

Automated invoice data extraction using image processing

Akanksh Aparna Manjunath, Manjunath Sudhakar Nayak, Santhanam Nishith, Satish Nitin Pandit,
Shreyas Sunkad, Pratiba Deenadhayalan, Shobha Gangadhara

Department of Computer Science and Engineering, R V College of Engineering, Mysore Rd, RV Vidyaniketan Post, Bengaluru, India

Article Info

Article history:

Received Dec 17, 2021

Revised Aug 28, 2022

Accepted Sep 27, 2022

Keywords:

Invoice recognition

Opensource computer vision library

Optical character recognition

Tesseract

Text extraction

ABSTRACT

Manually processing invoices which are in the form of scanned photocopies is a time-consuming process. There is a need to automate the task of extraction of data from the invoices with a similar format. In this paper we investigate and analyse various techniques of image processing and text extraction to improve the results of the optical character recognition (OCR) engine, which is applied to extract the text from the invoice. This paper also proposes the design and implementation of a web enabled invoice processing system (IPS). The IPS consists of an annotation tool and an extraction tool. The annotation tool is used to mark the fields of interest in the invoice which are to be extracted. The extraction tool makes use of opensource computer vision library (OpenCV) algorithms to detect text. The proposed system was tested on more than 25 types of invoices with the average accuracy score lying between 85% and 95%. Finally, to provide ease of use, a web application is developed which also presents the results in a structured format. The entire system is designed so as to provide flexibility and automate the process of extracting details of interest from the invoices.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Pratiba Deenadhayalan

Department of Computer Science and Engineering, R V College of Engineering

Mysore Rd, RV Vidyaniketan Post, Bengaluru, India

Email: pratibad@rvce.edu.in

1. INTRODUCTION

Large and medium scale companies deal with a large number of invoices on a daily basis. The companies need to keep track of the products being sold through the invoices, the cash flow generated, taxes which are paid and other product related analytics which are useful for their business. Companies have a dedicated set of data entry staff who manually enter these details into the systems or databases. This is a laborious task and time consuming. This entire process can be automated with the help of a system which extracts the data from these invoices and updates the databases with the product details. In general, all invoices have a standard template consisting of the seller company's name and address as well as the consumer's name and address. This is followed by an invoice number which may be of a different format as adopted by the company. Then there is a table with product details followed by the transactional details like tax and total amount. This would help design a system which provides flexibility to the user in selecting the fields of interest, based on their use case. Hence, such a system which aims to capture this functionality of invoices and provide automation resulting in saving countless hours of human effort will be described in our paper. There are two parts to the system, an opensource computer vision library (OpenCV) [1] based framework which runs in the backend detecting text from the invoice, an optical character recognition (OCR) engine to extract text and a web application which serves as a tool to upload the invoices and view the structured details as well as store them in the database. This paper in addition aims to present a system which also automates the processing of an invoice. Another useful application of the text detection and recognition

system has been described in [2] but this restricts the system to detect the text regions on objects which are less clustered when compared to text present in documents and invoices. However, the paper presents some useful approaches when dealing with problems belonging to the concerned domain.

2. RELATED WORK

Efficient and accurate scene text detector (EAST) is a deep learning-based architecture that is used to recognize text in an image which is captured in an outdoor environment [3]. When the architecture was applied to recognize text in documents, the results were not promising. In order to automate the process of reading information from these invoices and either storing them in the databases or producing javascript object notation (JSON) template files, character recognition engines are required which are called OCR engines [4]. However, before the use of such software is made it is important to preprocess the invoices so that the accuracy of text extraction will increase. The various steps involved in preprocessing are also talked about which involve thresholding, binarization and morphological transformations. These methods have been used to remove background noise and get accurate results. OpenCV is a popular framework among computer vision enthusiasts. It has been used in the field of text detection as well. Almost all invoices contain tabular data, [5] explains about the use of OpenCV methods in extracting tabular data as well as metadata about the table stored in portable document format (PDF) formats. It gives a sound explanation of the algorithm used while detecting tables.

