AUTOMATIC FEATURE EXTRACTION FROM BANK CHEQUES

A CAPSTONE PROJECT REPORT-14IT7C0

Submitted by

KALAIVANI KTL (16IT041)

MEENDADEVI K (16IT052)

SRIVIDHYA R (16IT105)

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI – 15

(A Government Aided ISO 9001 : 2008 Certified Autonomous Institute Affiliated to Anna University)

ANNA UNIVERSITY::CHENNAI 600 025

NOVEMBER 2019

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI-15

(A Government Aided ISO 9001 : 2008 Certified Autonomous Institute Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report **“AUTOMATIC FEATURE EXTRACTION FROM BANK CHEQUES”** is the bonafide work of **“K.T.L.KALAIVANI (16IT041), K.MEENADEVI (16IT052), R.SRIVIDHYA (16IT105)”** who carried out the project work under my supervision during the academic year 2019-2020.

SIGNATURE

Dr. C. DEISY, M.E., Ph.D.,

PROFESSOR &

HEAD OF THE DEPARTMENT

DEPARTMENT OF INFORMATION TECHNOLOGY,
THIAGARAJAR COLLEGE OF ENGINEERING,
MADURAI- 625 015.

SIGNATURE

MS. K.V. UMA, M.E.,

SUPERVISOR &

ASSISTANT PROFESSOR

DEPARTMENT OF INFORMATION TECHNOLOGY,
THIAGARAJAR COLLEGE OF ENGINEERING,
MADURAI- 625 015.

ACKNOWLEDGEMENT

We express our sincere gratitude to our Honorable Chairman and Correspondent **Mr.KarumuthuT.Kannan** for his encouragement and guidance in our academic life at Thiagarajar College of Engineering. We wish to express our profound gratitude to our Honorable Principal **Dr.V.Abhai Kumar** for his overwhelming support provided during our course span in Thiagarajar College of Engineering.

We are grateful to our Department head **Dr.C.Deisy,** for the academic support extended to our project. We thank our project supervisor **Ms.K.V.Uma,** for the successful completion of the project through guidance and led us from the scratch of the project's commencement with undivided attention and skillful expertise in developing this project in a systematic and professional manner.

We express our sincere thanks to all the staff members and also the lab technicians of the Department of Information Technology for their constant and dedicated service to brighten our career.

We thank God Almighty and our parents for helping us procure such a challenging and an interesting project, and in completing the same in due course without much difficulty.

KALAIVANI KTL (16IT041)
MEENDADEVI K (16IT052)
SRIVIDHYA R (16IT105)

ABSTRACT

A bank is a financial institution which is involved in borrowing and lending money. Banks take customer deposits in return for paying customers an annual interest payment. The bank then uses the majority of these deposits to lend to other customers for a variety of loans. Banks play an important role in the economy for offering a service for people wishing to save. Banks also play an important role in offering finance to businesses who wish to invest and expand. These loans and business investment are important for enabling economic growth. So time management is very important in banking sectors. But manual reading of bank cheque is really a time consuming task. Our approach is to automatically extract the details from the cheque images based on the image processing and feature extraction. Features play a very important role in the area of image processing. Before getting features, various image processing techniques are applied on the input bank cheque images. After that various feature extraction techniques are applied to get features that will be useful for payment processing. To implement such system, various image processing pattern recognition and OCR techniques must be involved.

Keywords : feature extraction, preprocessing, cheque, text, recognition

.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The aim of our project is to extract the information from the cheques. In the cheque, there are different data fields such as name, amount, signature etc. Our idea is to extract these details from the cheque and save in excel sheet.

1.2PROBLEM STATEMENT

Our project title is **"AUTOMATIC FEATURE EXTRACTION FROM BANK CHEQUES".** It is supposed to extract information from different data fields on the cheque such as amount, payee, date and signature. After applying feature extraction and image processing techniques, it can be used for payment processing. By doing this, most of the time gets saved and made people to engage in other important tasks. This system can also be used for reading bank deposit slips, forms, data entry, postal address reading, script recognition etc.

Bank transactions involving cheques are still increasing throughout the world in spite of the overall usage of electronic payment methods. Manual reading and storing the details of bank cheque is really a time consuming task. Our idea is to automatically extract the details from the scanned copy of bank cheques based on the image processing and feature extraction.

1.2 SOCIAL RELEVANCY

The processing and manual verification of bank cheques currently requires a large investment in human resource and it is also a time consuming task. This automation system may achieve better performance and allow the reallocation of human resources to other tasks. Also, it saves the time of the bank employees. Hence, this system has a positive impact in banking sector.

1.3 OBJECTIVES

In this system, the scanned copy of the bank cheque which is either in .png format or .jpg format is taken as a input. From this scanned copy of bank cheque, the main objective is to obtain the most relevant information from the original data and represent the information in a lower dimensionality space using feature extraction. The extracted information from the cheque is able to saved in a excel sheet either in .xls or .csv format or stored in a database.

1.4 SCOPE

The scope of the designed system works fine on scanned images of the cheques having the following specifications:

- Width : 202 mm
- Height : 93 mm
- Pixels : 692*320

The system designed to extract the features such as payee, account number, amount in words, amount in numbers and date from the scanned copies of the cheques that are in printed English alphabets. The proposed system identifies the text present in the scanned images of the cheque using pytesseract. Text includes numbers, lower case characters, upper case characters, and toggle case characters. Hence, the system designed to extract the features from digital cheques. Also, the system designed to extract the above stated features from demand drafts.

2. LITERATURE REVIEW

Table - 2.1 Detailed literature review

S. No	**Title (Journals)**	**Source with date**	**Inference (Problems addressed, Solution discussed, Pros & cons of solution)**
1	A Survey on free hand sketch recognition and retrieval	Source – Elsevier Author – Xianlin Zhang , Xueming Li, Yang Liu, Fangxiang Feng Year - 2019	This system mainly aims to provide exact description of free hand sketch images. The steps followed in this method includes pre-processing, matching, segmentation and representation. In pre-processing, the Gray-Scale level adjustment for noise removal is carried out which is then followed by Gray-Scale Transformation to process the input image. Once mapping is done with the input image, three-dimensional image is converted into two-dimensional image. The two-dimensional image is subjected to segmentation and it can be regarded as a process of sketch classification using deep learning models. The segmented image is used for template matching. SBIR algorithm is used for global feature detection. The classical approach for local feature detection uses linear matched filtering, i.e., it convolves the

			image with a template of the desired feature. Matching processes such as correlation are computationally costly and are also sensitive to geometrical distortion. Hence, some of the features could not be retrieved due to geometrical distortion in the process of Gray-Scale Transformation .
2	Correlation Based Template Matching For Recognition Of Bank Cheque Number	Source – IEEE Author – Raghavendra SP, Ajit Danti, Suresha M Year - 2018	This system uses OCR for cheque number information extraction. Static images are taken and the characters in it are identified as a concentrated block. Based on matching process, the input block of characters and the existing samples are compared from the repository and the result is produced based on calculations. Major steps includes pre-processing the input image which states the input is valid or not and then identifying change in positions, standard size, considering many resolutions, physical changes, Zonal characters, Measuring distances, features related to ends and edges are highlighted. Once the acknowledgment phase starts , it uses usual methods of matching, using methods related to

			stats, obtaining the traces of images. In the next step, it uses scale-space feature extractor to remove unwanted gaps witnessed in input image extracted from the text obtained in the video. After that, the authenticated text is being compared with each of the text frame by frame. Since the characters are not of the expected format and especially in expected proportion. So as in this paper, it includes the entire cross verifications performed for such characters with the challenging task of Identification of Hand written characters in cheques. Thus the accuracy of the system with handwritten text is low as compared to the digital or printed characters
3	Automatic processing of handwritten bank cheque images	Source – Springer Author – R. Jayadevan · S. R. Kolhe · P. M. Patil · U. Pal Year - 2018	The basic goal of this system is to develop a fast and accurate courtesy amount recognition , legal amount recognition, date recognition and Signature verification recognition in bank cheques and the system which is able to read the text automatically with minimal errors. The text (both alphabets and numerals) recognition

			process has been carried out using OpenCV and Tesseract based classification technique. As the input to the system is a scanned image of the bank cheque , pre-processing steps include binarization, skew and slant correction and normalization, different image processing techniques of image segmentation and image enhancement such as dilation, erosion, contour finding and thresholding are used. OCR machine is used to recognize the characters present in the bank cheque . The final output of the bank cheque was found to be very close to the actual value on the bank cheques. However, some errors occur in depending upon the quality of the original image, the lighting, the clarity and general condition of the bank cheque. Structural complexity of characters and variability of writing styles are the biggest challenges in recognizing legal amounts present on a bank cheque.
4	Text detection and recognition in raw	Source – Sciencedirect	This system detects and recognizes seven segment numerals from

	image dataset of seven segment digital energy meter display	Author - Karthick Kanagarathinam, Kavaskar Sekar Year - 2019	collected samples of energy meters that may be helpful in reducing the cost of advanced metering infrastructure (AMI). The images have been captured under day and night light conditions. The recognition of text from seven segment display in energy meters has been carried out using dataset under the challenging text recognition conditions like tilted position, blurred, day and night light captured images. MSER and labeling method based OCR algorithm has been used for text detection and recognition. MSER algorithm is used to detect the text regions in an image. MSER detection is a common task performed on unstructured scenes. The regions are defined solely by an extremal feature of the intensity function in the region and on its outer boundary. The OCR function available with MATLAB should be trained to increase the recognition rate. The disadvantage of this system is that it has been implemented using matlab which is not as accurate as that of a system implemented in python.

5	Text recognition for the defects extraction in sewers CCTV inspection videos	Source – sciencedirect Author - L. Minh Dang, Syed Ibrahim Hassan, Suhyeon Im, Irfan Mehmood, Hyeonjoon Moon Year - 2018	This system is an automated framework for analyzing and tracking sewer inspection close- circuit television (CCTV) videos. It supports the off-site examination and quality management process of the videos and enables efficient revaluation of CCTV videos to extract sewer condition data. It includes two main modules: text recognition and cracks extraction. In the first module, multi-frame integration (MFI) was applied to reduce the background complexity, time and computational requirements needed for the video processing. Then maximally stable extremal regions (MSER) has been used on the grayscale channel and HSV channel to effectively detect all the text edges. Saturation color channel has been applied to verify the detected text line and remove false alarms. In the second module, by utilizing the text information on each frame, the operator's operation during the inspection is simulated which would indicate valuable clues about the location and severity of the cracks.

6	Detection of text in the natural environment	Source – Sciencedirect Author - Leena Mary Francis , N. Sreenath Year - 2019	This system uses a deep-learning based approach, to detect the presence of text in the natural scene images. This system is built with the capability to distinguish text and non-text images from the live-stream of the smart-phone camera, thereby eliminating the need for capturing the image, to locate the presence of text. A streamlined Convolutional Neural Network (CNN) MobileNet is used for the process of distinguishing text images from non-text images. This system consists of several stages like image cleansing, image preprocessing, segmentation and character recognition. The first major problem of this classification is that apart from the focused 62 classes (a to z;A to Z; 0 to 9), all other objects are non-text . Non-text objects embody a very large number of sub-classes which varies from non-living objects like the traffic light, box, bottle, phone etc., to living objects like the cat, dog, human etc. So training the classifier over such

			vast variety of objects becomes really troublesome. The second issue is the live detection of text objects from the natural scene.
7	Contour Restoration of Text Components for Recognition in Video/Scene Images	Source – Published in: IEEE Transactions on Image Processing (Volume: 25 , Issue: 12 , Dec. 2016	This system recognize the text in video/natural scene images which is the more complexing task than recognize the text in plain background images. In this paper, the main aim is to restore complete character contours in video/scene images from gray values, In this system it **consider edge images/binary information as inputs for text detection and recognition.** Here the importance of zero crossing points given by the Laplacian is utilized to identify stroke candidate pixels (SPC). For each SPC pair, new symmetry features based on gradient magnitude and Fourier phase angles to identify probable stroke candidate pairs (PSCP) is proposed. The same symmetry properties are proposed at the PSCP level to choose seed stroke candidate pairs (SSCP). Finally, an iterative algorithm is proposed for SSCP to restore complete character

			contours. Experimental results on benchmark databases.
8	Research on English translation distortion detection based on image evolution	Source – EURASIP Journal on Image and Video Processing volume 2019, Article number: 25 (2019)	This study improves the traditional algorithm, uses the Canny edge detection method as the edge detection method through experimental comparison and analysis, and combines the image evolution to analyze the English character translation of multiple complex images to alleviate the problem which is the distortions in the translation of image English characters.
9	Multilingual Character Segmentation and Recognition Schemes for Indian Document Images	Source – IEEE Access (Volume: 6) year – 2018 Author - Parul Sahare ; Sanjay B. Dhok	The multilingual documents generally suffer from their layout organizations, local skews, and low print quality and contain intermixed texts (machine-printed and handwritten). In the proposed character segmentation algorithm, primary segmentation paths are obtained using structural property of characters, whereas overlapped and joined characters are separated using graph distance theory. For recognizing input character **K-Nearest neighbour** algorithm is used. Finally, segmentation results are

			validated using highly accurate **support vector machine classifier.** For the proposed character recognition algorithm, three new geometrical shape-based features are computed. By using this technique experiments are carried out in different databased containing both printed text and hand written texts.
10	Multi-Script-Oriented Text Detection and Recognition in Video/Scene/Born Digital Images	Source Published in: IEEE Transactions on Circuits and Systems for Video Technology (Volume: 29 , Issue: 4 , April 2019)	Achieving good text detection and recognition results for multi-script-oriented images is a challenging task. This system explore bit plane slicing in order to utilize the advantage of the most significant bit information to identify text components. A new iterative nearest neighbor symmetry is then proposed based on shapes of convex and concave deficiencies of text components in bit planes to identify candidate planes. Further, a new concept called mutual nearest neighbor pair components based on gradient direction to identify representative pairs of texts in each candidate bit plane. The representative pairs are used to restore

			words with the help of edge image of the input one, which results in text detection results (words). Extract features in contourlet wavelet domain to detect characters with the help of an SVM classifier. The proposed method is evaluated on standard databases.

3. METHODOLOGY

3.1 PROPOSED SOLUTION

In this approach, a manual bank cheque database is constructed for feature extraction. Along with the standard database, some other cheque images downloaded from the internet and demand drafts are also considered. The proposed methodology has four stages.

- Preprocessing of each cheque to enhance the dataset
- Segmentation of each cheque to extract the region of interest
- Text detection and extraction using pytesseract
- Capturing the details into excel file in CSV format

In this proposed work, the design of automated feature extraction from bank cheque system, it is possible to extract the features such as payee, account number, amount in words, amount in rupees and date from the scanned cheque image of deposited cheque with the help of inexpensive scanners. When a user deposits a cheque in the bank, the bank officials need to make manual entry about the details present in the cheques. Manual entry is a time consuming task. Automatic feature extraction from cheque is a very difficult process. Humans use object recognition and contextual information to identify the features of an image. Unfortunately, the state-of-the-art computer vision techniques still cannot infer the high-level knowledge abstraction of the objects in the real world. The alternative method is to exploit the low-level features from the images for feature extraction.

We assume that the input image is restricted to only four possible rotations that are multiples of 90 , i.e., the photograph which is scanned can differ from its correct orientation by 0 (no rotation), 90 , 180 , or 270 . This is reasonable since cheque papers placed on a scanner are usually aligned with horizontal or vertical boundaries of the scanner plate. In this solution, we present a heuristic method for automatic feature extraction from bank cheque image. This method is based on image processing and text extraction. The marginal regions of each cheque image are used to extract the low level regional colour features in terms of the first and second colour moments in the HSV colour space. As a result of the proposed method, low-level features such as width, height, colour, RGB values and high-level features such as payee, account number, amount in words, amount in rupees and date from the scanned cheque image are automatically detected.

