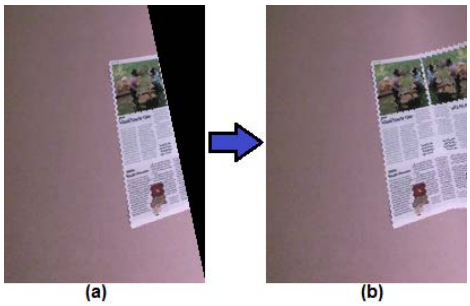

Fig. 4: CNN Input Image: After removing the part of image which is on other side of the line segment we get image (a) which is then transformed into image (b) so that there are no extra black pixels in it.

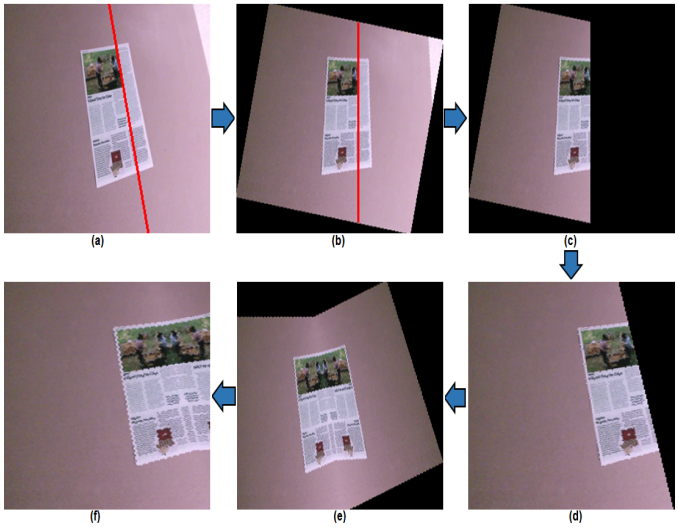

Fig. 5: CNN Input Image Formation: It shows how image is divided across a line segment and how the extra black part is removed from the image. (a) the original image is rotated in order to make detected line segment parallel to y-axis as shown in (b), so that it is easy to remove the part on the other side of line segment as shown in (c), then it is rotated back to it's original position (d), after that it's mirror image is concatenated with it as shown in (e), in the end image is cropped to its original size as shown in (f).

This formation of the input image to the CNN model is shown step by step in the Fig 5. Fig 5 (a) shows the detected line segment on the original image, this image is then rotated to make this detected line segment parallel to the y-axis so that it can be cropped easily as shown in Fig 5 (b), then the other side of the line segment is removed as shown in Fig 5 (c), then Fig 5 (d) is showing the rotation of the image to it's original position, then as showing in Fig 5 (e) it's mirror image is concatenated with it, in the end then the image is cropped according to it's original size so that we don't have any black area in our image as shown in Fig 5 (f).

Post-processing part is further divided into three steps as shown in the Figure 6. It's first task, 'Corner point detection' detects the four corner points in the document boundary by calculating the curvature for each boundary point and picking out the ones with the highest values. Second task, 'Mapping onto original image' maps back these corner point detected on the down-scaled image onto the original image by calculating the ratio by which the image was down-scaled in the very start. Third task, 'Refinement of corner points' refines the position of the corner point which has been mapped onto the original image by getting a window of 200x200 pixels around this mapped corner point and then again finds the corner point in that part by converting the image into binary and finding the corner point again using curvature values as shown in Figure 7. This binarization of

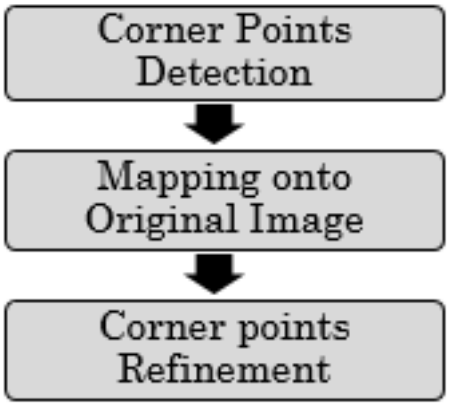

Fig. 6: Post-Processing Block Diagram

the image is not normal binarization, because in most of the cases the background color was in light color which outputs the wrong results in simple binarization, so to handle this situation image was first converted into HSV image and then **Otsu Threshold** [22] was applied on it's channel one. This refinement part was done because up-scaling an image back from the down-scaling causes some error in the position of the point therefore, in order to remove that ambiguity and error this refinement was required. In the end, we have the final corner points of document with better accuracy.

Complete step by step illustration of this process using an example has been shown in Figure 8.