The fundamentals of digital image processing [6] explains the fundamentals of image processing and various operations such as image enhancement and restoration, morphological processing, segmentation, object recognition followed by representation and description. The suitable operations were used while preprocessing the invoice. The operations in MATLAB [7] gave a clear picture of what transformation was being made to the invoice by the operation.

One of the recent technologies in the field of computer vision is the use of artificial neural networks. [8] explains the use of one such deep network while performing text detection. A neural network architecture called spatial transfer network (STN-OCR) is presented in which a single neural network is trained to detect and recognize text from images. It makes use of semi-supervised neural networks for scene text recognition which is further optimized. The model was tested against benchmark detection datasets [9] and gave promising accuracy scores, [10] details the difficulty and challenges in developing a custom dataset and finding high quality varied datasets to annotate and use for training and testing, hence it was made the most out of publicly available datasets. The paper also details their efforts into automating invoice processing by using feature extraction but does not consider the tables present in invoices and the items present in the tables which also need to be processed and extracted if there is a need. It was observed that although the network architecture is simple, it is not easy to train this system, as a successful training requires extensive pre-training on easier sub tasks before the model can converge on the real task. Marinai *et al.* [11] Extends the architecture of STN with an attention model and uses convolutional recurrent neural network (CRNN) instead of traditional convolutional networks. One such similar neural network architecture is described in [12] which is used to recognize characters. This also requires large volumes of training data in order to fine tune to recognize special characters. A popular algorithm for object recognition is the you only look once (YOLO) algorithm and this algorithm has also been used in order to recognize the various text regions which are present in the document [13]. However, the use of the YOLO based detection module gave unsatisfactory results when compared to OpenCV based method, with the YOLO algorithm failing to detect many text regions. Considering the information extraction task as an image segmentation problem loses the text semantics, which can further complicate the processing of unstructured documents [10]. The existence of a dataset in order to train the network for segmentation is another problem in the case of training networks for extracting relevant information from structured documents. Baviskar *et al.* [14] Provides a well annotated dataset which helps in training a network which can recognize varying invoices of the same format. However, in order to train a model which can recognize a wide range of invoices, the network should be trained across a large number of invoices as well.

In order to detect the text present in the document, a recent algorithm as described in the paper [15] has been considered. The algorithm uses neural nets in the backend in order to group characters together which make up the region of text. However, it becomes difficult to categorize the text which is the required use case. This has been mitigated with the help of an annotation tool developed as part of the web app. For the text recognition part as well, the use of popular neural network-based algorithms such as [16] and [17] have been considered and while they gave promising results during the trials, the performance of these algorithms in recognizing new characters was found to be underwhelming. There are many OCR engines available in the current day market, Tesseract being the most popular one [18]. Tesseract is an open-source OCR engine. It follows a traditional step by step pipeline for performing image to text conversion. Outlines of the components are detected which are grouped together to form blobs. These blobs are organized into text

lines which are broken down according to the character spaces. A two-pass recognition step is performed on these cells in order to identify the words which may have been skipped in the initial pass. Soille [19] Describes in detail the various morphological transformations and their working, which have been used in the image preprocessing module of the proposed system.

In order to improve the accuracy of the Tesseract engine and recognize new symbols and characters [20], the Tesseract base model was trained with custom data files. The methodologies discussed in the paper were followed and new data and config files were created. This paper helped in providing an insight into the Tesseract training which helped in improving the Tesseract accuracy [21]. There are also some pre-processing steps suggested for improving the accuracy of OCR in [22]. Characters and words are separated from the background by using binarization, noise reduction, skew correction, and slant removal. Image is segmented into text or word using word extraction and text line detection. Structural features like vertical and horizontal lines are also extracted as features. Algorithms like k-nearest neighbour (KNN), support vector machine (SVM), Naive Bayes and neural networks are used to classify characters of languages like latin and devanagari.

