# Real-time Document Localization in Natural Images by Recursive Application of a CNN

Khurram Javed*, Faisal Shafait†
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
Email: *14besekjaved@seecs.edu.pk, †faisal.shafait@seecs.edu.pk

***Abstract*—Seamless integration of information from digital and paper documents is crucial for efficient knowledge management. One convenient way to achieve this is to digitize a document from a natural image. This requires precise localization of the document in the image. Several methods have been proposed to solve this problem but they rely on traditional image processing techniques which are not robust to extreme viewpoint and background variations. Deep Convolutional Neural Networks (CNNs), on the other hand, have shown to be extremely robust to variations in background and viewpoint in object detection and classification tasks. Inspired by their robustness and generality, we propose a novel CNN based method to accurately localize documents in real-time.**

**We model localization problem as a key point detection problem. The four corners of the documents are jointly predicted by a Deep Convolutional Neural Network. We then refine our prediction using a novel recursive application of a CNN. Performance of the system is evaluated on ICDAR 2015 SmartDoc Competition 1 dataset. The results are comparable to state of the art on simple backgrounds and improve the state of the art to 94% from the previous 86% on the complex background. Code, dataset, and models are available at:** https://github.com/KhurramJaved96/Recursive-CNNs**.**

***Index Terms*—Document capture, Real-time, Document detection, Machine learning, CNN.**

## I. Introduction

Documents exist in both paper and digital form in our everyday life. Paper documents are easier to carry, read, and share whereas digital documents are easier to search, index, and store. For efficient knowledge management, it is often required to convert a paper document to digital format. In the past this has been achieved by scanning the document using a flat-bed scanner followed by application of OCR and document analysis techniques to extract the content from the scanned image. Scanners, however, are slow, costly, and not portable. Smart-phones, on the other hand, have become extremely accessible in the past decade and can take high resolution images. Consequently, there has been a recent trend to use smart-phones to digitize paper documents.

A natural image of a document taken by a smart-phone can not be directly digitized using the methods developed for scanned images. This is because hand-held camera images pose challenges such as presence of background texture or objects, perspective distortions, variations in light and illumination, and motion blur [1].

Document segmentation from the background is one of the major challenges of digitization from a natural image. An image of a document often contains the surface on which the document was placed to take the picture. It can also contain other non-document objects in the frame. Precise localization of document, often called document segmentation, from such an image is an essential first step required for digitization. Some of the challenges involved in localization are variations in document and background types, perspective distortions, shadows, and other objects in the image. An ideal method should be robust to these challenges. It should also be able to run on a smart-phone in a reasonable time. We present a data-driven method to solve the problem of document segmentation that satisfies all the requirements of a practical system. More precisely, we propose a method that uses deep convolutional neural networks combined with an algorithm to localize documents on a variety of backgrounds. Our method outperforms state of the art in literature on the complex background and can run in real-time on a smart-phone. Furthermore, it is more general compared to previous methods in the sense that it can be adapted to be robust to more challenges simply by training on better representative data. Traditional image processing methods, on the other hand, have at-least some intrinsic limitations because of assumptions made about the environment.

## II. Literature Review

One of the earliest schemes of document localization relied on a model of the background for segmentation [2] The background was modeled by taking a picture of the background without the document. The document was then placed in the environment and another picture was taken. The difference between the two images was used to identify the location of the document. This method had the obvious drawbacks that the camera had to be kept stationary and two pictures had to be taken.

Different methodologies have been proposed since then for this task, which can be broadly classified as camera calibration based systems [4], [5], document content analysis based systems, [6], [7], [8], [9], [10], and geometric analysis based systems [13], [14], [15], [16].

Camera calibration based systems utilize information of the position and angle of the camera relative to the expected doc-

2379-2140/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDAR.2017.26


Fig. 1: Overall architecture of the system. 32 x 32 Image is used by Corners Detector to predict four corners. This prediction is used by region extractor algorithm to extract corner images. Each of the four corner images are then recursively refined by Corner Refiner to get the final output.

ument to estimate the location of the document. However even the slightest change in perspective or position of the camera requires recalibration. This makes such systems impractical for hand-held devices.