The noise removal from the bank cheque is an essential part for its processing. The notified point after the execution of background elimination is the physical integrity of remaining information in the cheque image. This specifies that the background can be eliminated with simple thresholding techniques. Image binarization identifies and eliminates many occurrences of text or images that are on the back of the document but that show up on the front from the relevant peeks. In this, we eliminate not only the background but all the types of noises, by a morphological subtraction scheme between the marginal boundaries. Running Gaussian average, Temporal median filter, Mixture of Gaussians, Kernel density estimation (KDE),Sequential KD approximation, Co-occurence of image variations and Eigenbackgrounds are some other methods that can be carried out to remove noise from the scanned bank cheque image.

In standings of image processing and computer vision background subtraction is also known as foreground extraction. From the given bank cheque image the background is estimated, by applying the blob analysis and the background is subtracted from the original image. The resulted image is binarized with well-defined threshold through global thresholding technique.

After preprocessing, it is necessary to perform the extraction operation on different fields for recognition. Generally in India, payee, account number, amount in words, amount in rupees and date in bank cheques are always located in the same region. A box is detected by identifying the cross-section points where co-ordinates of horizontal and vertical lines meet. Thus, this method is used for the identification of necessary fields in bank cheques. For each of the resulting sub-images, the background pattern is eliminated by a subtraction operation; the machine printed character strings in the identified fields are identified by running optical character recognition (OCR) in background. Pytesseract supports in the text detection and extraction of the features with utmost accuracy.

The extracted low-level features such as width, height, colour, RGB values and high-level features such as payee, account number, amount in words, amount in rupees and date from the scanned cheque image dataset are captured into an Excel file in CSV format for further payment processing.

3.1.1 APPLICATIONS OF TEXT EXTRACTION

Text extraction can be used for:

- Data entry for business documents, e.g. check, demand drafts, passport, invoice, bank statement and receipt
- Automatic number license plate recognition
- Automatic insurance documents key information extraction
- Extracting business card information into a contact list
- More quickly make textual versions of printed documents, e.g. book scanning
- Make electronic images of printed documents searchable

- Assistive technology for blind and visually impaired users

3.2 DIAGRAMS

3.2.1 FLOW DIAGRAM

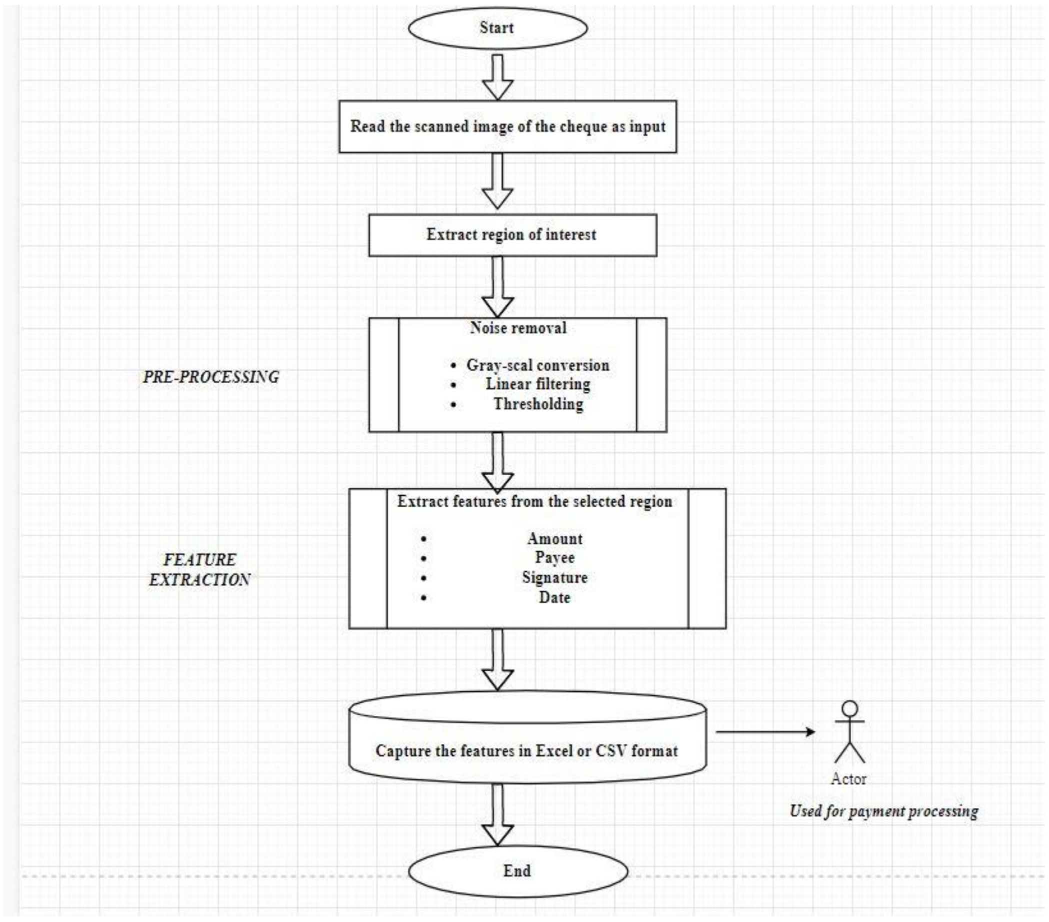

Start
Read the scanned image of the cheque as input
Extract region of interest
Noise removal
Gray-scal conversion
Linear filtering
Thresholding
PRE-PROCESSING
Extract features from the selected region
Amount
Payee
Signature
Date
FEATURE EXTRACTION
Capture the features in Excel or CSV format
Actor
Used for payment processing
End

Figure 3.2.1 Flow diagram

3.2.2 BLOCK DIAGRAM

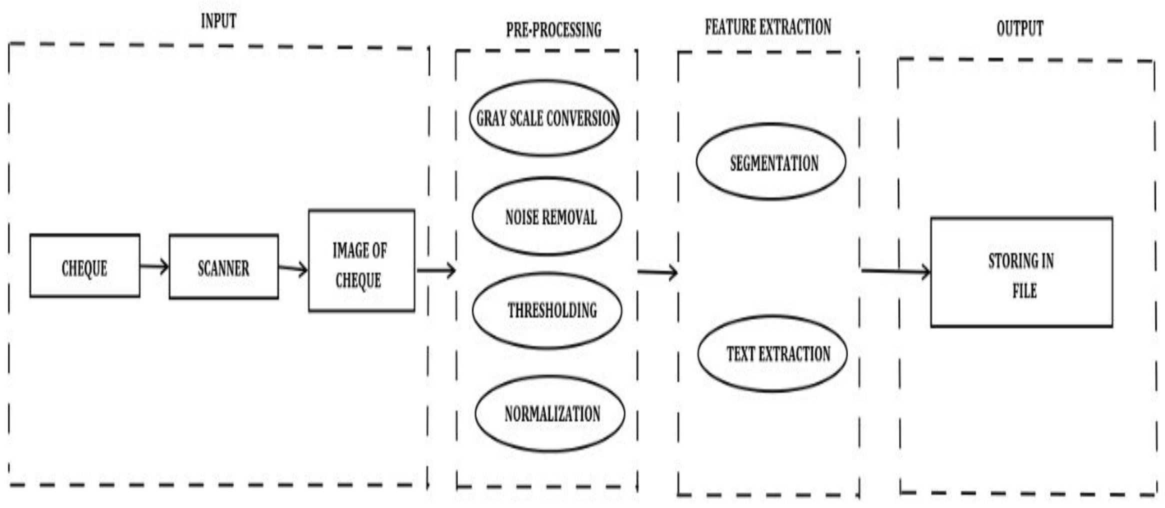
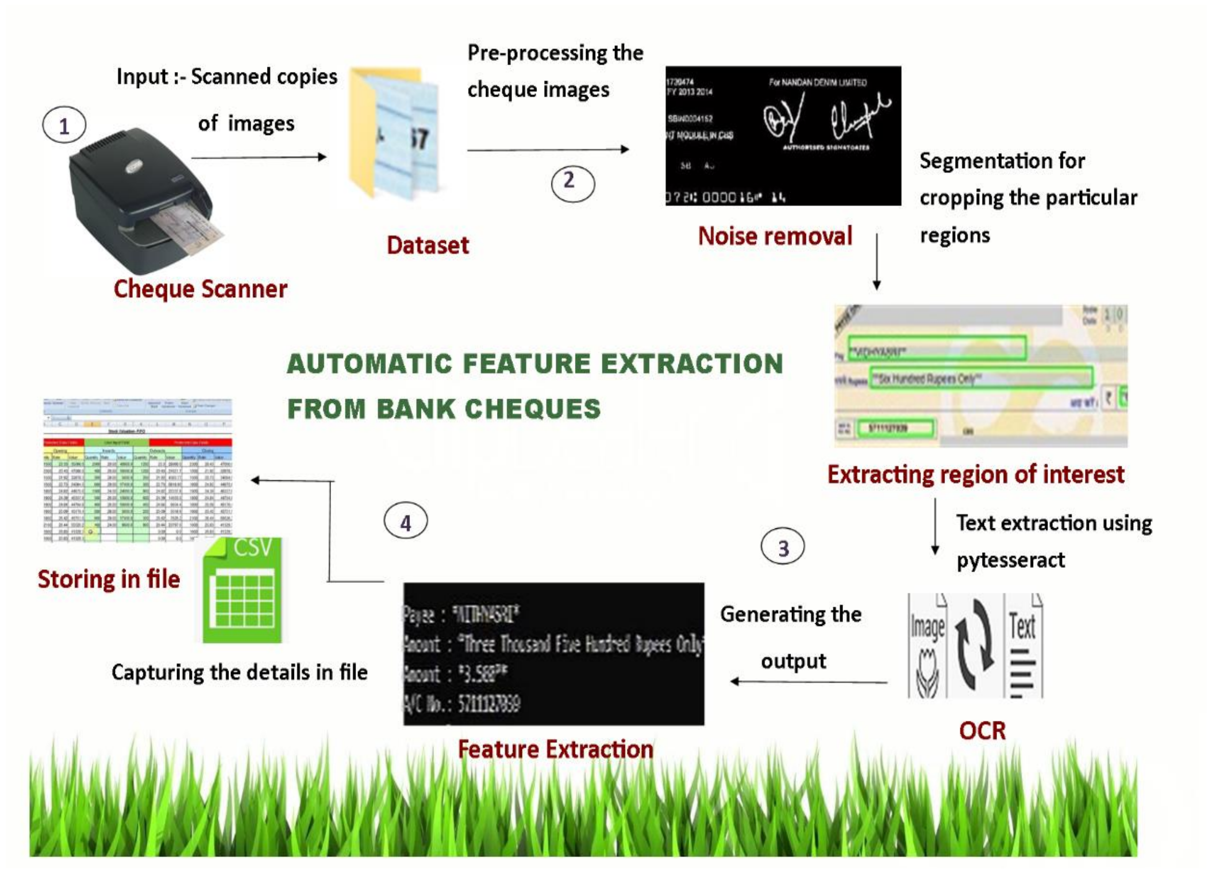


Figure 3.2.2 Block diagram

3.2.3 HIGH LEVEL DIAGRAM


1
Input :- Scanned copies of images
Cheque Scanner
Dataset
Pre-processing the cheque images
2
Noise removal
Segmentation for cropping the particular regions
AUTOMATIC FEATURE EXTRACTION FROM BANK CHEQUES
Extracting region of interest
Text extraction using pytesseract
3
Image
Text
OCR
Generating the output
4
Feature Extraction
Storing in file
CSV
Capturing the details in file

```
Payee : *NITHYASRI*
Amount : *Three Thousand Five Hundred Rupees Only*
Amount : *3.500/-*
A/C No.: 5711127839
```

Figure 3.2.3 High level diagram

3.3 PROCEDURE

- Analysed the bank cheques data set and identified the static components present in the images.
- Pre-processing is done to remove noise from the images.
- Extracted particular region of interest based on the co-ordinate values or boundary values.
- Detecting the characters and its numerals present in the particular region of interest and tried to extract its features.
- Identified and extracted portion of image having data fields on the cheque such as amount, payee, date and signature and capture the details in to CSV file for further payment processing.

Modules

1. Extracting the region of interest
2. Pre-processing
3. Feature Extraction
4. Exporting the result

3.3.1 EXTRACTING THE REGION OF INTEREST

Segmentation of the region of interest will be done by removing the unnecessary connected regions by applying second level cropping and dividing the region into 5 equal sub regions again giving sub region1, sub region2, sub region3, sub region4 and sub region5 respectively. Finally these sub regions will be considered as a potential region of interest for the next steps. Once the region of interest is made noise free using preprocessing, the next stage will be able to fetch extract the features such as payee, account number, amount in words, amount in rupees and date surrounded by cropping operation, which means to find the coordinates of text regions with respect to all four coordinates viz., uppermost, bottommost, leftmost and rightmost.

Input :- Bank cheque image

Output :- Extracted image

Steps :-

- Get the reference points (x-min, y-min, x-max, y-max)
- With the reference points, a rectangle will be drawn around the region of interest.
- Perform the cropping operation.
- Save the cropped image.

3.3.2 PRE-PROCESSING

One of the challenges faced in the adoption of image based cheque recognition is to ensure that high quality cheque images are needed for feature extraction. If the image is of low quality, then the proposed system may not be able to process the image for text detection as well as extraction and may result in low performance of the system. As the cost of preprocessing bank cheque is multi-fold compared to a normal processing, it is desirable to minimize the incidences of bad cheque-image quality. ‘3i-Infotech’ has developed an image quality assurance (IQA) validation engine, which is a standalone tool that can be used for performing IQA on any cheque image. Some of the image quality attributes considered are: ‘partial image’, ‘excessive image skew’, ‘piggyback image’, ‘image too light or too dark’, ‘streaks and/or bands on the image’, ‘below minimum image size and above maximum image size’. Silver Bullet’s ‘Ranger-IQA’ software is also used for cheque-image quality assurance during the image capturing step. This allows many cheque-image quality problems to be detected at the source before truncation. The quality attributes considered include the following: ‘undersize image’, ‘oversize image’, ‘below minimum compressed image size’, ‘above maximum compressed image size’, ‘excessive document skew’, ‘image too light’, ‘image too dark’, ‘horizontal streaks present in the image', 'folded or torn document corners and edges', 'document framing error', 'excessive spot noise in the image', 'front-rear image dimension mismatch' and 'carbon strip detected'. Image quality assurance can be done on the dataset to provide better quality of the bank cheque images.

Instead of performing preprocessing the whole bank cheque image dataset, we can perform noise removal only in the region of interest. The extracted region of interest from the previous step is fed as input to this stage and gray scale conversion is carried out for noise removal. The following steps are discussed regarding the preprocessing of bank cheque images. Consider a 24 bit RGB bank cheque colour image $f(x, y)$ and convert it into gray then into binary format $g(x, y)$. Noise removal will be done by removing the unnecessary pixels and small regions with area 4 times less than the characters to be clipped using region properties of all connected components. Fix a threshold and remove all those regions which are less than the given threshold.

Input :- Cropped image

Output :- Image free of noise

Steps :-

- Gray scale Conversion – Converting the cropped image into a gray scale image (RGB values = 24 bits to gray scale value = 8 bits).
- Dilation – Used to highlight the features in the gray scale converted image.
- Erosion – Erodes away the boundaries of foreground object.
- Thresholding – Used to create binary image.

3.3.3 FEATURE EXTRACTION

The accuracy in recognizing the text plays a big role in the recognition accuracy of the various fields which are in both numerals and strings. After successful segmentation and preprocessing of individual fields from the scanned image of the bank cheques, they have to be correctly recognized to get the details of the cheque. Pytesseract is used for text detection and extraction. It is an optical character recognition tool for python which will recognize the text embedded in the images. This tool support for Unicode character recognition and it is used to recognize more than 100 languages out of the box. In the proposed system, the scope is restricted to English language only. The algorithm runs in the background and extract the features.

Input :- Pre-processed image

Output :- Details of cheque

Steps :-

- Properties of the image – Finding the properties of image such as width, height, font size, colour of the cheque and RGB value.
- Segmentation – Partitions an image into distinct regions containing each pixels with similar attributes.
- Text extraction – Extracting the data such as amount, payee, date, signature.