3. METHOD AND IMPLEMENTATION

Selecting the methods of interest to us which best fit our use case, the proposed system consists of two main modules, a frontend web application which is used to mark the annotations, create template files and obtain the invoice data in a structured format. The other module is the backend which consists of an extraction tool and further includes modules such as image preprocessing, text detection and text recognition. The overall architecture of the system shown in Figure 1 contains two main sections, the annotation tool provided as part of the frontend used to mark the fields of interest and the core extraction part which runs at the backend and uses OCR to get fields and tables from invoice after performing the preprocessing steps.

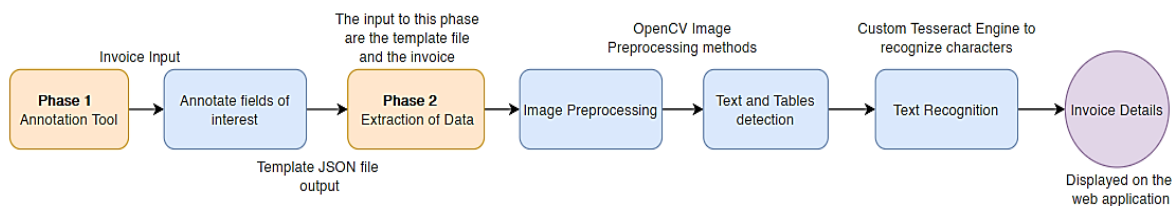


Figure 1. The overall architecture of the system

The web application would be used to mark and annotate the fields of interest by drawing bounding boxes around them. Further details with respect to the frontend are described in the following section. Before the introduction of the algorithm for extraction, few pre-processing steps to be performed on the invoice are discussed. The extraction part is concluded with a note on improving Tesseract accuracy by training the base model and on expanding its functionality by increasing the number of characters it can recognize.

3.1. Web application

The web application was built using basic hypertext markup language (HTML) and javascript. popular python web framework Django [23] was used in the backend to transform user uploaded files into the required format to enable processing. Poppler was used to extract the pages from the PDF and render them on the HTML canvas. A conservative estimate of 900×1,200 pixels was assumed to give the best result and hence the canvas size was also set to be the same. However, the image would be scaled to match the canvas and the annotation coordinates would be with respect to the canvas. During extraction, the magnitude of scaling will be used to correspond the canvas coordinates to the corresponding image coordinates so as to ensure accurate extraction. This was done so as to avoid downscaling the image and hence losing clarity and quality which would make it difficult to annotate in a previously low-resolution image. JavaScript is mainly involved in registering the user click inputs and outputs on the canvas and draws rectangles with a unique identifier (ID) associated with each of them. These are called annotations. As certain customizations had to be made for flexibility during annotation, a custom annotation tool was developed using JavaScript and HTML and an option was provided to edit a previous template by uploading it and editing, a feature which is not included in publicly available annotation tools.

Once the fields of interest have been marked, the invoice and the fields of interest registered as coordinates are used to extract the details from the invoice. The steps have been divided into image preparation or preprocessing to obtain higher accuracy results by normalizing the invoice, this step is followed by a table detection step which detects the table of interest to the user. Once all the text has been detected, it will be extracted by a custom trained Tesseract OCR engine in the text recognition phase.

3.2. Image preprocessing

The invoice has to be processed in order to improve the accuracy of detection and later on recognition. In order to eliminate background noise, preprocessing is done. The various steps performed as part of the preprocessing are mentioned in the subsequent sections. The first step is to remove noise using the OpenCV's denoising library. Denoising is done to remove the unwanted noise from the image so that presence of such noise will not affect the further transformations. Denoising is followed by two morphological operations performed in the same order, erosion, and dilation. Erosion removes white noises but it also shrinks the objects in focus, in order to increase the size of words dilation is performed. A sample invoice Figure 2 has been used to capture these operations and display the before and after state after processing the invoice.