Document content analysis techniques locate areas in the image resembling text, and assume the background to be dark and plain in contrast to the document [10]. This assumption often fails on documents with graphics and fancy backgrounds such as brochures.

Geometric analysis techniques identify quadrilateral objects in the images as documents, and are robust to rotation or perspective distortions [11]. Generally, these techniques first identify lines in the image. These lines are then used as features to identify quadrilaterals in the images which are similar to documents [13]. These methods fail if the document lacks straight borders (i.e. the pages are curled or not flat).

A lot of work on Document Localization was done as a result of ICDAR 2015 SmartDoc Competition [11]. Eight submissions were made for the competition out of which we discuss the top two submissions. LRDE, the highest scoring method, represents the image in a hierarchical way in a tree structure called tree of shapes. For each frame, the shape that looks most like a paper sheet is selected on the basis of energy that depends on how well the shape matches a quadrilateral. LRDE also uses the prediction made in earlier frames to select the final shape in a given frame. Even though LRDE is highest scoring submission, it is less generic in the sense that it exploits the video nature of dataset. The performance of the system on single images may not be as good. ISPL-CVML, the second best submission, uses LSD (Line Segment Detector) on down-sampled images to get horizontal and vertical segments. It then exploits edge and color features to select the most probable document candidate. Finally, the boundaries are refined on the high-resolution image. Both of these methods perform extremely well on simple backgrounds and adequately on the complex background of SmartDoc dataset.

It is pertinent to mention that Convolutional Neural Networks have been used for facial key point detection in an approach similar to ours [17]. However, there are some fundamental differences in the two approaches. For facial key point detection, the author used 23 different networks arranged in three layers to make the predictions. The networks in layer two and three were only used for fine-tuning the result. We, on the other hand, train only two networks and recursively use one of the networks multiple times to achieve extremely high precision.

## III. Methodology

We model the problem of document localization as *key point detection* such that the corners of a document are the four key points to be detected. These four points are: *Top left corner* (**TL**), *Top right corner* (**TR**), *Bottom right corner* (**BR**), and *Bottom left corner* (**BL**).

*a) Definition:* Let top left corner of the image be the origin. Also, let $(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$, and $(x_4, y_4)$ be the co-ordinates of the four points. Then TL = $(x_k, y_k)$ such that $x_k + y_k$ is minimum. The remaining points are traversed in clockwise direction starting from TL and labeled as TR, BR, and BL respectively.

Our proposed approach can be divided into two steps; first, we jointly predict all four corners of the document using a deep convolutional neural network with a similar structure as AlexNet [18]. Secondly, we refine each of the predictions separately using recursive application of a shallow convolutional neural network.

### A. Joint corner points detection

We use a 8-layer deep convolutional neural network with two hidden fully connected layer and eight regression heads to jointly predict values of TL, BR, BL, and TR respectively. To avoid the high computational and memory cost of deep networks, we resize the image to 32 x 32 before feeding it to the network.

Ideally we would want our model to precisely locate the corners. This can not happen for two reasons:

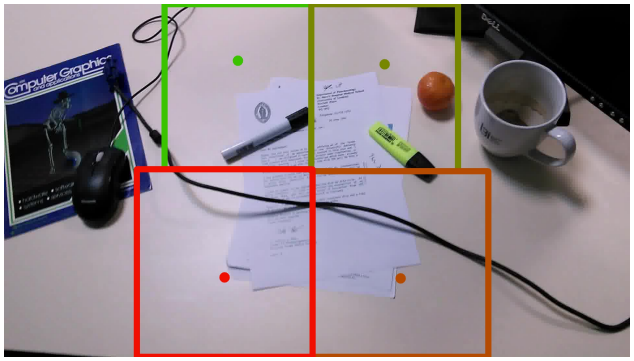

Fig. 2: Results of joint corner detection and corner cropping. The dots are the output of the network whereas the rectangles are the regions extracted from these predictions. We get the four regions by cropping the image at the midpoint of adjacent predictions.