3.3.4 EXPORTING THE RESULT

The proposed system extracts low-level features such as width, height, colour, RGB values and high-level features such as payee, account number, amount in words, amount in rupees and date from the scanned cheque image dataset. Finally, these details of the bank cheques are captured into an Excel file in CSV format for further payment processing. In CSV, we can write column headers only once where as in Excel, we have to have a start tag and end tag for each column in each row. Importing CSV files can be much faster, and it also consumes less memory. It's easy to programmatically manipulate CSV since, they are simple text files. Hence, the extracted features are stored in file for future use.

Input :- Properties of image and text extracted from image

Output :- Storing details in excel

Steps :-

- Create an Excel file initially.
- Append the results at the end of spread sheet each time while executing.
- Save the results in the file.
- Ensure that the file is not opened while executing the application.

4. IMPLEMENTATION

4.1 PROGRAMMING LANGUAGE USED

Figure – 4.1 Python

Python - Python is a widely used general-purpose, high level programming language. Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a procedural way, an object-orientated way or a functional way. A lot of general computer vision libraries are readily available for python which will be useful for developing applications involving image processing techniques. As our project is based on text extraction from images of bank cheques, we have chosen python as programming language.

4.1.1 Applications of Python

- GUI based desktop applications
    - Image processing and graphic design applications
    - Scientific and computational applications
    - Games
- Web frameworks and web applications
- Enterprise and business applications
- Operating systems

- Language development
- Prototyping

4.1.2 Advantages/Benefits of Python

The diverse application of the Python language is a result of the combination of features which give this language an edge over others. Some of the benefits of programming in Python include:

1. Presence of Third Party Modules:

The Python Package Index (PyPI) contains numerous third-party modules that make Python capable of interacting with most of the other languages and platforms.

2. Extensive Support Libraries:

Python provides a large standard library which includes areas like internet protocols, string operations, web services tools and operating system interfaces. Many high use programming tasks have already been scripted into the standard library which reduces length of code to be written significantly.

3. Open Source and Community Development:

Python language is developed under an OSI-approved open source license, which makes it free to use and distribute, including for commercial purposes.

Further, its development is driven by the community which collaborates for its code through hosting conferences and mailing lists, and provides for its numerous modules.

4. Learning Ease and Support Available:

Python offers excellent readability and uncluttered simple-to-learn syntax which helps beginners to utilize this programming language. The code style guidelines, PEP 8, provide a set of rules to facilitate the formatting of code. Additionally, the wide base of users and active developers has resulted in a rich internet resource bank to encourage development and the continued adoption of the language.

5. User-friendly Data Structures:

Python has built-in list and dictionary data structures which can be used to construct fast runtime data structures. Further, Python also provides the option of dynamic high-level data typing which reduces the length of support code that is needed.

6. Productivity and Speed:

Python has clean object-oriented design, provides enhanced process control capabilities, and possesses strong integration and text processing capabilities and its own unit testing framework, all of which contribute to the increase in its speed and productivity. Python is considered a viable option for building complex multi-protocol network applications.

4.2 TOOL USED

ANACONDA®

Figure – 4.2 Anaconda IDE logo

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 15 million users and includes more than 1500 popular data-science packages suitable for Windows, Linux, and MacOS. Anaconda Distribution contains conda and Anaconda Navigator, as well as Python and hundreds of scientific packages. Conda works on command line interface such as Anaconda Prompt on Windows and terminal on macOS and Linux.

Packages available in Anaconda

- Over 250 packages are automatically installed with Anaconda.
- Over 7,500 additional open-source packages (including R) can be individually installed from the Anaconda repository with the conda install command.
- Thousands of other packages are available from Anaconda Cloud.
- We can download other packages using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in some cases they can work together. However, the preference should be to install the conda package if it is available.
- We can also make your own custom packages using the conda build command, and we can share them with others by uploading them to Anaconda Cloud, PyPi, or other repositories.

4.3 LIBRARIES USED

1. cv2

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

2. PIL

Python Imaging Library (abbreviated as **PIL**) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. Pillow offers several standard procedures for image manipulation. These include:

- Per-pixel manipulations,
- Masking and transparency handling,
- Image filtering, such as blurring, contouring, smoothing, or edge finding,
- Image enhancing, such as sharpening, adjusting brightness, contrast or colour,
- Adding text to images and much more.

Some of the file formats supported are PPM, PNG, JPEG, GIF, TIFF, and BMP. It is also possible to create new file decoders to expand the library of file formats accessible.

3. webcolors

webcolors is a Python (2.7, 3.5+) module for working with HTML/CSS color definitions.Support is included for normalizing and converting between the following formats (RGB colorspace only; conversion to/from HSL can be handled by the colorsys module in the Python standard library):

- Specification-defined color names
- Six-digit hexadecimal

- Three-digit hexadecimal
- Integer rgb() triplet
- Percentage rgb() triplet

webcolors is used in this project to identify the background colour of the cheque image.

4. OS

The OS module in Python provides a way of using operating system dependent

functionality. The functions that the OS module provides allows us to interface with the

underlying operating system that Python is running on – be that Windows, Mac or

Linux.

OS Functions

Executing a shell command

os.system()

Get the users environment

os.environ()

Returns the current working directory.

os.getcwd()

Return the real group id of the current process.

os.getgid()

Return the current process's user id.

os.getuid()

Returns the real process ID of the current process.

os.getpid()

Set the current numeric umask and return the previous umask.

os.umask(mask)

Return information identifying the current operating system.

os.uname()

Change the root directory of the current process to path.

os.chroot(path)

Return a list of the entries in the directory given by path.

os.listdir(path)

Create a directory named path with numeric mode mode.

os.mkdir(path)

Recursive directory creation function.

os.makedirs(path)

Remove (delete) the file path.

os.remove(path)

Remove directories recursively.

os.removedirs(path)

Rename the file or directory src to dst.

os.rename(src, dst)

Remove (delete) the directory path.

os.rmdir(path)

Split the pathname path into a pair (root, ext) such that root + ext == path, and ext is empty or begins with a period and contains at most one period. Leading periods on the basename are ignored; splitext('.cshrc') returns ('.cshrc', '').

os.path.splitext(path)

Join one or more path components intelligently. The return value is the concatenation of *path* and any members of **paths* with exactly one directory separator (os.sep) following each non-empty part except the last, meaning that the result will only end in a separator if the last part is empty. If a component is an absolute path, all previous components are thrown away and joining continues from the absolute path component.

os.path.join(*path*, **paths*)

5. openpyxl

Openpyxl is a Python library for reading and writing Excel 2010(with extension xlsx/xlsm/xltx/xltm) files. It was born from lack of existing library to read/write natively from Python the Office Open XML format. The openpyxl module allows Python program to read and modify Excel files.

6. pytesseract

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images. Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

Optical Character Recognition

Optical character recognition (OCR) algorithms allow computers to analyze printed or handwritten documents automatically and prepare text data into editable formats for computers to efficiently process them. It is another way to extract and leverage business-critical data.

Working of OCR

Human eyes naturally recognize various patterns, fonts or styles. For computers, it is hard work to do. Any scanned document is a graphics file, i.e., a pattern of pixels. A computer localizes, detects and recognizes characters on an image and turns the image of paper documents into a text file. Then, it becomes possible to extract meaningful information. Texts in a machine-readable form can then be used for different purposes. They can be scanned in search of patterns and vital data, used to generate reports, draw up charts and distributed into spreadsheets.

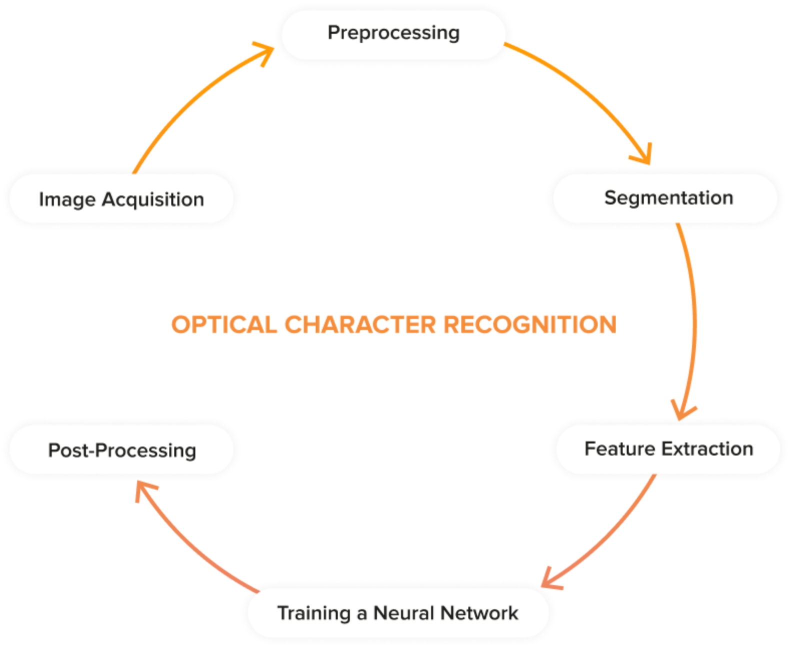


Figure – 4.3 OCR Workflow

1. Image Acquisition

The first step is to acquire images of paper documents with the help of optical scanners. This way, an original image can be captured and stored. Most of the paper documents are black and white, and an OCR scanner should be able to threshold images. In other words, it should replace each pixel in an image with a black or a white pixel. It is a method of image segmentation.

2. Preprocessing

The goal of preprocessing is to make raw data usable by computers. The noise level on an image should be optimized and areas outside the text removed. Preprocessing is especially vital for recognizing handwritten documents that are more sensitive to noise. Preprocessing allows obtaining a clean character image to yield better results of image recognition.

3. Segmentation

The process of segmentation is aimed at grouping characters into meaningful chunks. There can be predefined classes for characters. So, images can be scanned for patterns that match the classes.

4. Feature Extraction

This step means splitting the input data into a set of features, that is, to find essential characteristics that make one or another pattern recognizable. As a result, each character gets classified in a particular class.

5. Training a Neural Network

Once all the features are extracted, they can be fetched to a neural network (NN) to train it to recognize characters. A training dataset and the methods applied to achieve the best output will depend on a problem that requires an OCR-based solution.

6. Post-Processing

This stage is the process of refinement as an OCR model can require some corrections. However, it isn't possible to achieve 100% recognition accuracy. The identification of characters heavily depends on the context. The verification of the output requires a human-in-the-loop approach.

The usage of OCR makes it easy to meet in-house document standards, give a head start to workflow automation, fully or partially eliminate the need for paper workflow. High-level optical character recognition services can assist many mid- and large-scale companies to make a profit from using custom-tailored algorithms. Industries like banking and finance, healthcare, tourism, and logistics may benefit the most from the successful implementation of OCR.

4.4 ALGORITHM

4.4.1 Pseudo Code

```
begin
        import the necessary packages and libraries
        specify the directory and valid image extensions
        list all files in directory and append files with a vaild extention to
image_path_list
        loop through image_path_list to open each image and extract text from
each image
        begin loop
```

```
    for each image do
        open the image
        find the properties of image and store it in separate variables.
        initialize the list of reference points to crop the region of interest
        such as  payee, amount in words, amount and account number
        draw a rectangle around the region of interest based on reference
        points
        while ‘q’ key is not pressed do
            display the cropped image and wait for a key press
            if the ‘c’ key is pressed then
                save the cropped region of interest image
                break from the loop
            end if
        end while
        extract the text from the croppes images and store it in a variable
        write the properties and text extracted from cropped images to
        Excel file
    end for
    end loop
end
```