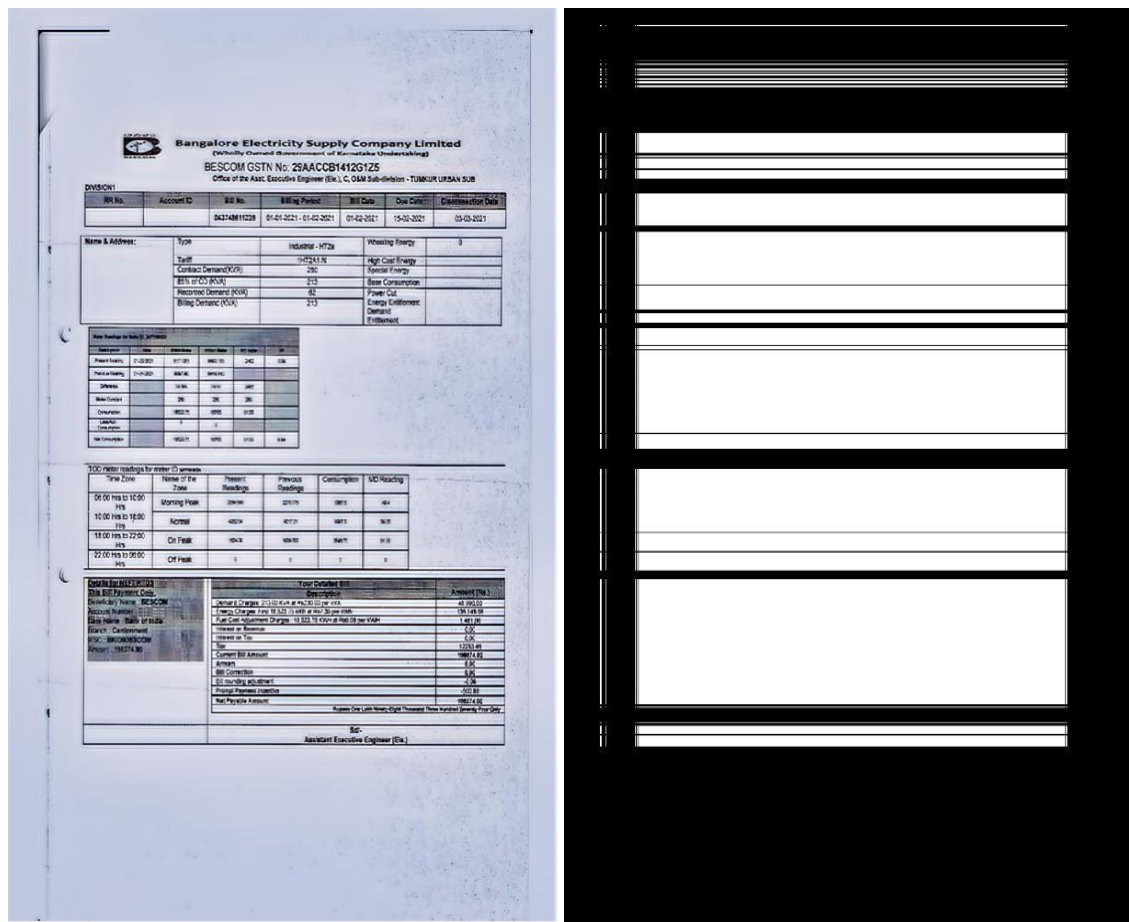


Figure 2. Picture of original image and image after performing preprocessing operations

3.3. Table detection

3.3.1. Bordered table detection

The horizontal and vertical lines in an image are identified using OpenCV functions. By superimposing these identified horizontal lines and vertical lines, we will be able to identify the structure of the table. After identifying the table structure, we will be applying morphological operations like dilation and erosion to get the mask of the table region. After obtaining the masks, we will apply canny edge detection to identify the edges of the rectangular region and then identify the contours. These contours are cropped from the original image.

3.3.2. Unbordered table detection

The table structure is identified by using a mix of template matching and an openCV based algorithm. The start of the table is specified in a template file JSON. The template file can be created using the user interface (UI) interface provided by the website. The number of columns in the table is also taken as an input parameter. Morphological operations are performed on the image (like dilation) to convert the regions of text into thick black boxes. Later contours are detected from these boxes. These contours are the regions of text in the image. The contours lying along the same horizontal line are grouped together. If the number of text regions along the same line match with the number of columns in the table, it is considered as a row of the table. Each region in the row will be considered as a cell of the table. There are some limitations to this approach as well:

- The text should be in dark colors and the background should be in white color. Otherwise, the text region identification would fail.
- If some text region is not part of the table, but has the same number of columns as the table, then that would be considered as part of the table.