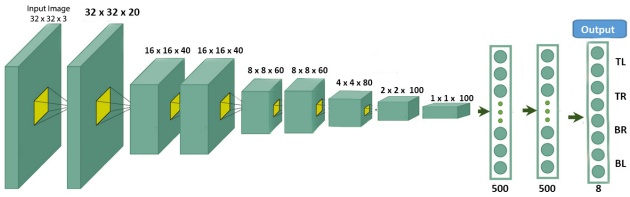


Fig. 3: Model architecture for corner detection. 5x5 kernels are used and Rectified Linear Unit (ReLU) is applied after every convolutional layer. The eight outputs are interpreted as co-ordinates of four corners.

*1) Lack of precise features:* Standard Deep Convolutional Neural Network architectures are inherently bad at precise localization and segmentation tasks [19]. This is because the last convolutional layer only contains high-level features of the whole image. While these features are extremely useful for classification and bounding box detection, they lack the information for pixel level segmentation. Several complicated architectures have been proposed that are better at precise spatial predictions [20], but these architectures are often difficult to implement and train. Furthermore models that only use high level features of the whole image can be argued to have better generalization.

*2) Upscale error:* Upscaling a prediction made on $32 \times 32$ image introduces a large error. For instance for an original image of $640 \times 640$ pixels, the upscaling can introduce an error of $\pm$ 10 pixels.

Because of this imprecision, output of the model is not used as the final result. Instead, we use the output to extract four regions from the image such that each region contains exactly one of the four corners. An algorithm is required to extract these regions on the basis of prediction. We call this algorithm Region Extractor.

*Region Extractor:* Let TL', TR', BR', and BL' be the predicted corners of the document. For each of the prediction,

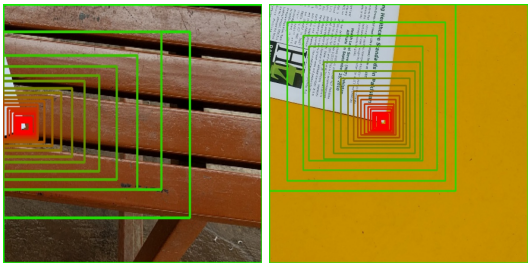

Fig. 4: Recursive Corner Refinement. Based on prediction the area least likely to contain the point is discarded and cropped image is fed to the model again. The squares represent the area that the model sees in each iteration.

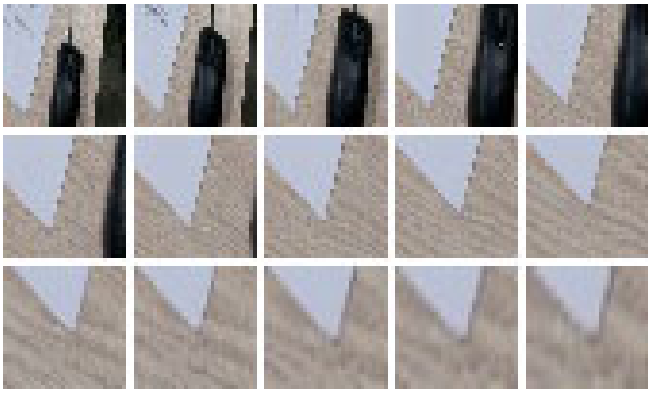

Fig. 5: Recursive corner refinement of a corner image. Each subsequent image is a slightly cropped version based on prediction of Corner Refiner on previous image.

we extract the region containing corresponding corner by cropping the image along x-axis at a point above and below the prediction and along y-axis at a point to the right and left of the prediction. Assume, without loss of generality, that we want to get region containing TR based on prediction TR'. We find this region by discarding the image that is to the left of line parallel to y-axis and passing through the x-coordinate of midpoint of TR' and TL'. Similarly, we discard image that lies below the line parallel to x-axis and passing through y-coordinate of midpoint of TR' and BR'. Finally, if the area above and to the right of TR' is greater than area below or to the left, we make it equal by discarding a strip of image from above or right side of the image. All four regions can be extracted using the same idea. Our particular implementation of Region Extractor has the advantage that the regions are normalized w.r.t document size. Sample output of the first model and the extracted regions can be seen in Fig. 2 whereas the model architecture can be seen in Fig. 3.