4.4.2 Code

```
# import the necessary packages

from xlwt import Workbook

from xlrd import open_workbook

import openpyxl

import argparse

import cv2

import numpy as np

import webcolors

import os, os.path

from PIL import ImageFont, ImageDraw, Image

import pytesseract

from pytesseract import image_to_string


#pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"


#debug info OpenCV version

print ("OpenCV version: " + cv2.__version__)


#image path and valid extensions

imageDir = "Data/" #specify your path here"
```

```
image_path_list = []
valid_image_extensions = [".jpg", ".jpeg", ".png", ".tif", ".tiff"] #specify your vald extensions here
valid_image_extensions = [item.lower() for item in valid_image_extensions]

#create a list all files in directory and
#append files with a vaild extention to image_path_list
for file in os.listdir(imageDir):
    extension = os.path.splitext(file)[1]
    if extension.lower() not in valid_image_extensions:
        continue
    image_path_list.append(os.path.join(imageDir, file))

#loop through image_path_list to open each image
for imagePath in image_path_list:
    #image = cv2.imread(imagePath)
    image = cv2.imread(imagePath, cv2.IMREAD_COLOR)
    #im=Image.open(image)
    #print(pytesseract.image_to_string(image,lang='eng'))
    print("\t\t\t===================================\n")
    print("\t\t\tFEATURE EXTRACTION FROM THE IMAGE\n")
    print("\t\t\t===================================\n")
    img = cv2.imread(imagePath, cv2.IMREAD_COLOR)                    # rgb
    alpha_img = cv2.imread(imagePath, cv2.IMREAD_UNCHANGED) # rgba
    gray_img  =  cv2.imread(imagePath,  cv2.IMREAD_GRAYSCALE)         #
grayscale

    print ("\nProperties of the Image")
    print ("*************************\n\n")
    print ('RGB shape: ', img.shape)
    print ("-------------------------\n")
    print ('ARGB shape:', alpha_img.shape)
    print ("-------------------------\n")
    print ('Gray shape:', gray_img.shape)
    print ("-------------------------\n")
    print ('img.dtype: ', img.dtype)
    print ("-------------------------\n")
    print( 'img.size: ', img.size)
    print ("-------------------------\n")

    #Extracting for RGB
    rgb=img[0][0]
    #print(rgb)
```

```
r=rgb[0]

g=rgb[1]

b=rgb[2]

#print(r)

#print(g)

#print(b)

def closest_colour(requested_colour):

    min_colours = {}

    for key, name in webcolors.css3_hex_to_names.items():

        r_c, g_c, b_c = webcolors.hex_to_rgb(key)

        rd = (r_c - requested_colour[0]) ** 2

        gd = (g_c - requested_colour[1]) ** 2

        bd = (b_c - requested_colour[2]) ** 2

        min_colours[(rd + gd + bd)] = name

    return min_colours[min(min_colours.keys())]


def get_colour_name(requested_colour):

    try:

        closest_name            =            actual_name            =
webcolors.rgb_to_name(requested_colour)

    except ValueError:

        closest_name = closest_colour(requested_colour)
```

```
            #actual_name = None
        return    closest_name

requested_colour = (r,g,b)
closest_name = get_colour_name(requested_colour)
print("\nClosest colour name of the image :", closest_name)

# Open our image
im = Image.open(imagePath)
# Convert our image to RGB
rgb_im = im.convert('RGB')

# Use the .size object to retrieve a tuple contain (width,height) of the image
# and assign them to width and height variables
width = rgb_im.size[0]
height = rgb_im.size[1]
wid=str(width)
hgt=str(height)
print ("\nWidth and height of the image")
print("=================================")
print("\nWidth = " + wid + " pixels")
print("Height = " + hgt + " pixels")
print("\t\t\t===================================\n")

print('--- Start recognize text from image ---')

print("Extracting Text .....\nThe text in the image is\n")

print("\t\t\t===================================\n")

# initialize the list of reference points and boolean indicating

# whether cropping is being performed or not

#payee

ref_point = []

cropping = False

xmin=57#12

ymin=56#30

xmax=363#257

ymax=83#356

def shape_selection(event, x, y, flags, param):

  # grab references to the global variables

  global ref_point, cropping, xmin, ymin, xmax, ymax


  ref_point = [(xmin, ymin)]

  cropping = True

  ref_point.append((xmax, ymax))

  cropping = False

  # draw a rectangle around the region of interest
```

```
    cv2.rectangle(image, ref_point[0], ref_point[1], (0, 255, 0), 2)
    cv2.imshow("image", image)
  clone = image.copy()
  cv2.namedWindow("image")
  cv2.setMouseCallback("image", shape_selection)

# keep looping until the 'q' key is pressed
  while True:
# display the image and wait for a key press
    cv2.imshow("image", image)
    key = cv2.waitKey(1) & 0xFF

# if the 'r' key is pressed, reset the cropping region
    if key == ord("r"):
      image = clone.copy()
      xmin=0
      ymin=0
      xmax=0
      ymax=0

# if the 'c' key is pressed, break from the loop
    elif key == ord("c"):
      break

  # if there are two reference points, then crop the region of interest
  # from the image and display it
  if len(ref_point) == 2:
    crop_img = clone[ref_point[0][1]:ref_point[1][1], ref_point[0][0]:ref_point[1][0]]
    #cv2.imshow("crop_img", crop_img)
    cv2.imwrite("payee.png", crop_img)
    cv2.waitKey(0)

  pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
  im=Image.open("G:/7 SEM MATERIALS/CAPSTONE/payee.png")
  payee=pytesseract.image_to_string(im,lang='eng')
  print('Payee : ' + payee)

  #Amount in words
  ref_point = []
  cropping = False
  xmin=95#12
  ymin=92#30
  xmax=478#257

  ymax=116#356

  def shape_selection1(event, x, y, flags, param):

    # grab references to the global variables

    global ref_point, cropping, xmin, ymin, xmax, ymax


    ref_point = [(xmin, ymin)]

    cropping = True

    ref_point.append((xmax, ymax))

    cropping = False

    # draw a rectangle around the region of interest

    cv2.rectangle(image, ref_point[0], ref_point[1], (0, 255, 0), 2)

    cv2.imshow("image", image)


  clone = image.copy()

  cv2.namedWindow("image")

  cv2.setMouseCallback("image", shape_selection1)


# keep looping until the 'q' key is pressed

  while True:

# display the image and wait for a key press

    cv2.imshow("image", image)
```

```
   key = cv2.waitKey(1) & 0xFF

 # if the 'r' key is pressed, reset the cropping region
   if key == ord("r"):
    image = clone.copy()
    xmin=0
    ymin=0
    xmax=0
    ymax=0

 # if the 'c' key is pressed, break from the loop
   elif key == ord("c"):
    break

# if there are two reference points, then crop the region of interest
# from the image and display it
  if len(ref_point) == 2:
   crop_img = clone[ref_point[0][1]:ref_point[1][1], ref_point[0][0]:ref_point[1][0]]
   #cv2.imshow("crop_img", crop_img)
   cv2.imwrite("amt_words.png", crop_img)
   cv2.waitKey(0)
```

```
    pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
    im=Image.open("G:/7 SEM MATERIALS/CAPSTONE/amt_words.png")
    amt_words=pytesseract.image_to_string(im,lang='eng')
    print('Amount : ' + amt_words)

#Amount in Rupees
    ref_point = []
    cropping = False
    xmin=526#12
    ymin=116#30
    xmax=639#257
    ymax=136#356
    def shape_selection2(event, x, y, flags, param):
  # grab references to the global variables
      global ref_point, cropping, xmin, ymin, xmax, ymax

      ref_point = [(xmin, ymin)]
      cropping = True
      ref_point.append((xmax, ymax))
      cropping = False
  # draw a rectangle around the region of interest
      cv2.rectangle(image, ref_point[0], ref_point[1], (0, 255, 0), 2)
      cv2.imshow("image", image)


    clone = image.copy()
    cv2.namedWindow("image")
    cv2.setMouseCallback("image", shape_selection2)


# keep looping until the 'q' key is pressed
    while True:
  # display the image and wait for a key press
      cv2.imshow("image", image)
      key = cv2.waitKey(1) & 0xFF


  # if the 'r' key is pressed, reset the cropping region
      if key == ord("r"):
        image = clone.copy()
        xmin=0
        ymin=0
        xmax=0
        ymax=0
```

```
 # if the 'c' key is pressed, break from the loop
   elif key == ord("c"):
    break

# if there are two reference points, then crop the region of interest
# from the image and display it
  if len(ref_point) == 2:
   crop_img = clone[ref_point[0][1]:ref_point[1][1], ref_point[0][0]:ref_point[1][0]]
   #cv2.imshow("crop_img", crop_img)
   cv2.imwrite("amt.png", crop_img)
   cv2.waitKey(0)

  pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
  im=Image.open("G:/7 SEM MATERIALS/CAPSTONE/amt.png")
  amt=pytesseract.image_to_string(im,lang='eng')
  print('Amount : ' + amt)

#Account Number
  ref_point = []
  cropping = False
  xmin=74#12
  ymin=155#30
  xmax=203#257
  ymax=172#356
  def shape_selection3(event, x, y, flags, param):
 # grab references to the global variables
    global ref_point, cropping, xmin, ymin, xmax, ymax

    ref_point = [(xmin, ymin)]
    cropping = True
    ref_point.append((xmax, ymax))
    cropping = False
 # draw a rectangle around the region of interest
    cv2.rectangle(image, ref_point[0], ref_point[1], (0, 255, 0), 2)
    cv2.imshow("image", image)

  clone = image.copy()
  cv2.namedWindow("image")
  cv2.setMouseCallback("image", shape_selection3)

# keep looping until the 'q' key is pressed
  while True:
```

```
  # display the image and wait for a key press
    cv2.imshow("image", image)
    key = cv2.waitKey(1) & 0xFF

  # if the 'r' key is pressed, reset the cropping region
    if key == ord("r"):
      image = clone.copy()
      xmin=0
      ymin=0
      xmax=0
      ymax=0

  # if the 'c' key is pressed, break from the loop
    elif key == ord("c"):
      break

# if there are two reference points, then crop the region of interest
# from the image and display it
  if len(ref_point) == 2:
    crop_img = clone[ref_point[0][1]:ref_point[1][1], ref_point[0][0]:ref_point[1][0]]
    #cv2.imshow("crop_img", crop_img)
```

```
      cv2.imwrite("num.png", crop_img)

      cv2.waitKey(0)


   pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"

   im=Image.open("G:/7 SEM MATERIALS/CAPSTONE/num.png")

   num=pytesseract.image_to_string(im,lang='eng')

   print('A/C No.: ' + num)


   print("------ Done -------")


   print("\n\nText from the image is extracted successfully !!!\n\n")


# Function to get the last RowCount in the Excel sheet , change the index of the sheet accordingly to get desired sheet.

   def getDataColumn():

      #define the variables

         rowCount=0

         columnNumber=0

         wb = open_workbook('output.xlsx')

         ws = wb.sheet_by_index(0)

         rowCount = ws.nrows
```

```
          rowCount+=1

          columnNumber=1

         #print("The number of data in Excel file is ",rowcount)

          print("\n\nThe number of data in Excel file is ",rowCount)

          writedata(rowCount,columnNumber)


#Data to specified cells.

   def writedata(rowNumber,columnNumber):

          book = openpyxl.load_workbook('output1.xlsx')

          sheet = book.get_sheet_by_name('Sheet1')


data=[(str(imagePath),str(imagePath),hgt,wid,closest_name,str(rgb),payee,num,
amt,amt_words)]

        #sheet.cell(row=rowNumber, column=columnNumber).value = 'kvs'

        # append all rows

          for row in data:

                 sheet.append(row)

          book.save('output1.xlsx')

          print('\n\nThe above outputs are saved successfully in Excel File...\n')


   getDataColumn()
```

```
    # display the image on screen with imshow()
    # after checking that it loaded
    if image is not None:
        cv2.imshow(imagePath, image)
    elif image is None:
        print ("Error loading: " + imagePath)
        #end this loop iteration and move on to next image
        continue

    # wait time in milliseconds
    # this is required to show the image
    # 0 = wait indefinitely
    # exit when escape key is pressed
    key = cv2.waitKey(0)
    if key == 27: # escape
        break

# close any open windows
cv2.destroyAllWindows()
```

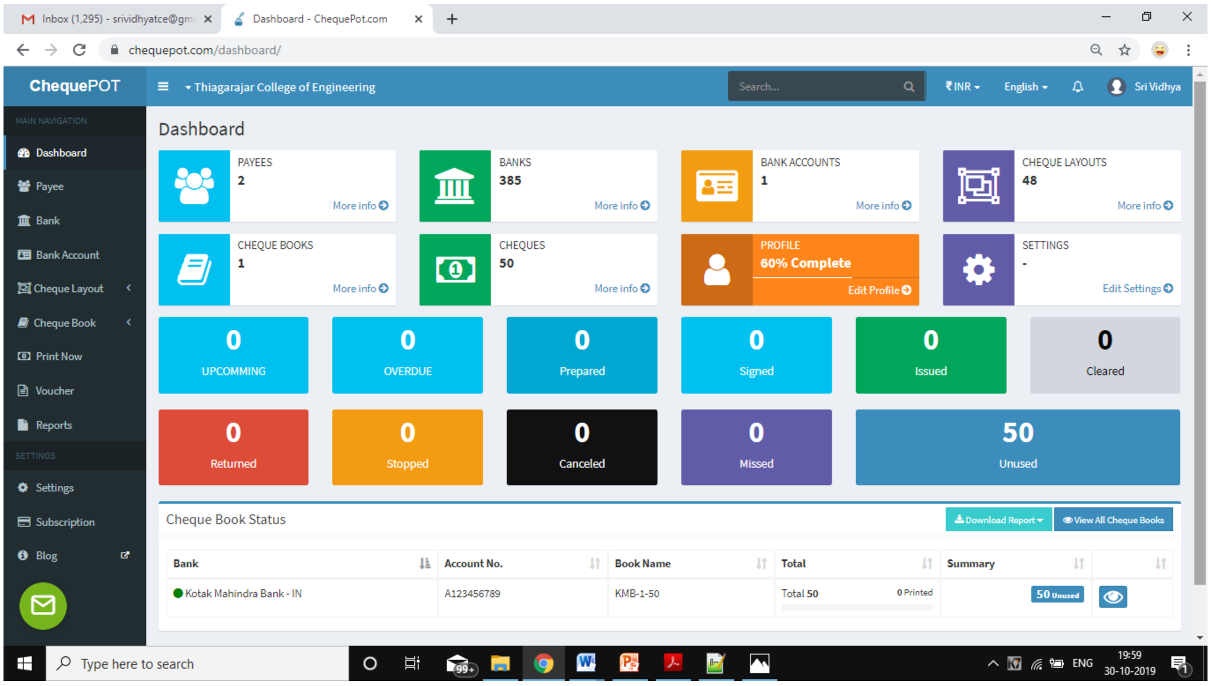
Table - 4.4.2 Functions used in implementation

Function name	Purpose
closest_colour	To find the rgb value of image
get_colour_name	To convert the rgb values to English colour name
get_string	To recognize the text from extracted image
getDataColumn	To read the excel file
Writedata	To append the extracted feature values to Excel file
shape_selection	Extracting region of interest

4.4.3 DATASET DESCRIPTION

- The dataset consists of 100 cheque images of Kotak Mahindra bank.
- All the cheque images are in the dimension 692 X 320.
- All the cheque images are kept in a folder and given as input to the application.
- The dataset has been prepared with the help of website - https://www.chequepot.com/
- The experiment has been done with the images prepared from the chequepot website.
- Cheques of various banks are available as template in this website. We have chosen the template of Kotak Mahindra Bank and entered different payee names and amount to prepare the dataset.

Images of chequepot website



chequepot.com/dashboard/

Dashboard

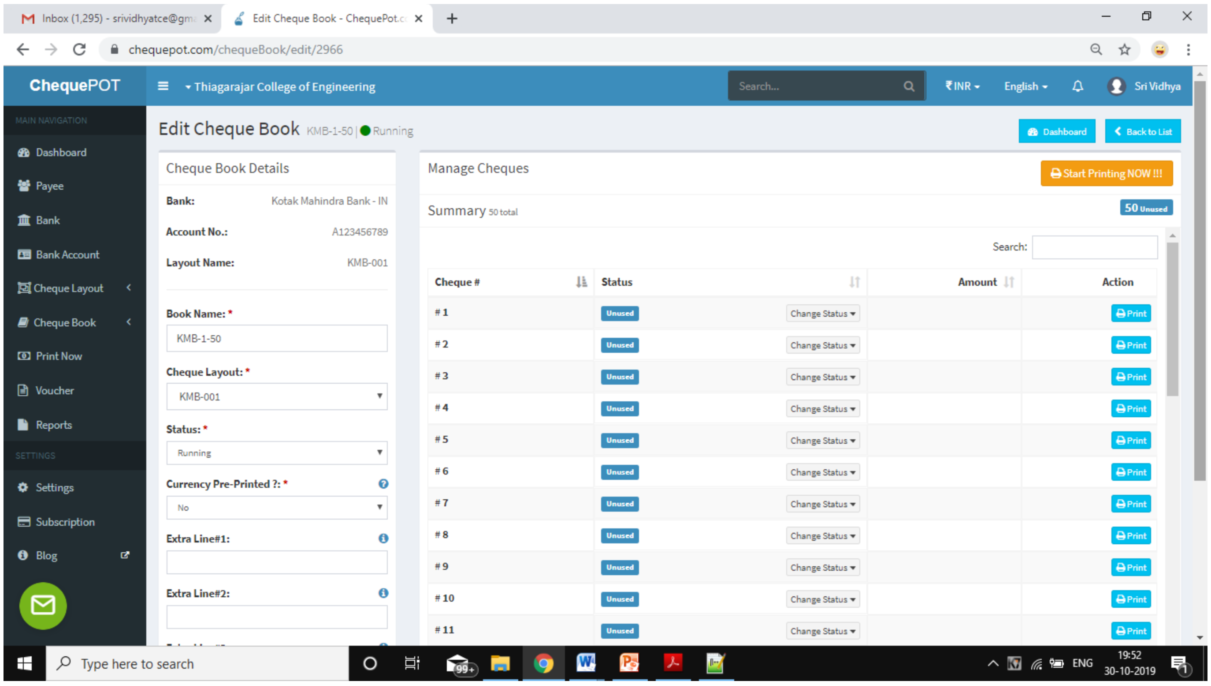
PAYEES 2 More info
BANKS 385 More info
BANK ACCOUNTS 1 More info
CHEQUE LAYOUTS 48 More info
CHEQUE BOOKS 1 More info
CHEQUES 50 More info
PROFILE 60% Complete Edit Profile
SETTINGS - Edit Settings

0	0	0	0	0	0
UPCOMMING	OVERDUE	Prepared	Signed	Issued	Cleared
0 Returned	0 Stopped	0 Canceled	0 Missed	50 Unused	

Cheque Book Status

Bank	Account No.	Book Name	Total	Summary
Kotak Mahindra Bank - IN	A123456789	KMB-1-50	Total 50 0 Printed	50 Unused

Figure – 4.4.1 Dashboard



chequepot.com/chequeBook/edit/2966

Edit Cheque Book KMB-1-50 Running

Cheque Book Details

Bank: Kotak Mahindra Bank - IN
Account No.: A123456789
Layout Name: KMB-001
Book Name: * KMB-1-50
Cheque Layout: * KMB-001
Status: * Running
Currency Pre-Printed ?: * No
Extra Line#1:
Extra Line#2:

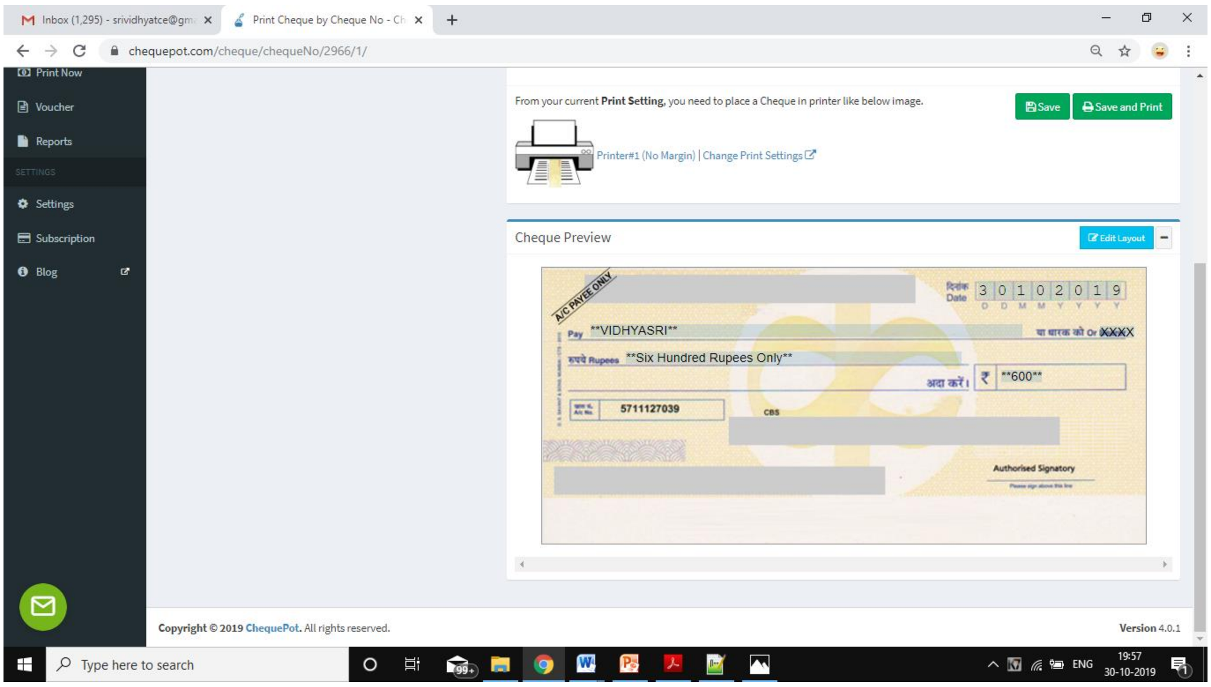
Manage Cheques

Summary 50 total — 50 Unused

Search:

Cheque #	Status		Amount	Action
# 1	Unused	Change Status		Print
# 2	Unused	Change Status		Print
# 3	Unused	Change Status		Print
# 4	Unused	Change Status		Print
# 5	Unused	Change Status		Print
# 6	Unused	Change Status		Print
# 7	Unused	Change Status		Print
# 8	Unused	Change Status		Print
# 9	Unused	Change Status		Print
# 10	Unused	Change Status		Print
# 11	Unused	Change Status		Print

Figure – 4.4.2 Cheque book layouts



From your current **Print Setting**, you need to place a Cheque in printer like below image.

Printer#1 (No Margin) | Change Print Settings

A/C PAYEE ONLY

दिनांक Date 3 0 1 0 2 0 1 9

Pay **VIDHYASRI** या धारक को Or XXXX

रुपये Rupees **Six Hundred Rupees Only**

अदा करें। ₹ **600**

खाता सं. A/c No. 5711127039

CBS

Authorised Signatory

Copyright © 2019 ChequePot. All rights reserved.

Version 4.0.1

Figure – 4.4.3 Cheque preview

Image of cheque dataset

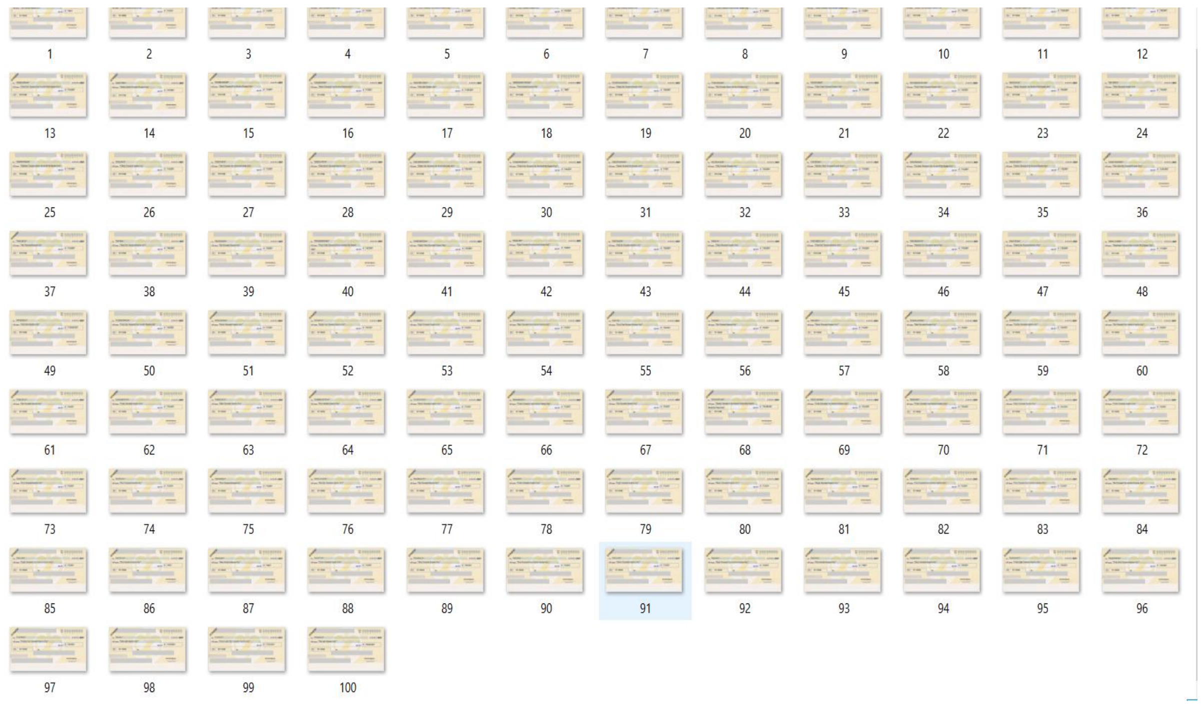


Figure – 4.4.4 Dataset

5. EXPERIMENTAL RESULTS

5.1 SCREESNHOTS

DIGITAL CHEQUES

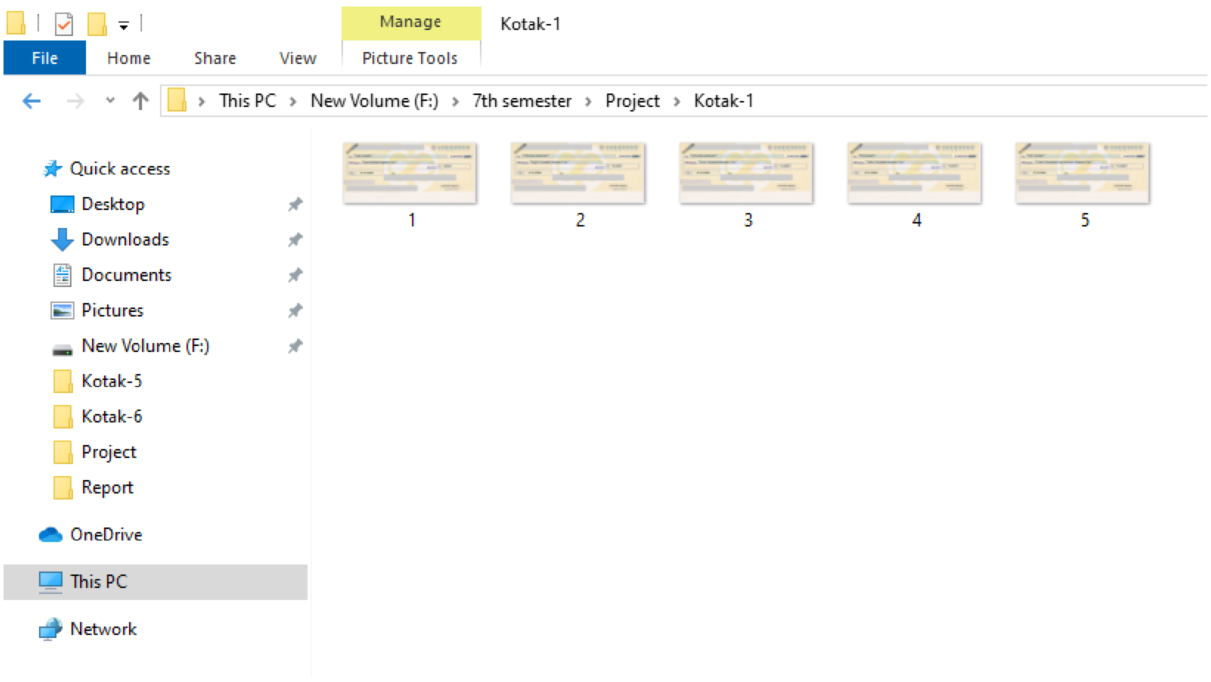


Figure – 5.1.1 Input folder

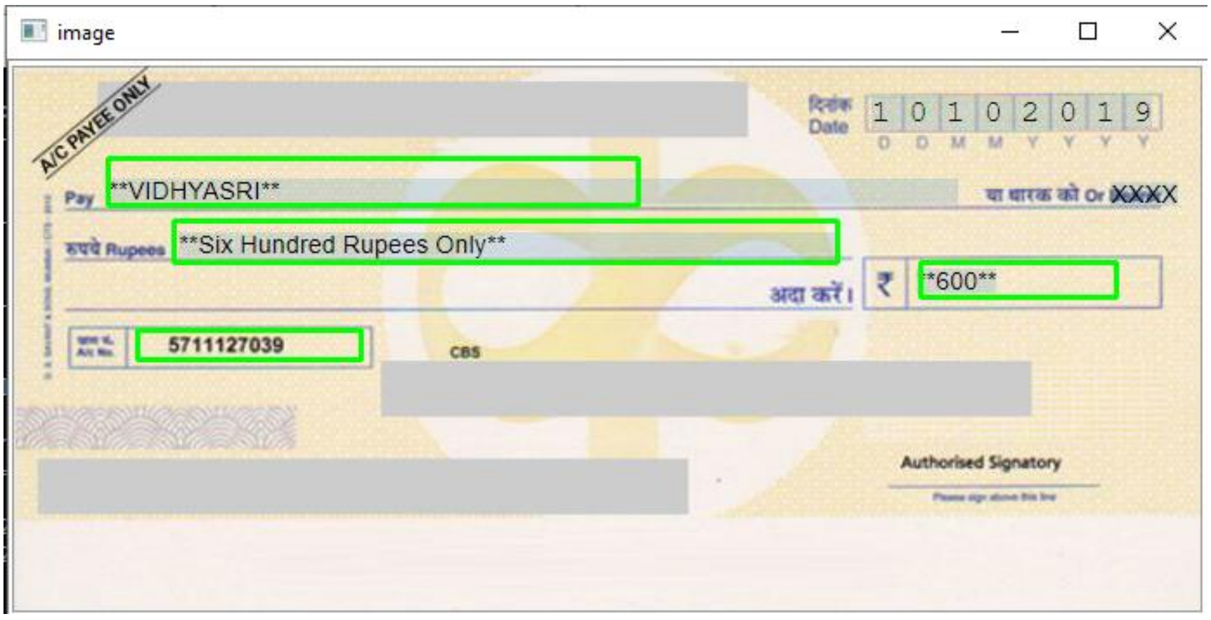
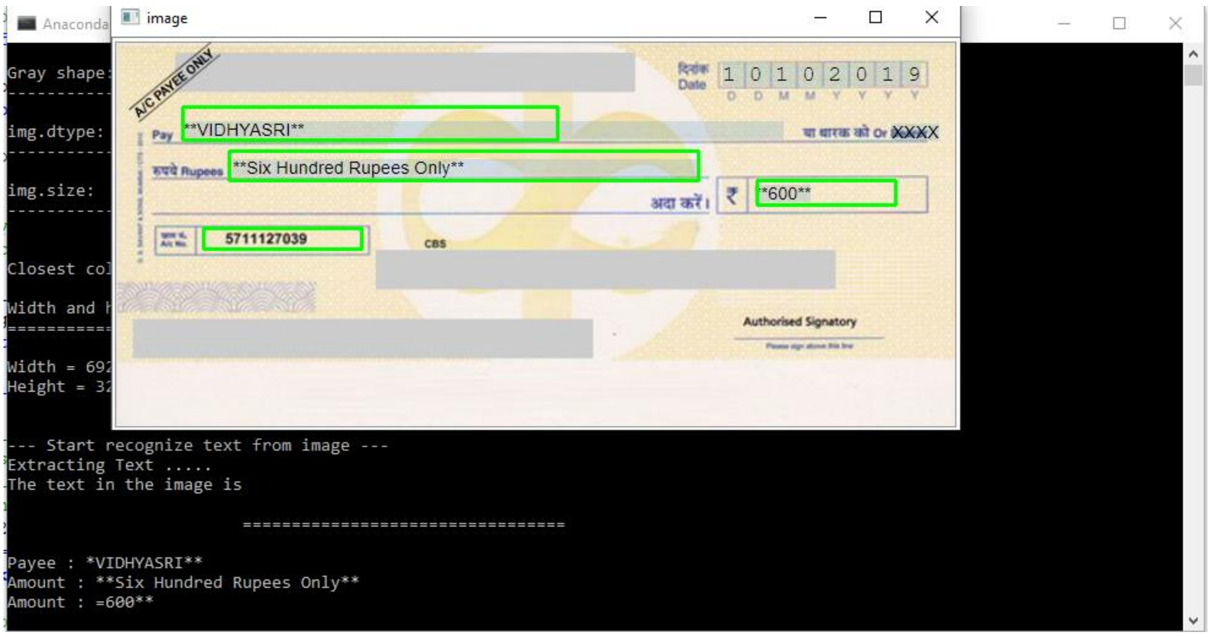


Figure – 5.1.2 Region of interest in cheque 1



```
Gray shape:
img.dtype:
img.size:
Closest col
Width and h
Width = 692
Height = 32

--- Start recognize text from image ---
Extracting Text .....
The text in the image is

                        ================================

Payee : *VIDHYASRI**
Amount : **Six Hundred Rupees Only**
Amount : =600**
```