3.4. Text recognition

In order to recognize the characters, present in the text region detected previously, OCR engines such as Tesseract were used. The comparison of different OCR engines has been detailed further. Tesseract is a popular OCR engine and its task is to recognize the characters present in an image or text file. The Tesseract engine is used in the form of a Python wrapper called PyTesseract [24] which is installed as a dependency in the Django platform. We went through a few opensource OCR engines like OCRopus, PaddleOCR and Tesseract. OCRopus is a tool to extract text from scanned documents. The basic pipeline includes binarization, segmentation and text recognition. It gave good results on normal English text but it was not able to recognize certain special characters like the rupee symbol. We could not find good methods to train the special characters on the existing model. Therefore, we could not use OCRopus in our implementation. However, Tesseract was used instead of the other available OCRs mainly because it had good documentation and an option to train special characters on the existing model.

While testing the performance of the system, few characters such as the Indian rupee or the pound symbol were not recognized with the default trained data file provided during with Tesseract. In order to rectify this issue, the base Tesseract model has to be trained to recognize special characters which are of interest. This training has to be done with datasets which include properly formatted files having the required special characters with annotated bounding boxes [25]. Open-source pretrained Tesseract files, trained on special characters such as Indian rupee symbol were used during the recognition pipeline.

4. RESULTS AND DISCUSSION

The extraction step involves two parts, one being extracting product details inside the table and the second part being extracting specific fields like company name, address and the total amount on the bill. The results are presented for each part in the following section. For demonstration purposes the following publicly, available invoice is chosen [9]. It is to be noted that the invoice in Figure 3(a) can be an image taken by a smartphone or a computer-generated invoice. The image in Figure 3(b) is obtained after performing the image preprocessing operations. The white regions in Figure 3(b) are contours detected. Using OCR engine text is now detected in each one of these cells and for the purpose of data extraction from tables only the part of the image containing the table is considered. By default, Tesseract expects a page of text when it segments an image. However, as OCR was to be applied on a small region, different page segmentation modes were experimented on. page segmentation mode (PSM) 4 and PSM 6 gave the best results and PSM 6 was used to extract text from the annotation as they resembled a uniform block of text which matched the configuration expected by Tesseract with PSM 6. After running Tesseract on table, we get the output as Figure 4.

Finally in the web application for the user who uploads the invoice the results are displayed as can be seen from Figure 5(a) and Figure 5(b). Figure 5(a) shows the actual image which was extracted from the PDF. Figure 5(b) shows the tables extracted from these images. The system was tested on more than 25 invoices which were captured with varying background images and in varying shades of light. Few invoices were computer generated whereas images of few invoices were clicked on smartphone devices. Using the confidence score given by Tesseract the accuracy of extraction of products from tables was found to be between 80% and 95%.

The accuracy of the Tesseract engine is obtained in the form of confidence score which is given as output by the python wrapper (PyTesseract) for Tesseract. The wrapper returns a list of bounding boxes, the confidence score (accuracy) for each bounding box, detected text within each box when a call is made to the `image_to_data` function. However, when the results of extraction were manually tested it was found that the accuracy score on average was more than 85%. Text extraction on the entire invoice gave similar results with

the average accuracy being around 85%. It was observed that accuracy was significantly higher for computer generated invoices and was on the lower end for the invoices which weren't generated by machines. The code repository with contents details as well as the steps to setup, debug and run the program is available at [26]. Following this will lead to a web application where the features discussed in the paper are offered in the tool.