### *B. Recursive Refinement of corners*

In the next step, each of the four images is resized to 32 x 32 and passed to a second Convolutional Neural Network with two regression heads. We call this network **Corner Refiner**.

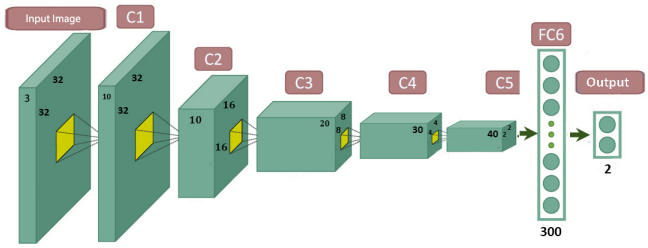


Fig. 6: Model architectures of Corner Refiner. All kernels are 5x5 and applied with padding. ReLU activation is applied after every Convolutional Layer. Two outputs are interpreted as co-ordinates of corner point.

Fig. 7: Samples from dataset we collected. A total of 120 images are captured on 50 different backgrounds

Corner Refiner is trained to detect a corner point in an image containing exactly one corner. It is again not possible to precisely detect the corner with the network in a single go for the above stated reasons. Consequently, we use Corner Refiner multiple times to converge to the correct location. In each iteration, we use the model's prediction on 32 x 32 image to discard a part of full resolution image least likely to contain the corner and feed the remaining image to the same network again. To quantify how much of image should be discarded in each iteration, we define a hyper parameter RF (Retain Factor) such that $0 < RF < 1$. In each iteration, we only retain RF fraction of image closest to the predicted point in both vertical and horizontal direction. This means that after $n$ iterations, the $(H, W)$ image is cropped to $(H \times RF^n, W \times RF^n)$. Image size $<(10 \times 10)$ is used as the stopping criterion. This recursive application of the model allows us to converge to the corner with extremely high precision. Affect of value of RF on time and accuracy can be seen in Table. II whereas a visualization of convergence and model architecture can be seen in Fig. 4 and Fig. 6 respectively.

## IV. Training and Implementation Details

### A. Dataset

'ICDAR 2015 SmartDoc Challenge 1' dataset is used for training and testing. The dataset contains 120 videos. Each video is around 10 seconds long and contains one of the six types of documents placed on one of the five distinct backgrounds. A background is defined to be complex if it contains other objects in the frame placed near or on top of the document. Only background 5 is complex in this dataset. We split these videos into three disjoint sets. First 50% of the frames from each video are used for training, next 20% are used for validation, and remaining 30% are used for testing. We do not randomly split frames into three sets because nearby frames are almost identical and should not be in different sets. We also leave all instances of "Tax" document from the train, validation, and test set to quantify the generalization of the system on unseen documents. In addition to SmartDoc dataset, we collect 120 images to increase variation in train set (Fig. 7). These 120 images are augmented to 15,360 images by randomly rotating, cropping, changing contrast, and changing brightness of each image. Our final train set is union of SmartDoc train set and the self-collected images.

Ground truth is available in the form of labels tl, tr, br, and bl for SmartDoc dataset. It should be noted that our definition of labels TL, TR, BR, and BL is not consistent with the provided ground truth. The ground truth is from the perspective of how a person would read the document. This means that tl corner of a document rotated by 180 degrees would be near the bottom right corner of image. For training, we rename the labels such that they satisfy our definition.

### B. Training Details

*1) Dataset Preparation:* To prepare data for the first model, we randomly crop images with the constraint that each cropped image has all four corners. We then resize all the images to 32 x 32. Lastly, we calculate image mean from train set and subtract the mean from every image during train and test time.

For the second model, we crop each image into four corner images. The corner images range from 10x10 to largest possible image such that it has exactly one corner point. All the images are resized to 32 x 32.