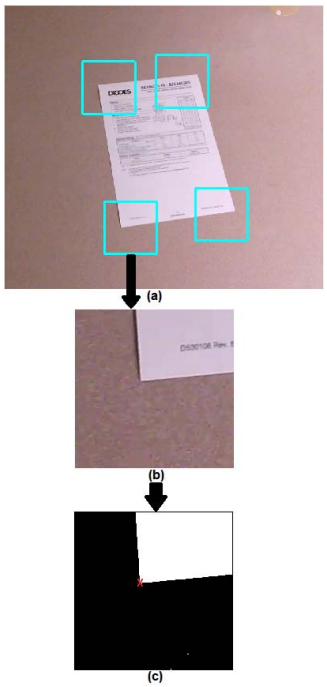

Fig. 7: Corner Points Refinement: (a) shows the original image with small windows around the detected corner point, (b) shows one of the four cropped window part, (c) shows the binarization of the cropped part and refined detected corner point

IV. Experiment and Results

A. DataSet

Deep learning's efficiency is based on how good the network has been trained. Training of deep learning network requires huge amount of data in order to get the desired results.

The dataset used in this scenario is from the SmartDoc competition [5]. This data is in the form of videos, each of them is ten seconds long having different backgrounds and document layouts. Training dataset contains only two videos, while Test dataset is divided into five categories based on their backgrounds where each category is having 30 videos in it, these 30 videos are 30 different layouts of the documents. Dataset of this SmartDoc competition was not for the purpose of training neural network models therefore, the amount of data is very very limited, in total the distinct images i.e. one frame from each video equals to 152 images combining the training and test dataset. In order to increase our training dataset we reorganised the distribution of training and test dataset. so our dataset distribution is like this:

$$\text{TrainingDataset} = \text{TrainingDataset'} \cup \text{TestDataset(1)'} \cup \text{TestDataset(2)'} \quad (1)$$

$$\text{TestDataset} = \text{TestDataset(3)'} \cup \text{TestDataset(4)'} \quad (2)$$

Here the TrainingDataset' and TestDataset' are the datasets from SmartDoc competition and TestDataset(n)' is the nth category of SmartDoc's test dataset. Fifth category of SmartDoc's Test dataset is not yet included in our experiments because it's background is a bit more complex and completely different from the backgrounds of other categories, this category will be used in the future releases of this proposed method.

After this distribution the total number of distinct images in our training dataset and test dataset are 62 and 60 respectively.

Above mentioned training dataset is still very small to properly retrain a CNN model, because shortage of training dataset may lead to overfitting problem therefore, after the above mentioned data distribution, data augmentation techniques have been applied to increase the amount of training dataset to avoid the overfitting problem.

Training dataset on which CNN model was trained contains 8052 images having 7332 positive and 720 negative samples. In this we picked one frame from each video and in total the starting number of images become 2+30+30=62 images. After this we applied data augmentation techniques of two types: one is of geometric type including crop and rotate and the other is comprised of the image processing filters like contrast adjustment, brightness adjustment, blur, sharpening and noise to make these 62 images into 8052 training images.

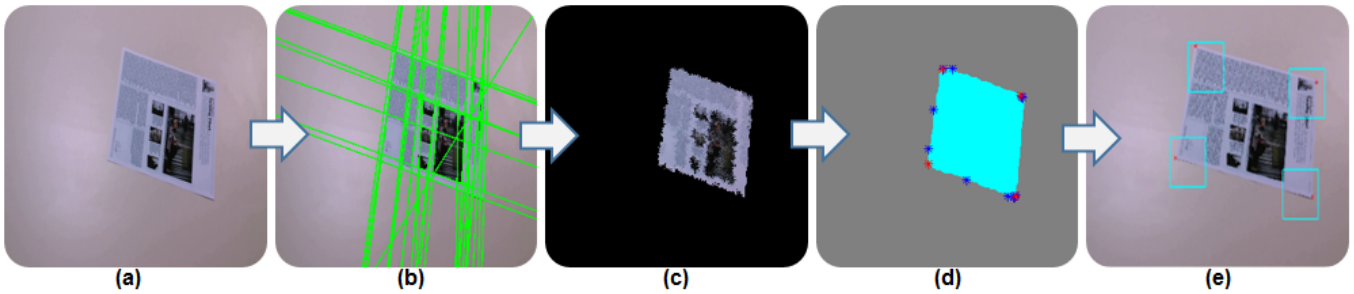

Fig. 8: Complete work-flow of the algorithm: (a) Downscaled original Image containing document, (b) LSD (Line segment detection) on the image, (c) Result of applying the CNN model on each side of the lines detected in the previous image, (d) Extracting corner points from the detected boundary of the document, (e) Refinement of the corner points after mapping the points onto original image

Test dataset contains 600 total images. As each category has 30 videos so we picked out ten frames from each category so for two categories then in total it becomes 30*10*2 = 600.
All the model training work is done in Matlab using MatConvnet library.