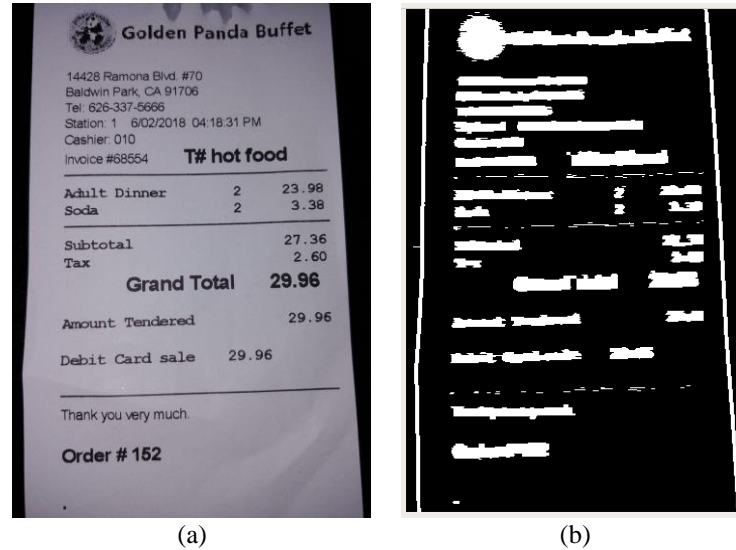


Figure 3. Sample invoice under consideration for (a) demonstration purposes and (b) the image obtained after converting to grayscale, performing binarization, applying morphology gradient, thresholding using Otsu binarization, finding and drawing the contours

```
{ "Page1": { "Name": (773, 442, 2118, 516), "GST number": (861, 544, 1736, 601)}, "ncols": "0"}
{ "Name": "amgalore Electricity Supply Company Limited wholly Owned Government of Karnataka Undertaking -", "GST number": "BESCOM GoIN NO! 2gvAACUDT4a 12 0140 Office of the Acct Executive Engineer Ele., C, O&M Sub-divis"}
["tables\\TABLE 1.jpg", "tables\\TABLE 2.jpg", "tables\\TABLE 3.jpg", "tables\\TABLE 4.jpg", "tables\\TABLE 5.jpg"]
```

Figure 4. The tables recognized and the text extracted from the regions of interest in the template is shown

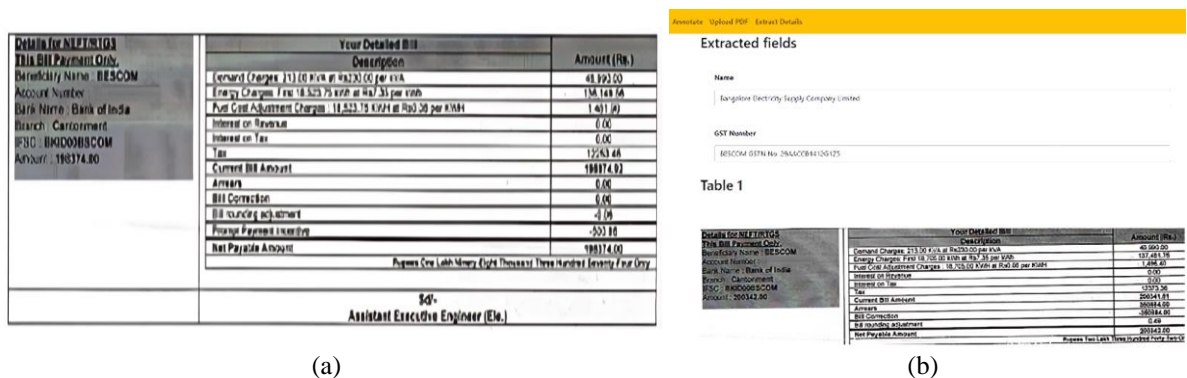


Figure 5. Subsection of the invoice and the corresponding text details obtained from the system (a) actual image with the bounding boxes around the regions of interest and (b) OCR performed on the regions of interest and the tables detected by OpenCV based algorithms.