*2) Frameworks, Hyper parameters, and Loss function:* The machine learning models are implemented in Tensorflow [22] whereas the algorithms are implemented in Python using Numpy. Weights for the models are initialized from a truncated normal distribution with a standard deviation of 0.1. Dropout [21] of 0.8 is used on fully connected layers and Adam Optimizer [23] is used to minimize L2 distance between predicted co-ordinates and ground truth co-ordinates. Learning rate of 1e-5 is used for corner refinement model and 5e-4 for joint corner detection model. We also randomly change the brightness and contrast of our images before feeding them to the model to prevent over-fitting.

## V. Experiments and Results

### A. Experiment Design

We design three experiments to test the performance of the system on test set, on unseen backgrounds, and on unseen documents. All results are reported with an RF (Retain Factor) value of 0.85.

*Experiment 1:* In the first experiment we run the model on test set and compare our results with previous reported results on the same dataset. Although this experiment is enough to test the performance of the system on the competition dataset, it

TABLE I: Performance comparison. On the complex background 5, our system outperforms all existing methods by a large margin. All results are taken from SmartDoc competition [11] unless referenced otherwise.

| Method | Background 1 | Background 2 | Background 3 | Background 4 | Background 5 | Overall |
|---|---|---|---|---|---|---|
| A2iA-1 | 0.9724 | 0.8006 | 0.9117 | 0.6352 | 0.1890 | 0.7788 |
| A2iA-2 | 0.9597 | 0.8063 | 0.9118 | 0.8264 | 0.1892 | 0.8090 |
| ISPL-CVML | **0.9870** | 0.9652 | 0.9846 | 0.9766 | 0.8555 | 0.9658 |
| LRDE | 0.9869 | 0.9775 | 0.9889 | **0.9837** | 0.8613 | 0.9716 |
| NetEase | 0.9624 | 0.9552 | 0.9621 | 0.9511 | 0.2218 | 0.8820 |
| SEECS-NUST | 0.8875 | 0.8264 | 0.7832 | 0.7811 | 0.0113 | 0.7393 |
| RPPDI-UPE | 0.8274 | 0.9104 | 0.9697 | 0.3649 | 0.2163 | 0.7408 |
| SmartEngines | 0.9885 | **0.9833** | **0.9897** | 0.9785 | 0.6884 | 0.9548 |
| L. R. S. Leal, et al. [12] | 0.9605 | 0.9444 | 0.9647 | 0.9300 | 0.4117 | 0.8950 |
| **SEECS-NUST-2** | 0.9832 | 0.9724 | 0.9830 | 0.9695 | **0.9478** | **0.9743** |

has two limitations; first, it can not tell us how well our system generalizes to unseen backgrounds since all backgrounds were used for training. Secondly, it can not tell us how well our system generalizes to new documents on similar background.

*Experiment 2:* To address the first limitation and quantify generalization on unseen background, we cross-validate our system by removing each background from training set and then testing on the removed background.

*Experiment 3:* To address the second limitation and quantify generalization on unseen document, we test our system just on the 'Tax' document. All instances of this particular document were excluded from the train, validation, and test set [1].

### B. Evaluation Protocol

We evaluate the performance of the system using the method described in the SmartDoc competition report [11], [24]. We first remove the perspective transform of the ground truth quadrilateral ($G$) and predicted quadrilateral ($S$) using the document size. Let's call the corrected co-ordinates $G^{'}$ and $S^{'}$. Then for each frame, we compute Jaccard index (JI) as follows:

$$JI = \frac{\text{area}(G^{'} \cap S^{'})}{\text{area}(G^{'} \cup S^{'})}$$

Overall score is the average of JI of all the frames.

### C. Results

The results of experiment 2 and 3 are shown in Fig. 8. It can be seen that our system generalizes to unseen document types perfectly. This is not surprising since it is not possible for our model to rely on features of document's layout or content because of the low resolution of input image (Fig. 9). From the results we can also infer that our system generalizes well to unseen simple backgrounds. Even when background 1, 2, 3, and 4 were removed from training, the accuracy on them dropped only slightly. Generalization to the more difficult background 5, on the other hand, is a different story.