Figure – 5.1.3 Text extraction

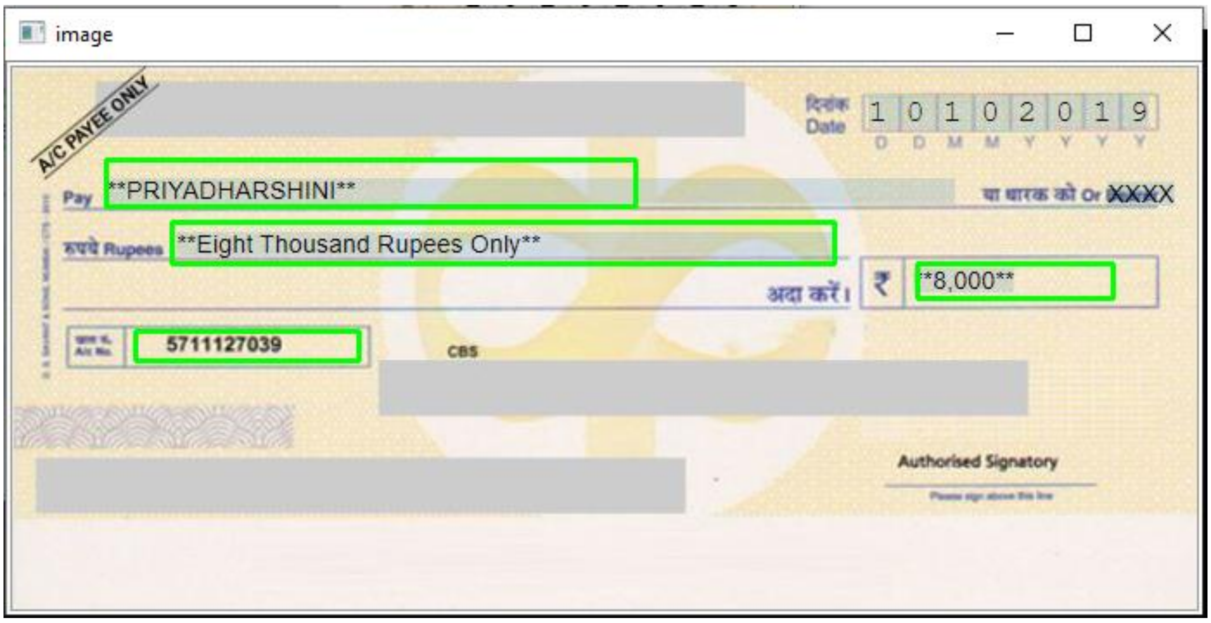
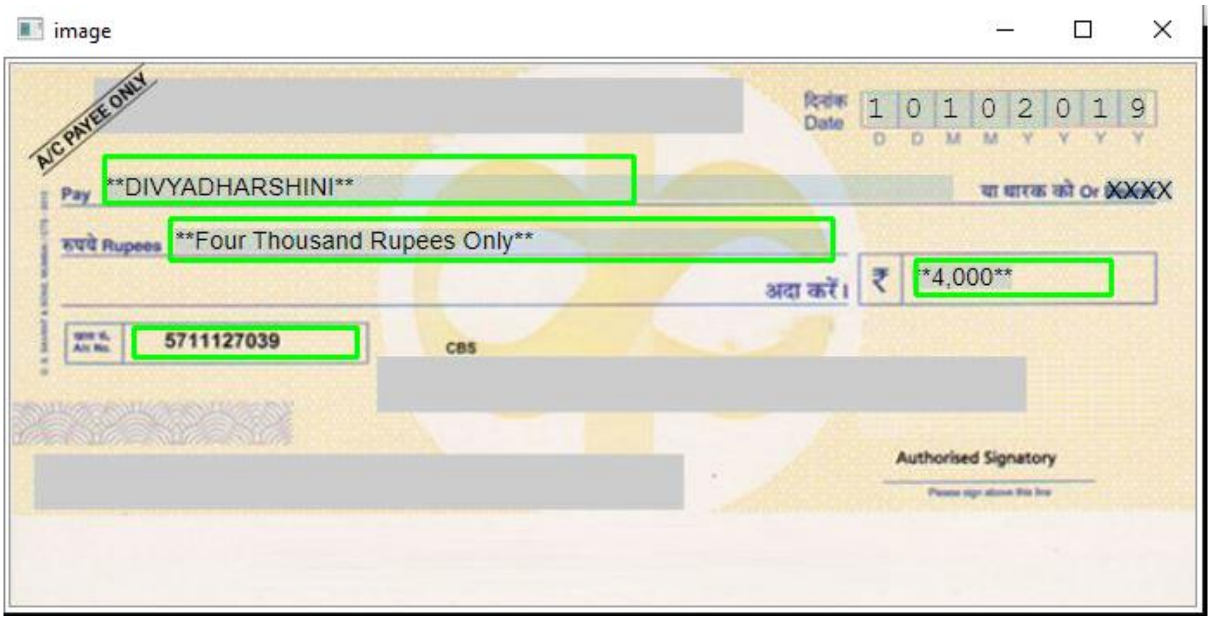


Figure – 5.1.4 Region of interest in cheque 2



image

A/C PAYEE ONLY

[redacted]

दिनांक Date 1 0 1 0 2 0 1 9
D D M M Y Y Y Y

Pay **DIVYADHARSHINI** या धारक को Or ~~XXXX~~

रुपये Rupees **Four Thousand Rupees Only**

अदा करें। ₹ *4,000**

खाता सं. A/c No. 5711127039

CBS

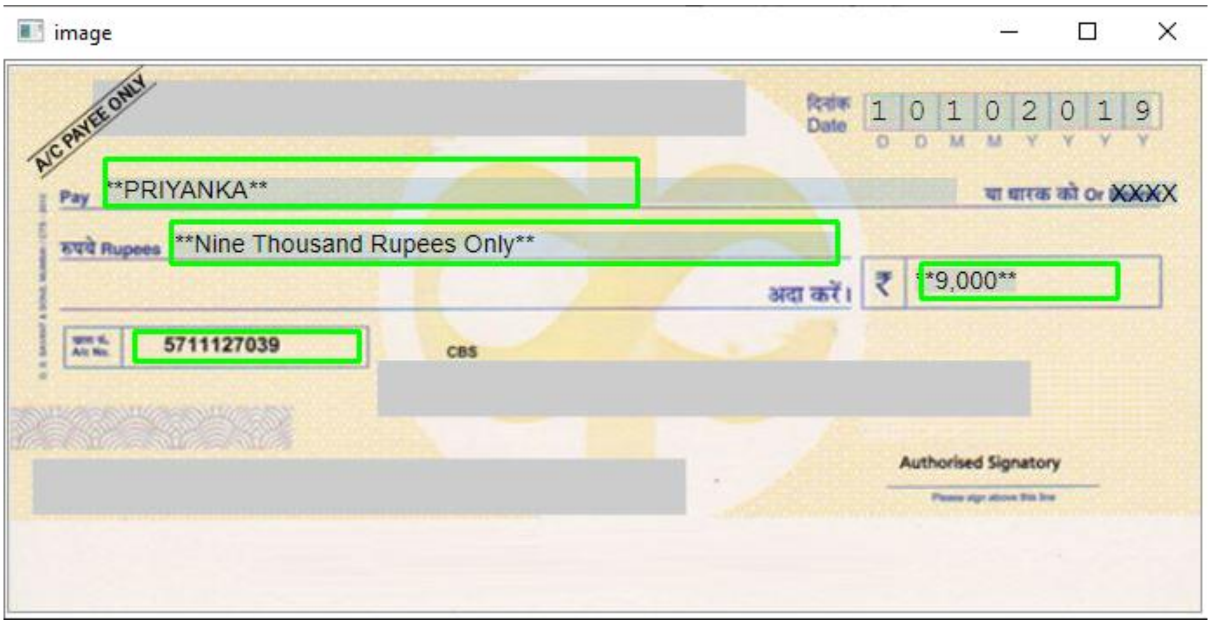
[redacted]

[redacted]

Authorised Signatory

Please sign above this line

Figure – 5.1.5 Region of interest in cheque 3



image

A/C PAYEE ONLY

[redacted]

दिनांक Date 1 0 1 0 2 0 1 9
D D M M Y Y Y Y

Pay **PRIYANKA** या धारक को Or ~~XXXX~~

रुपये Rupees **Nine Thousand Rupees Only**

अदा करें। ₹ *9,000**

खाता सं. A/c No. 5711127039

CBS

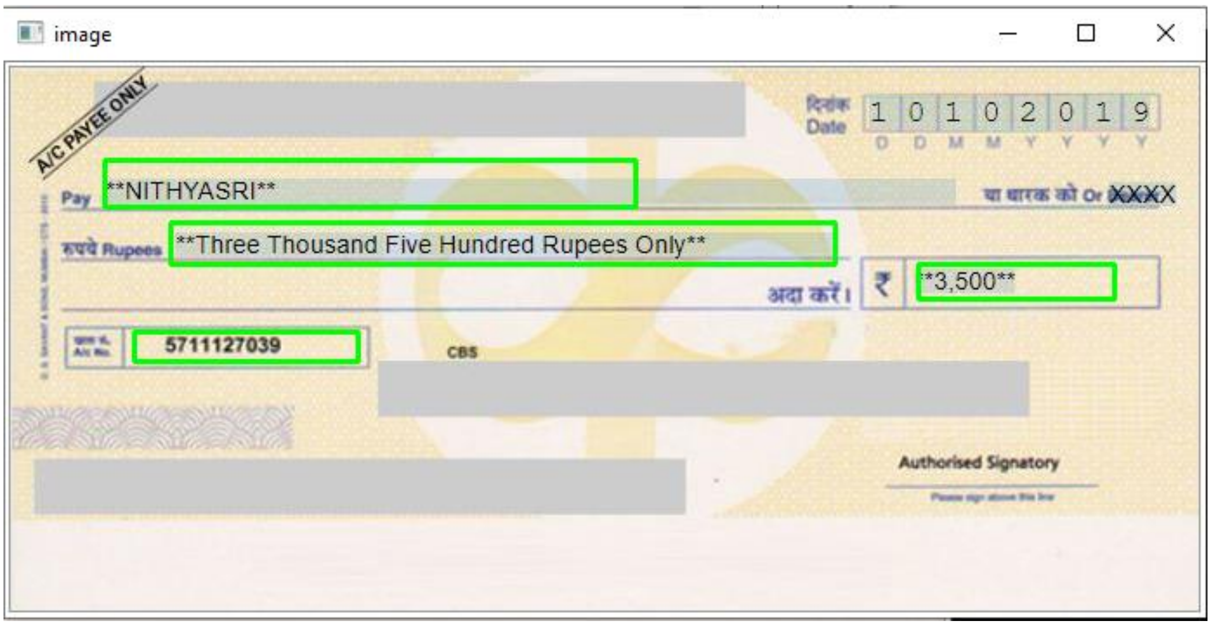
[redacted]

[redacted]

Authorised Signatory

Please sign above this line

Figure – 5.1.6 Region of interest in cheque 4



image

[redacted]

A/C PAYEE ONLY

दिनांक Date 1 0 1 0 2 0 1 9
D D M M Y Y Y Y

Pay **NITHYASRI** या धारक को Or XXXX

रुपये Rupees **Three Thousand Five Hundred Rupees Only**

अदा करें। ₹ *3,500**

खाता सं. A/c No. 5711127039

CBS

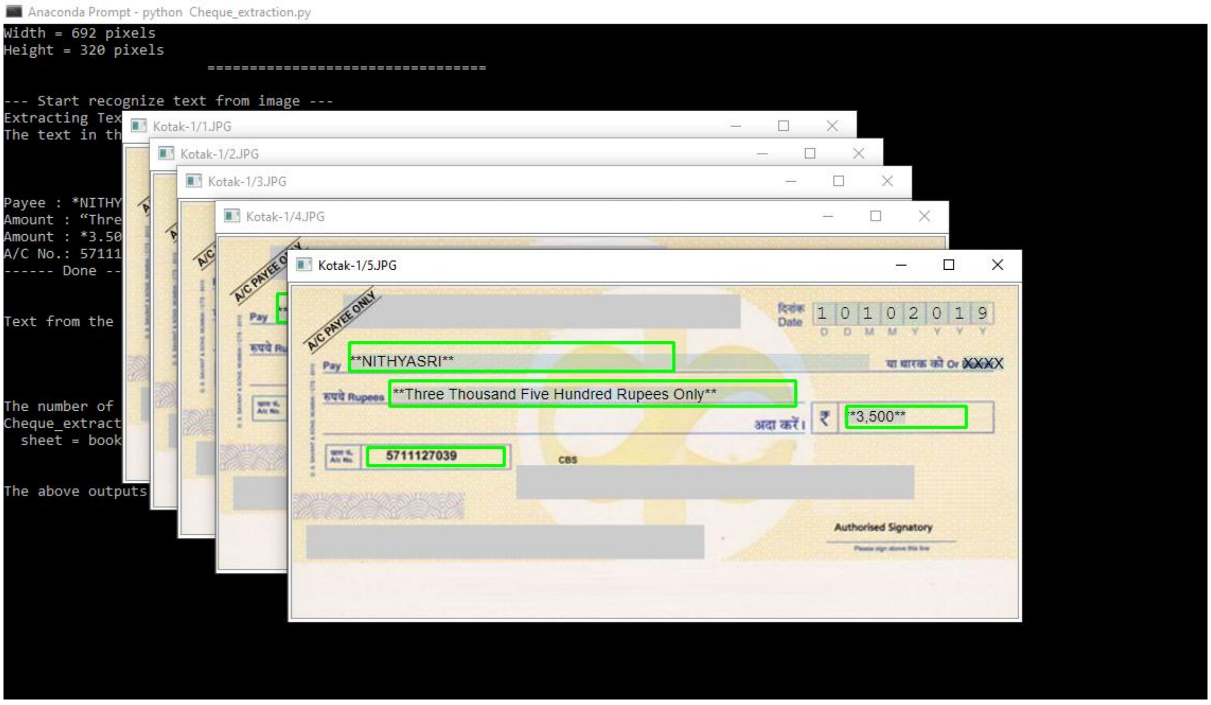
[redacted]

[redacted]

Authorised Signatory

Please sign above this line

Figure – 5.1.7 Region of interest in cheque 5


Anaconda Prompt - python Cheque_extraction.py

```
Width = 692 pixels
Height = 320 pixels
                        =================================

--- Start recognize text from image ---
Extracting Tex
The text in th


Payee : *NITHY
Amount : “Thre
Amount : *3.50
A/C No.: 57111
------ Done --


Text from the



The number of
Cheque_extract
  sheet = book


The above outputs
```

Kotak-1/1.JPG

Kotak-1/2.JPG

Kotak-1/3.JPG

Kotak-1/4.JPG

Kotak-1/5.JPG

[redacted]

A/C PAYEE ONLY

दिनांक Date 1 0 1 0 2 0 1 9
D D M M Y Y Y Y

Pay **NITHYASRI** या धारक को Or XXXX

रुपये Rupees **Three Thousand Five Hundred Rupees Only**

अदा करें। ₹ *3,500**

खाता सं. A/c No. 5711127039

CBS

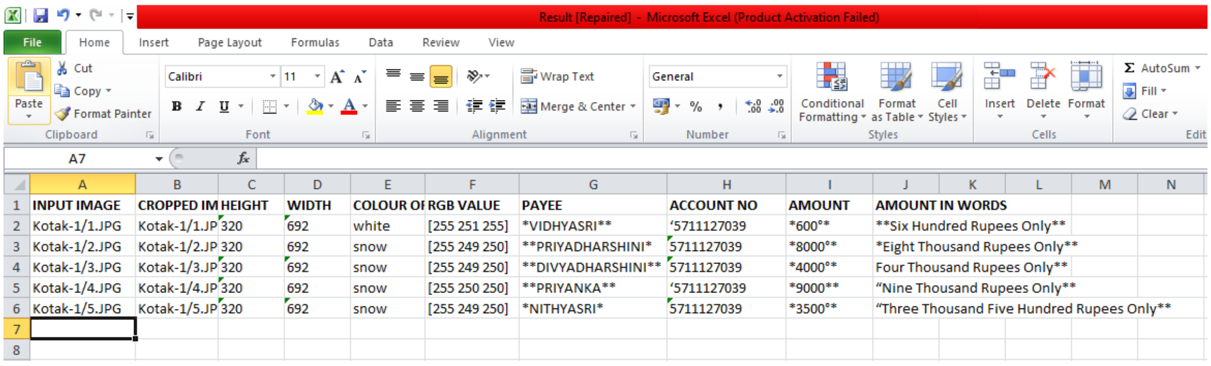
[redacted]

[redacted]