5. CONCLUSION

Computer vision has found many applications in various fields. In this paper one such application of computer vision in the field of text recognition and extraction was presented. A system is presented which involves a web application at the frontend, OpenCV and a fine-tuned Tesseract OCR engine to perform text extraction at the backend. Algorithms used in image preprocessing for the invoices are also discussed. With

the help of OpenCV background noises are removed and the lucidity of the pictures is improved. Image preprocessing techniques have been applied in order to improve the image quality to get accurate results. List of products were extracted separately and rest of the fields are based on the user's needs. After the fine tuning of the base Tesseract model, the system is able to recognize the special characters and symbols with an accuracy of 90%. The system was tested on more than 25 types of invoices and gave promising accuracy scores with the average score lying between 85% and 95%.

ACKNOWLEDGEMENTS




We are indebted to our guide, Dr. Pratiba D, Assistant Professor, Department of Computer Science and Engineering for her wholehearted support, suggestions and invaluable advice throughout our proposed work and also help in the preparation of this article. We also express gratitude to Dr. Shobha G, Professor, Department of Computer Science and Engineering for her valuable comments and suggestions. We express sincere gratitude to our beloved Principal, Dr. K. N. Subramanya for his appreciation towards our work.

REFERENCES




- [1] G. Bradski, "The OpenCV library," *Dr. Dobbs's Journal of Software Tools*, vol. 120, pp. 122–125, 2000.
- [2] S. Deshpande and R. Shriram, "Real time text detection and recognition on hand held objects to assist blind people," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICADOT)*, Sep. 2016, pp. 1020–1024. doi: 10.1109/ICADOT.2016.7877741.
- [3] X. Zhou *et al.*, "EAST: an efficient and accurate scene text detector," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2642–2651. doi: 10.1109/CVPR.2017.283.
- [4] K. Karthick, K. B. Ravindrakumar, R. Francis, and S. Ilankannan, "Steps involved in text recognition and recent research in OCR; a study," *International Journal of Recent Technology and Engineering*, vol. 8, no. 1, pp. 3095–3100, 2019.
- [5] J. Yuan, H. Li, M. Wang, R. Liu, C. Li, and B. Wang, "An OpenCV-based framework for table information extraction," in *2020 IEEE International Conference on Knowledge Graph (ICKG)*, Aug. 2020, pp. 621–628. doi: 10.1109/ICKG50248.2020.00093.
- [6] R. C. Gonzalez, R. E. Woods, and S. L., "Morphological image processing," 2004.
- [7] "MATLAB, version 7.10.0," in *Natick, Massachusetts: The MathWorks Inc.*, 2010.
- [8] C. Bartz, H. Yang, and C. Meinel, "STN-OCR: a single neural network for text detection and text recognition," 2017.
- [9] "ICDAR 2019 robust reading challenge on scanned receipts OCR and information extraction," *ICDAR*, 2019. <https://drive.google.com/drive/folders/1ShItNWXYiY1tFDM5W02bceHuJjyeeJl2> (accessed Jan. 12, 2020).
- [10] D. Baviskar, S. Ahirrao, and K. Kotecha, "Multi-layout unstructured invoice documents dataset: a dataset for template-free invoice processing and its evaluation using AI approaches," *IEEE Access*, vol. 9, pp. 101494–101512, 2021, doi: 10.1109/ACCESS.2021.3096739.
- [11] S. Marinai, M. Gori, and G. Soda, "Artificial neural networks for document analysis and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 23–35, Jan. 2005, doi: 10.1109/TPAMI.2005.4.
- [12] W. Liu, C. Chen, K.-Y. Wong, Z. Su, and J. Han, "STAR-Net: a spatial attention residue network for scene text recognition," in *Proceedings of the British Machine Vision Conference 2016*, 2016, pp. 43.1–43.13. doi: 10.5244/C.30.43.
- [13] H. Wang and Z. Zhang, "Text detection algorithm based on improved YOLOv3," in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, Jul. 2019, pp. 147–150. doi: 10.1109/ICEIEC.2019.8784576.
- [14] D. Baviskar, S. Ahirrao, and K. Kotecha, "Multi-layout invoice document dataset (MIDD): a dataset for named entity recognition," *Data*, vol. 6, no. 7, p. 78, Jul. 2021, doi: 10.3390/data6070078.
- [15] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 9357–9366. doi: 10.1109/CVPR.2019.00959.
- [16] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017, doi: 10.1109/TPAMI.2016.2646371.
- [17] B. Wang, J. Xu, J. Li, C. Hu, and J.-S. Pan, "Scene text recognition algorithm based on faster RCNN," in *2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS)*, Jun. 2017, pp. 1–4. doi: 10.1109/EIIS.2017.8298720.
- [18] R. Smith, "An overview of the tesseract OCR engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, Sep. 2007, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.
- [19] P. Soille, "Erosion and dilation," in *Morphological Image Analysis*, Springer Berlin Heidelberg, 1999, pp. 49–88.
- [20] D. Sporici, E. Cuşnir, and C.-A. Boiangiu, "Improving the accuracy of Tesseract 4.0 OCR engine using convolution-based preprocessing," *Symmetry*, vol. 12, no. 5, p. 715, May 2020, doi: 10.3390/sym12050715.
- [21] C. Clausner, A. Antonacopoulos, and S. Pletschacher, "Efficient and effective OCR engine training," *International Journal on Document Analysis and Recognition (IJ DAR)*, vol. 23, no. 1, pp. 73–88, Mar. 2020, doi: 10.1007/s10032-019-00347-8.
- [22] D. Baviskar, S. Ahirrao, V. Potdar, and K. Kotecha, "Efficient automated processing of the unstructured documents using artificial intelligence: a systematic literature review and future directions," *IEEE Access*, vol. 9, pp. 72894–72936, 2021, doi: 10.1109/ACCESS.2021.3072900.
- [23] N. George, "Models," in *Mastering Django Core*, 1 st., GNW Independent Publishing, 2016.
- [24] Madmaze/pytesseract, "A Python wrapper for Google Tesseract," *GitHub*, 2022. <https://github.com/madmaze/pytesseract> (accessed Sep. 20, 2022).
- [25] Tesseract-ocr/tessdata, "Trained models with support for legacy and LSTM OCR engine," *GitHub*, 2022. <https://github.com/tesseract-ocr/tessdata> (accessed Mar. 26, 2022).
- [26] Snitin08/Annotate_AIMS, "Automated Invoice management system (AIMS)," *GitHub*, 2021. https://github.com/snitin08/Annotate_AIMS (accessed Nov. 17, 2021).