B. Experiments and Results

We have tested two experiments on this dataset for performance evaluation of this proposed method. One of them is explained in this sub-section and the other one is explained in the next sub-section under comparative evaluation.
In first experiment we calculated the accuracy of each corner point of the document using pixel difference as the error.
Accuracy is being calculated as shown in equation 3. Main calculation is being done in the error calculation.

$$\text{Accuracy} = 1 - \text{Error} \tag{3}$$

This 'Error' is ratio of the difference of pixels between the actual point and the resulted point to the total number of pixels lying inside the area of the actual document as expressed in equation 4 below. Here, only the pixel difference between the actual and resulted point is not taken rather it's ratio with the total pixels covered by the document is taken because in case of an image where the document is covering almost all of the space in the image and the other case where the document is in a very small size and is covering less than half of the image's area, so if there is a difference of 50 pixels then if we see them visually the error will seem almost negligible in the first case where the document is on the whole image while in the second case this 50 pixel difference will be seeing as a lot of error because it'll seem to be far away from the document.

$$\text{Error per document} = \frac{\text{pixelDifference}}{\text{totalPixelsCovered}} \tag{4}$$

After individual accuracy calculation, accuracy of each document is calculated by taking the mean of the accuracies of 4 corner points. Similarly accuracy of each category is then calculated by again taking the mean of the accuracies of all of the documents included in that category.
Accuracy of the corner points generated by our proposed method according to the background category are listed in the Table I.

Accuracy of result from this experiment seems promising

Category	Accuracy
Background3	0.9418
Background4	0.9480
Overall	0.9449

TABLE I: Experiment-1 Accuracy Results

but requires benchmarking against accuracies obtained by other approaches.

In the mentioned accuracy of this proposed method we have excluded the images on which our algorithm has failed. Failed images were detected by checking their accuracy if the corner point detection accuracy lies below 50 percentage then this algorithm has failed to detect the document, and in this way the total number of failed images we get are 22 out of 600.
The reason why our method fails on some images is just due to lack of training dataset's negative samples, as we couldn't produce negative sample equal to the number positive samples due to which the system sometimes recognizes a non-document part of the image as a positive sample, an example of this has been shown in the Figure 9.

C. Algorithm Analysis

After the basic implementation of proposed algorithm, it's analysis has been done by tuning the different parameters being used in it.
First of all in the CNN model training part, accuracy of algorithm is observed against different values of a parameter which is a part of training SVM classifier known as 'misclassification cost (c)', it's graph is shown in Figure 10. From the graph it is clear that at c = 0.6 our algorithm gives the best results.

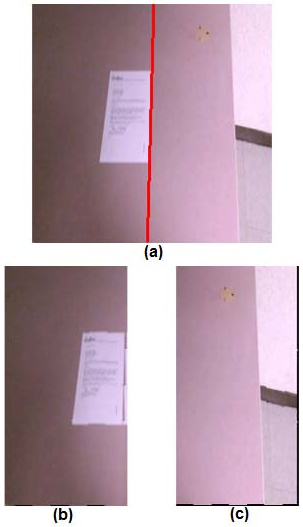

Fig. 9: Algorithm failure example: (a) shows the line segment around which the image is divided, here algorithm recognizes (b) and (c) both as having document in them (positive sample) due to lack of negative sample images in the training dataset

Afterward the minimum length line segment threshold was

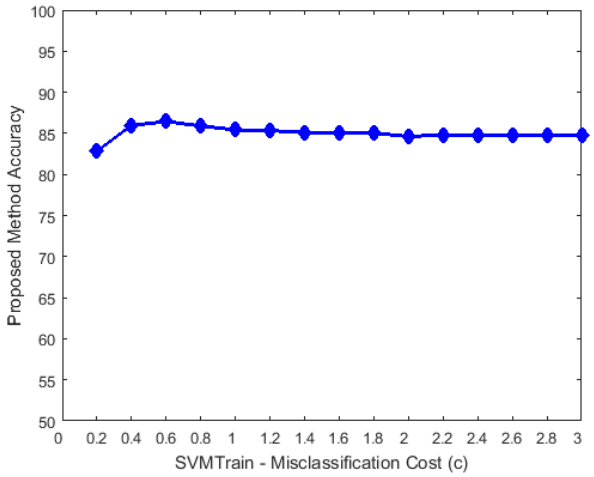

Fig. 10: Proposed method's accuracy for different values of SVM training parameter 'Misclassification Cost' (c).

tuned as shown in Figure 11. This threshold was applied in the initial stages of the algorithm where LSD detects the line segments, and those line segments whose length were less than this threshold were then being eliminated to reduce the useless processing on such small sized line segments. Graph shows that at threshold value = 20, we are having the best accuracy. Graph also shows that the accuracy tends to decrease as we increase the threshold value because increasing this threshold to a large value eliminates most of the useful line segments too.