Accuracy on background 5 dropped to 0.66 from 0.94 when all samples of background 5 were removed from training. This

[1]We did not feel the need to cross validate on all document types because generalization on unseen document was expected because of low resolution of input image as shown in Fig. 9.

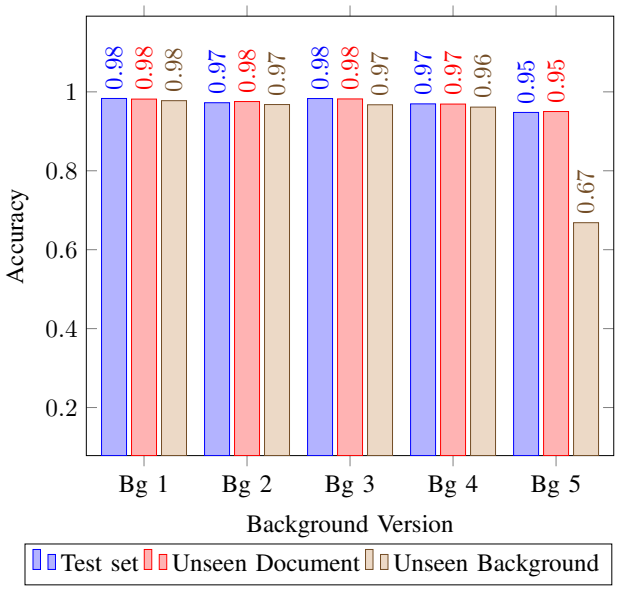


Fig. 8: Experiment 2 and 3 results. System shows good generalization on unseen simple backgrounds and unseen document type

significant drop can be explained by the fact that apart from background 5 images, almost all images in our train set contain only a single object (document) on a surface (There were only 10 exceptions). Background 5, on the other hand, has every day objects placed on top of and near the document. This dissimilarity in train and test set could be the reason of bad performance. We suspect that by simply adding samples which contain arbitrary everyday objects, we would get much better generalization. We leave this idea untested for now.

### D. Performance

The system is tested on an Intel i5-4200U 1.6 Ghz CPU with 8 GB ram on 1920 x 1080 images. Total time depends on the hyper-parameter Retain Factor (RF). Time taken with different values of RF is shown in Table. II.

It should be noted that the system is not implemented with efficiency in mind. In the current implementation, the four

Fig. 9: Two different documents on similar background look identical. Low resolution prevents the model from relying on content or structure of document allowing it to generalize on unseen documents.

TABLE II: Relationship between time, Retain Factor, and accuracy. It is possible to get massive performance gains at the cost of a slight decrease in accuracy

| Retain Factor | Accuracy | Time in milliseconds |
|---|---|---|
| 0.85 | 0.9743 | 320 |
| 0.75 | 0.9701 | 210 |
| 0.65 | 0.9617 | 150 |
| 0.60 | 0.9604 | 130 |
| 0.50 | 0.9513 | 100 |

corners images are processed sequentially. It is possible to process them simultaneously by passing them through the model as a batch of four images. This should give us a noticeable performance boost.

Lastly, it is easy to see that given N = number of pixels in the image, the growth rate of the corner refinement algorithm is $\Theta(\sqrt{N})$. This is because both length and width of the image are cut by the RF in each iteration.

## VI. Conclusion

In this paper we present a novel approach to localize document in a natural image. We model the problem of localization as a key point detection problem. By using a deep convolutional network and recursive application of a shallow convolutional network, we demonstrate that our method gives results comparable to state of the art on simple backgrounds and improves the state of the art to 94% from the previous 86% on the complex background. We also demonstrate that our system is able to generalize well on new documents and unseen simple backgrounds. Furthermore, we suspect that generalization on unseen complex backgrounds can be improved either by adding more complex images in training or by synthetically degrading the training images by adding patches of different colors. We believe the latter because in a 32 x 32 downscaled complex image, the objects lose their distinctive features and appear more like a patch of their most common color.