BIOGRAPHIES OF AUTHORS






Akanksh Aparna Manjunath    is currently pursuing a B.E. degree in computer science engineering from RV College of Engineering, Bengaluru, India (2018-2022). His research interests include web development and development of deep learning models for computer vision. He can be contacted at email: akanksham.cs18@rvce.edu.in.






Manjunath Sudhakar Nayak    is currently pursuing a B.E. degree in computer science engineering from RV College of Engineering, Bengaluru, India (2018-2022). His research interests include web development and deep learning. He can be contacted at email: manjunathsnayak.cs18@rvce.edu.in.






Santhanam Nishith    is currently pursuing a B.E. degree in computer science engineering from RV College of Engineering, Bengaluru, India (2018-2022). His research interests include machine learning and deep learning. He can be contacted at email: snishith.cs18@rvce.edu.in.






Satish Nitin Pandit    is currently pursuing a B.E. degree in computer science engineering from RV College of Engineering, Bengaluru, India (2018-2022). His research interests include web development and machine learning. He can be contacted at email: snitinpandit.cs18@rvce.edu.in.






Shreyas Sunkad    is currently pursuing a B.E. degree in computer science engineering from RV College of Engineering, Bengaluru, India (2018-2022). His research interests include cloud computing. He can be contacted at email: shreyass.cs18@rvce.edu.in.



Dr. Pratiba Deenadhayalan    is working as an Assistant professor in the Computer Science and Engineering department at RVCE. She received her Ph. D degree from VTU. She has published over 40 research papers. She can be contacted at email: pratibad@rvce.edu.in.



Dr. Shobha Gangadhara    Professor, Computer Science and Engineering Department, RV College of Engineering, Bengaluru, India has a teaching experience of 26 years, her specialization includes data mining, machine learning and image processing. She can be contacted at email: shobhag@rvce.edu.in.