Authorised Signatory

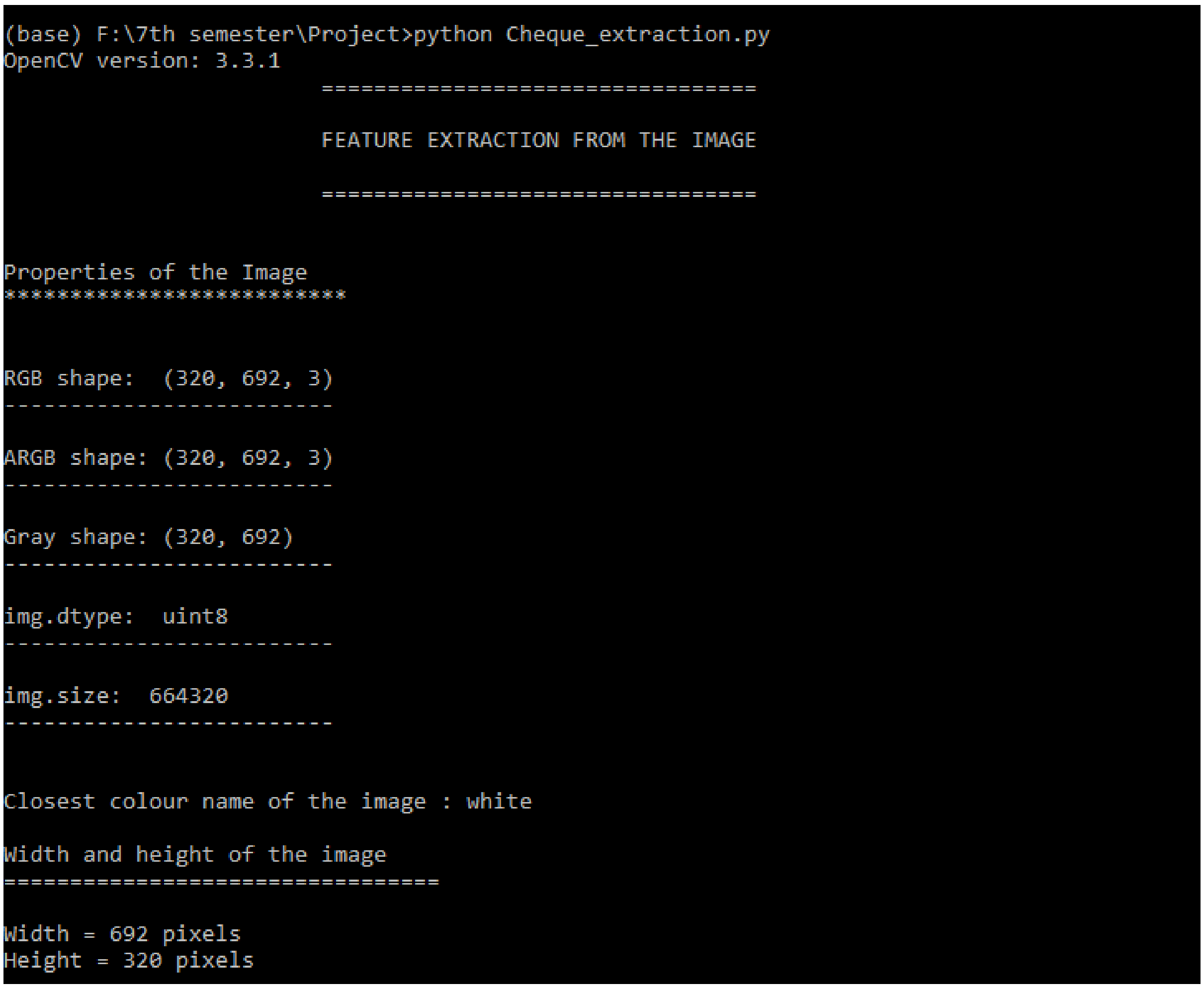
Please sign above this line

Figure – 5.1.8 Region of interest in all cheques



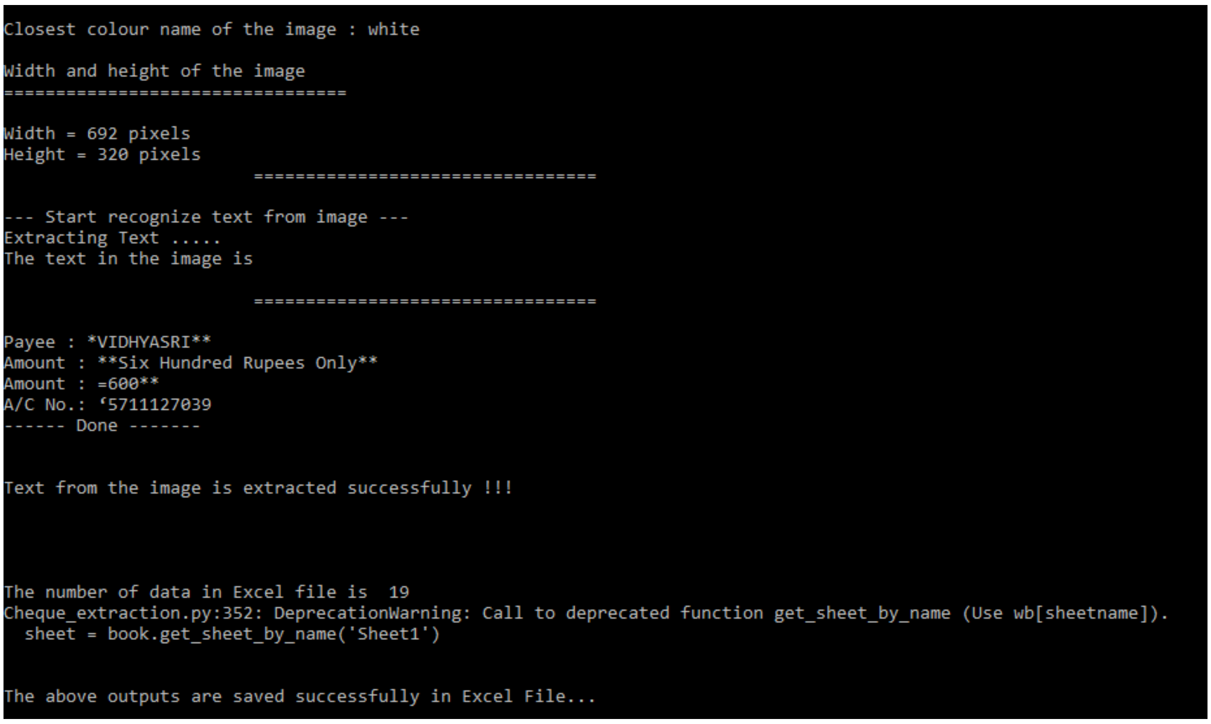
	A	B	C	D	E	F	G	H	I	J
1	INPUT IMAGE	CROPPED IM	HEIGHT	WIDTH	COLOUR OF	RGB VALUE	PAYEE	ACCOUNT NO	AMOUNT	AMOUNT IN WORDS
2	Kotak-1/1.JPG	Kotak-1/1.JP	320	692	white	[255 251 255]	*VIDHYASRI**	'5711127039	*600°*	**Six Hundred Rupees Only**
3	Kotak-1/2.JPG	Kotak-1/2.JP	320	692	snow	[255 249 250]	**PRIYADHARSHINI*	5711127039	*8000°*	*Eight Thousand Rupees Only**
4	Kotak-1/3.JPG	Kotak-1/3.JP	320	692	snow	[255 249 250]	**DIVYADHARSHINI**	5711127039	*4000°*	Four Thousand Rupees Only**
5	Kotak-1/4.JPG	Kotak-1/4.JP	320	692	snow	[255 250 250]	**PRIYANKA**	'5711127039	*9000**	"Nine Thousand Rupees Only**
6	Kotak-1/5.JPG	Kotak-1/5.JP	320	692	snow	[255 249 250]	*NITHYASRI*	5711127039	*3500°*	"Three Thousand Five Hundred Rupees Only**
7										
8										

Figure – 5.1.9 Text extracted to excel file



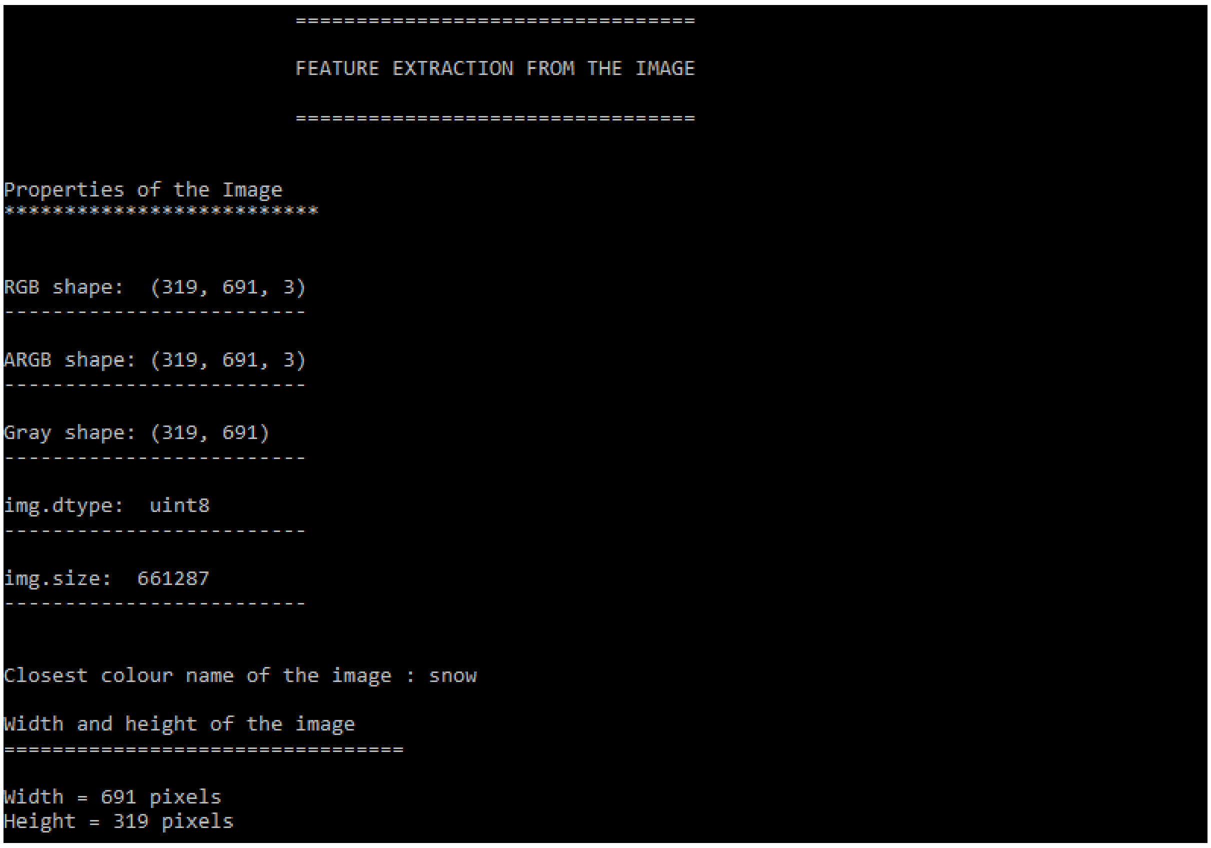
```
(base) F:\7th semester\Project>python Cheque_extraction.py
OpenCV version: 3.3.1
                        =================================

                        FEATURE EXTRACTION FROM THE IMAGE

                        =================================


Properties of the Image
**************************


RGB shape:  (320, 692, 3)
-------------------------

ARGB shape: (320, 692, 3)
-------------------------

Gray shape: (320, 692)
-------------------------

img.dtype:  uint8
-------------------------

img.size:  664320
-------------------------


Closest colour name of the image : white

Width and height of the image
=================================

Width = 692 pixels
Height = 320 pixels
```

Figure – 5.1.10 Output in anaconda prompt 1



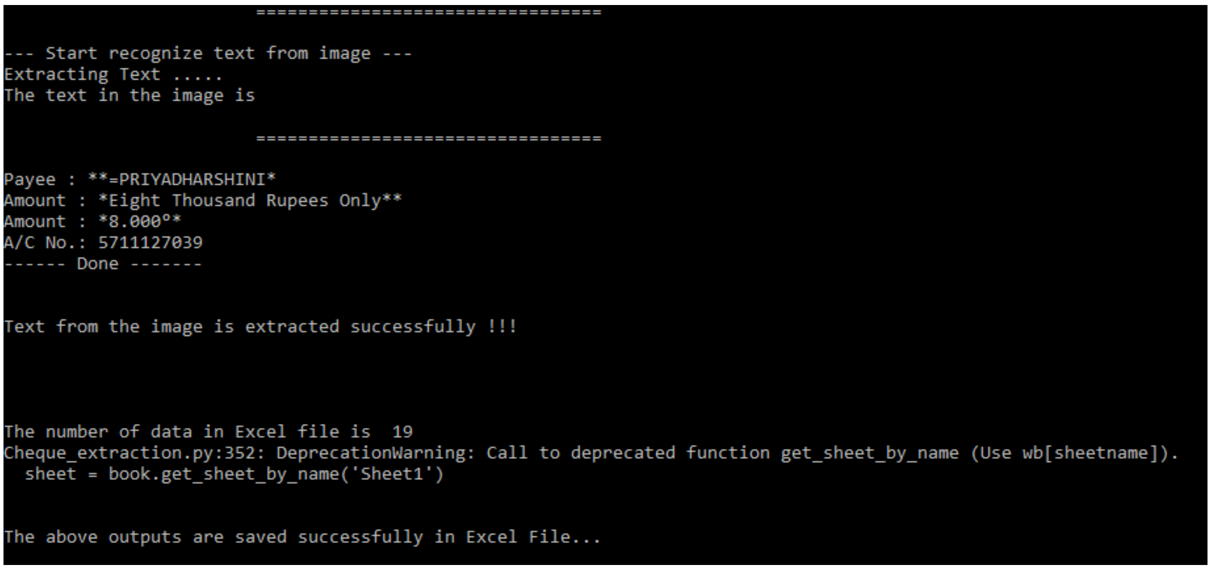
```
Closest colour name of the image : white

Width and height of the image
=================================

Width = 692 pixels
Height = 320 pixels
                        =================================

--- Start recognize text from image ---
Extracting Text .....
The text in the image is

                        =================================

Payee : *VIDHYASRI**
Amount : **Six Hundred Rupees Only**
Amount : =600**
A/C No.: ‘5711127039
------ Done -------


Text from the image is extracted successfully !!!



The number of data in Excel file is  19
Cheque_extraction.py:352: DeprecationWarning: Call to deprecated function get_sheet_by_name (Use wb[sheetname]).
  sheet = book.get_sheet_by_name('Sheet1')


The above outputs are saved successfully in Excel File...
```

Figure – 5.1.11 Output in anaconda prompt 2



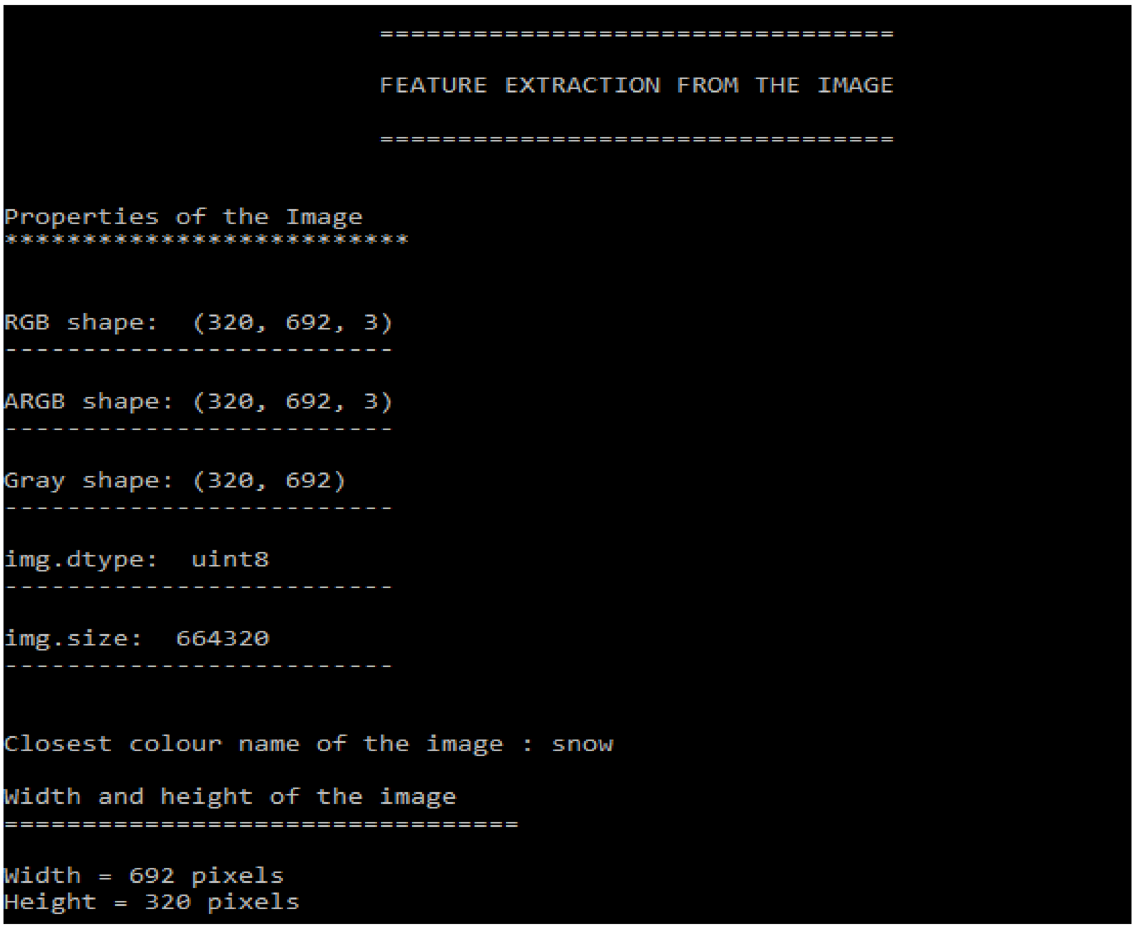
```
                            =================================

                            FEATURE EXTRACTION FROM THE IMAGE

                            =================================


Properties of the Image
**************************


RGB shape:  (319, 691, 3)
-------------------------

ARGB shape: (319, 691, 3)
-------------------------

Gray shape: (319, 691)
-------------------------

img.dtype:  uint8
-------------------------

img.size:  661287
-------------------------


Closest colour name of the image : snow

Width and height of the image
=================================

Width = 691 pixels
Height = 319 pixels
```