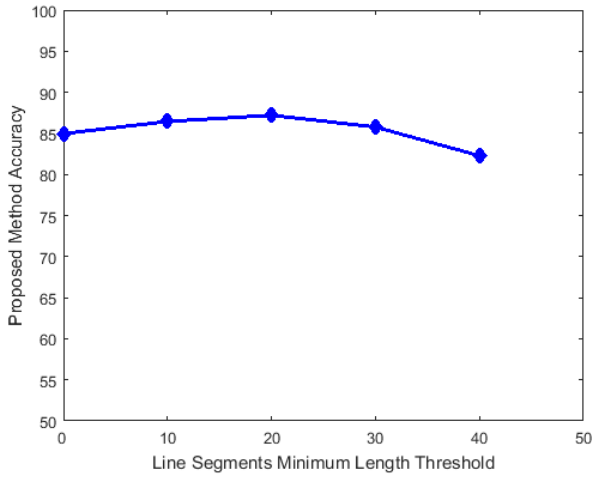

Fig. 11: Proposed method's accuracy for different values of minimum length line segment threshold.

At the last, the number of pixels being used for calculating

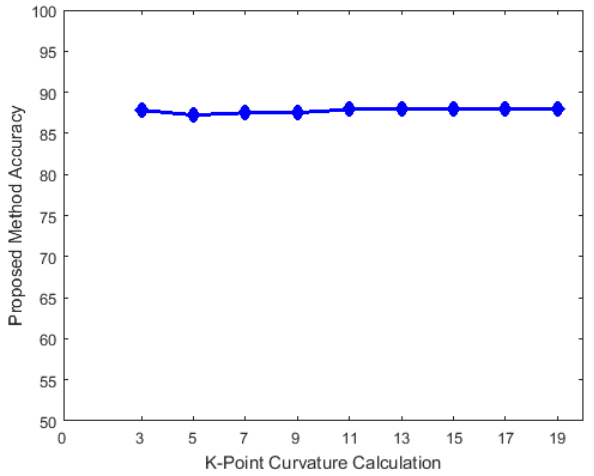

Fig. 12: Proposed method's accuracy for different number of points being used for curvature calculation.

the curvature in the post-processing part were analysed by testing the algorithm with different number of pixels for curvature calculation as shown in Figure 12, it shows that at value 11 it gives the best result and after that the accuracy tends to remain the same therefore, we selected the 11 point curvature calculation for better performance of proposed algorithm.

D. Comparative Evaluation

Another experiment that we did on this dataset for comparative analysis is by using Jaccard Index [23] expressed in equation 5. The reason why we are using this evaluation method is because in paper [5] they have used this method

for the evaluation of the methods participated in it and so now we'll have the comparison in this case.

$$JI(f) = \frac{area(G' \cap S')}{area(G' \cup S')} \quad (5)$$

Accuracy of the corner points generated by our proposed method according to the background category are listed in the Table II along with the SmartDoc competition participants' methods accuracies.

Accuracy Results		
Method	Background 3	Background 4
A2iA-1	0.9117	0.6352
A2iA-2	0.9118	0.8264
ISPL-CVML	0.9846	0.9766
LRDE	0.9889	0.9837
NetEase	0.9621	0.9511
Proposed Method	0.9068	0.8978
SEECS-NUST	0.7811	0.7832
RPPDI-UPE	0.9697	0.3649
SmartEngines	0.9897	0.9785

TABLE II: Experiment-2 Accuracy comparison between proposed method and SmartDoc's participants' methods

V. Future Work

A lot more training data is required to make this algorithm work in the ubiquitous environment as we had insufficient amount of data to test it into every type of environment. Increasing the size of training data will also fine-tune the CNN model further resulting in better accuracy. As the last category from SmartDoc's test dataset was excluded because it's background was not similar to the ones on which the model was trained therefore, any future release of proposed method results needs to accommodate this category.
In order to meet these goals we need to collect a lot of data of documents with different backgrounds.