## References


[1] J. Liang, D. Doermann, et al. "Camera-based analysis of text and documents: a survey." International Journal on Document Analysis and Recognition, vol. 7, no. 2, pp. 84-104, 2005.

[2] H. Lampert H, T. Braun , et al. "Oblivious document capture and real-time retrieval." Proceedings. Camera-Based Document Analysis and Recognition, 79-86, 2005.

[3] F. Chen, S. Carter, et al. "SmartDCap: semi-automatic capture of higher quality document images from a smartphone." Proceedings. international conference on Intelligent user interfaces, ACM, 2013.

[4] E Guillou, D. Meneveaux, et al. "Using vanishing points for camera calibration and coarse 3D reconstruction from a single image." The Visual Computer 16.7 : 396-410, 2000

[5] C. Kofler, D. Keysers, et al. Gestural Interaction for an Automatic Document Capture System, 2007.

[6] P. Clark, M. Mirmehdi. "Rectifying perspective views of text in 3D scenes using vanishing points." Pattern Recognition 36.11 : 2673-2686, 2003

[7] S. Lu,, M. Chen, et al. "Perspective rectification of document images using fuzzy set and morphological operations." Image and Vision Computing 23.5: 541-553, 2005

[8] S. Lu, L. Tan. "The restoration of camera documents through image segmentation." International Workshop on Document Analysis Systems. Springer Berlin Heidelberg, 2006.

[9] L. Miao, S. Peng. "Perspective rectification of document images based on morphology." International Conference on Computational Intelligence and Security. Vol. 2. IEEE, 2006.

[10] N. Stamatopoulos,, B. Gatos, et al. "Automatic borders detection of camera document images." 2nd International Workshop on Camera-Based Document Analysis and Recognition, 2007.

[11] JC. Burie, J. Chazalon, et al. "ICDAR2015 competition on smartphone document capture and OCR (SmartDoc)."13th International Conference on Document Analysis and Recognition, IEEE, 2015.

[12] LRS Leal, BLD Bezerra. "Smartphone camera document detection via Geodesic Object Proposals." Computational Intelligence (LA-CCI), 2016 IEEE Latin American Conference on. IEEE, 2016.

[13] WH. Kim, Woong Hee, J. Hwang, et al. "Document capturing method with a camera using robust feature points detection." International Conference on Digital Image Computing Techniques and Applications, IEEE, 2011.

[14] Z. Zhang, L. He et al. "Whiteboard scanning and image enhancement." Digital Signal Processing 17.2 : 414-432, 2007.

[15] F. Yusaku, K. Fujii. "Perspective rectification for mobile phone camera-based documents using a hybrid approach to vanishing point detection." 2nd International Workshop on Camera-Based Document Analysis and Recognition, 2007.

[16] X.C. Yin, J. Sun, et al. "A multi-stage strategy to perspective rectification for mobile phone camera-based document images." Ninth International Conference on Document Analysis and Recognition. Vol. 2. IEEE, 2007.

[17] Y. Sun, X. Wang, et al. "Deep convolutional network cascade for facial point detection." Conference on Computer Vision and Pattern Recognition, 2013.

[18] A. Krizhevsky, I. Sutskever, et al. "ImageNet classification with deep convolutional neural networks." Advances Neural Information Processing Systems 25 : 1097 - 1105, 2012

[19] B. Hariharan, P. Arbelez, et al. "Hypercolumns for object segmentation and fine-grained localization" Computer Vision and Pattern Recognition, 2015.

[20] A. Newell, K. Yang, et al. "Stacked hourglass networks for human pose estimation." European Conference on Computer Vision. Springer International Publishing, 2016.

[21] N. Srivastava, G. Hinton, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research, pages 19291958, 2014.

[22] M. Abadi, A. Agarwal, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems, 2015". Software available from tensorflow.org.

[23] DP. Kingma, J. Ba "Adam: A method for stochastic optimization" CoRR, abs/1412.6980, 2014

[24] M. Everingham, L.V. Gool, et al. "The pascal visual object classes (voc) challenge." International journal of computer vision 88.2 : 303-338, 2010.