VI. Conclusion

As seen from the results in Table II our method's results are promising in case of Background4 but not much competitive in case of Background3. This accuracy achieved is not the best this algorithm can achieve. It can be increased if we train our model more as in our case we were short on the training data. Also the adaptive nature of the machine learning can make this algorithm work with all kinds of documents in any environment by just retraining the model according to the environment and document type we need.
This algorithm is a more generalized solution of the given problem because it can be extended to any scenario, providing that we have that much data to properly train our model so that it can handle every scenario correctly.
Pertaining to the adaptive nature of machine learning, this algorithm can also be used for similar object detections which are not curved and thus line segment detection will do the supportive work with CNN model for them too.

References

[1] Zhang, Yunsheng, Yang Liu, and Zhengrong Zou. "Comparative study of line extraction method based on repeatability." J. Comput. Inf. Syst 8 (2012): 10097-10104.
[2] von Gioi, Rafael Grompone, et al. "LSD: a line segment detector." Image Processing On Line 2 (2012): 35-55.
[3] Lampert, Christoph H., et al. "Oblivious document capture and real-time retrieval." Proc. CBDAR2005 (2005): 79-86.
[4] Chen, Francine, et al. "SmartDCap: semi-automatic capture of higher quality document images from a smartphone." Proceedings of the 2013 international conference on Intelligent user interfaces. ACM, 2013.
[5] Burie, Jean-Christophe, et al. "ICDAR2015 competition on smartphone document capture and OCR (SmartDoc)." Document Analysis and Recognition (ICDAR), 2015 13th International Conference on. IEEE, 2015.
[6] Kim, Woong Hee, Jongwoon Hwang, and Thomas Sikora. "Document Capturing Method with a Camera Using Robust Feature Points Detection." Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on. IEEE, 2011.
[7] Zhang, Zhengyou, and Li-Wei He. "Whiteboard scanning and image enhancement." Digital Signal Processing 17.2 (2007): 414-432.
[8] Rodrguez-Pieiro, Jos, et al. "A new method for perspective correction of document images." IST/SPIE Electronic Imaging. International Society for Optics and Photonics, 2011.
[9] Fujimoto, Yusaku Fujii Katsuhito. "Perspective rectification for mobile phone camera-based documents using a hybrid approach to vanishing point detection."
[10] Yin, Xu-Cheng, et al. "A multi-stage strategy to perspective rectification for mobile phone camera-based document images." Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). Vol. 2. IEEE, 2007.
[11] Guillou, Erwan, et al. "Using vanishing points for camera calibration and coarse 3D reconstruction from a single image." The Visual Computer 16.7 (2000): 396-410.
[12] Kofler, Christian, et al. "Gestural interaction for an automatic document capture system." Proceedings of the Second International Workshop on Camera-Based Document Analysis and Recognition. 2007.
[13] Clark, Paul, and Majid Mirmehdi. "Rectifying perspective views of text in 3D scenes using vanishing points." Pattern Recognition 36.11 (2003): 2673-2686.
[14] Lu, Shijian, Ben M. Chen, and Chi Chung Ko. "Perspective rectification of document images using fuzzy set and morphological operations." Image and Vision Computing 23.5 (2005): 541-553.
[15] Lu, Shijian, and Chew Lim Tan. "The restoration of camera documents through image segmentation." International Workshop on Document Analysis Systems. Springer Berlin Heidelberg, 2006.
[16] Miao, Ligang, and Silong Peng. "Perspective rectification of document images based on morphology." 2006 International Conference on Computational Intelligence and Security. Vol. 2. IEEE, 2006.
[17] Stamatopoulos, N., B. Gatos, and A. Kesidis. "Automatic borders detection of camera document images." 2nd International Workshop on Camera-Based Document Analysis and Recognition, Curitiba, Brazil. 2007.
[18] Simard, Patrice Y., David Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." ICDAR. Vol. 3. 2003.
[19] Breuel, Thomas M. "Two geometric algorithms for layout analysis." International workshop on document analysis systems. Springer Berlin Heidelberg, 2002.
[20] Breuel, Thomas M. "Fast recognition using adaptive subdivisions of transformation space." Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on. IEEE, 1992.
[21] http://stats.stackexchange.com/questions/180850/how-are-filters-and-activation-maps-connected-in-convolutional-neural-networks
[22] "Opencv: Image Thresholding". Docs.opencv.org. N.p., 2016. Web. 28 July 2016.
[23] Everingham, Mark, et al. "The pascal visual object classes (voc) challenge." International journal of computer vision 88.2 (2010): 303-338.