Figure – 5.1.12 Output in anaconda prompt 3



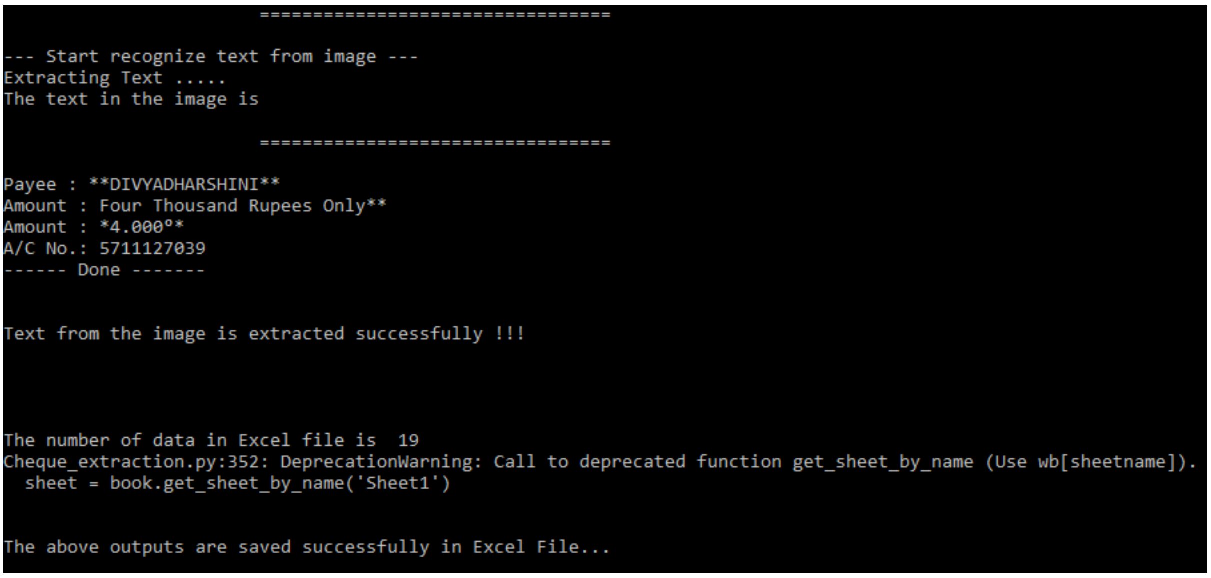
```
                        =================================

--- Start recognize text from image ---
Extracting Text .....
The text in the image is

                        =================================

Payee : **=PRIYADHARSHINI*
Amount : *Eight Thousand Rupees Only**
Amount : *8.000°*
A/C No.: 5711127039
------ Done -------


Text from the image is extracted successfully !!!



The number of data in Excel file is  19
Cheque_extraction.py:352: DeprecationWarning: Call to deprecated function get_sheet_by_name (Use wb[sheetname]).
  sheet = book.get_sheet_by_name('Sheet1')


The above outputs are saved successfully in Excel File...
```

Figure – 5.1.13 Output in anaconda prompt 4



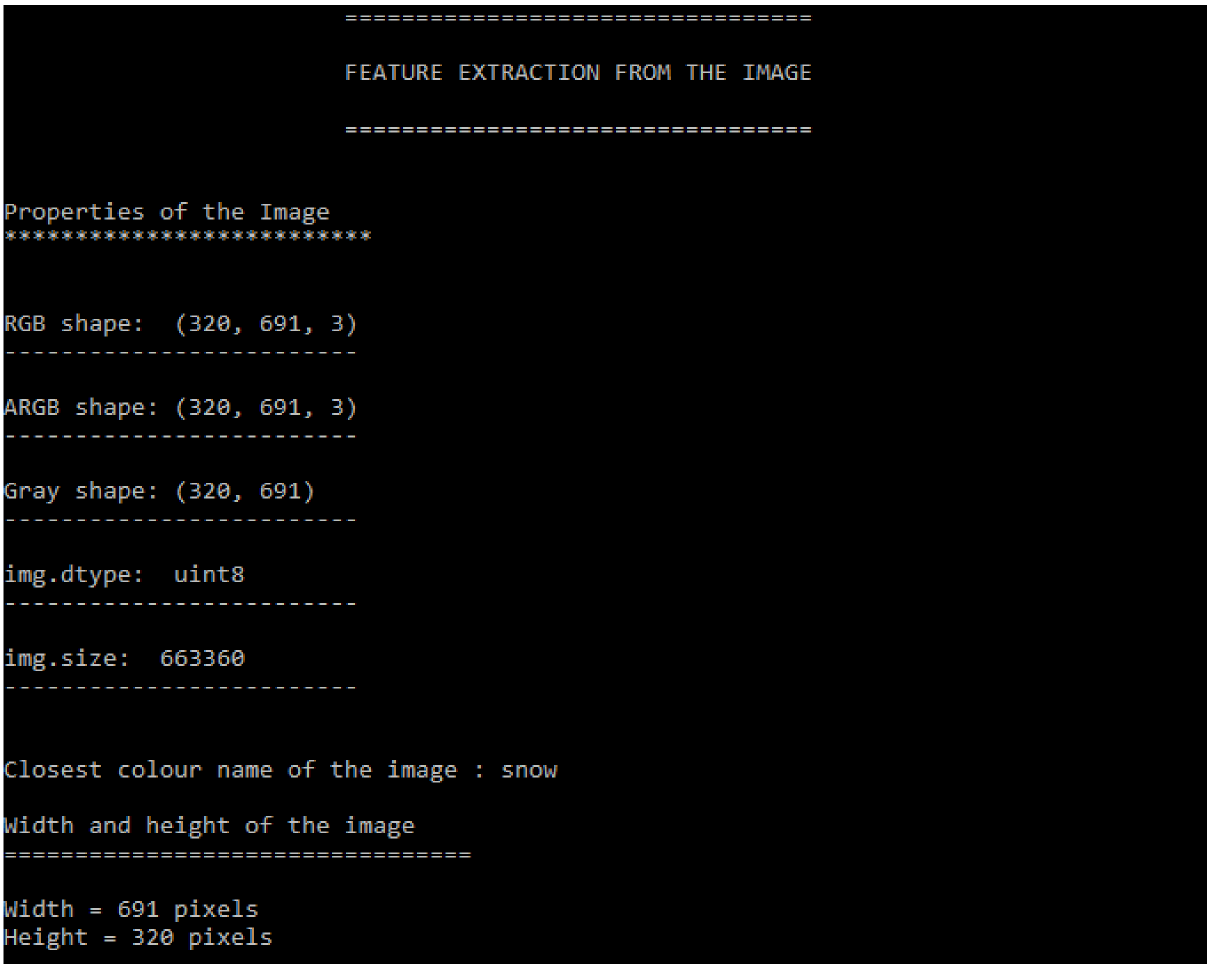
```
                        =================================

                        FEATURE EXTRACTION FROM THE IMAGE

                        =================================


Properties of the Image
**************************


RGB shape:  (320, 692, 3)
-------------------------

ARGB shape: (320, 692, 3)
-------------------------

Gray shape: (320, 692)
-------------------------

img.dtype:  uint8
-------------------------

img.size:  664320
-------------------------


Closest colour name of the image : snow

Width and height of the image
=================================

Width = 692 pixels
Height = 320 pixels
```

Figure – 5.1.14 Output in anaconda prompt 5



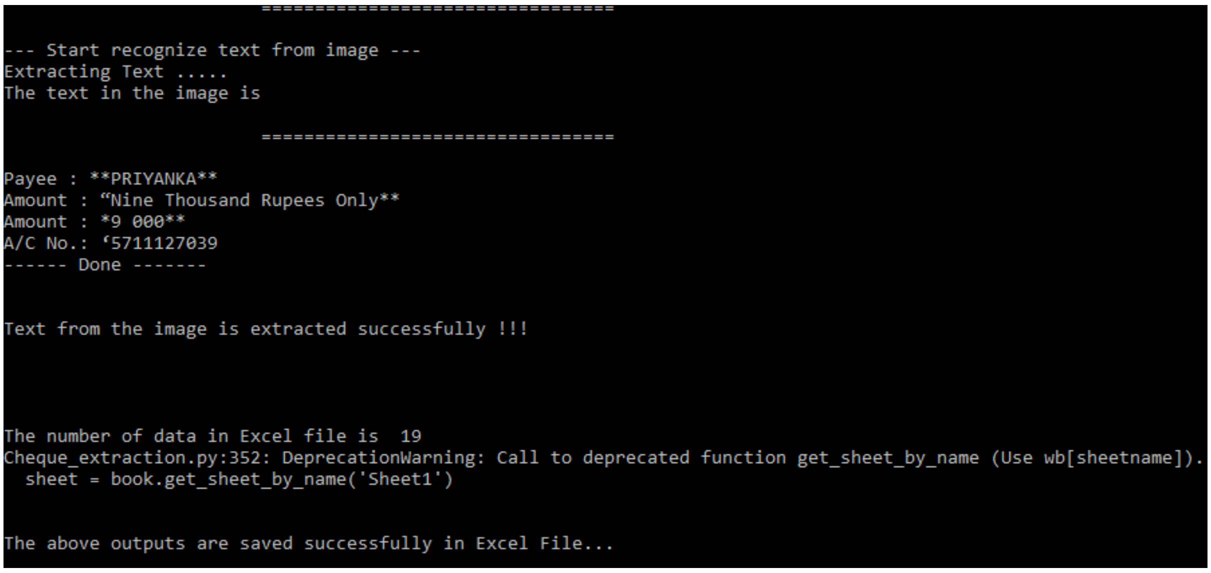
```
                        =================================

--- Start recognize text from image ---
Extracting Text .....
The text in the image is

                        =================================

Payee : **DIVYADHARSHINI**
Amount : Four Thousand Rupees Only**
Amount : *4.000°*
A/C No.: 5711127039
------ Done -------


Text from the image is extracted successfully !!!



The number of data in Excel file is  19
Cheque_extraction.py:352: DeprecationWarning: Call to deprecated function get_sheet_by_name (Use wb[sheetname]).
  sheet = book.get_sheet_by_name('Sheet1')


The above outputs are saved successfully in Excel File...
```

Figure – 5.1.15 Output in anaconda prompt 6



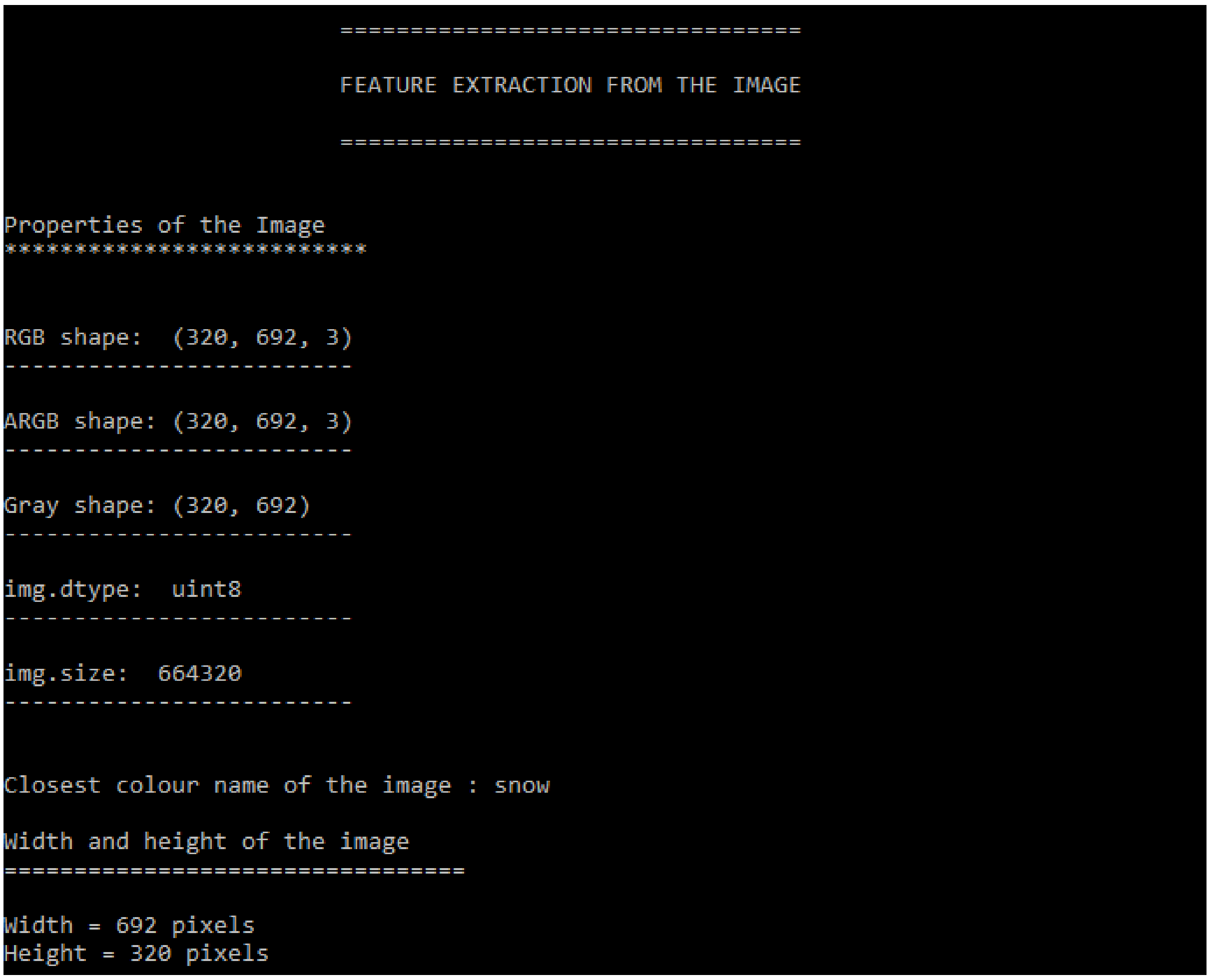
```
                                =================================

                                FEATURE EXTRACTION FROM THE IMAGE

                                =================================


Properties of the Image
**************************


RGB shape:  (320, 691, 3)
-------------------------

ARGB shape: (320, 691, 3)
-------------------------

Gray shape: (320, 691)
-------------------------

img.dtype:  uint8
-------------------------

img.size:  663360
-------------------------


Closest colour name of the image : snow

Width and height of the image
=================================

Width = 691 pixels
Height = 320 pixels
```

Figure – 5.1.16 Output in anaconda prompt 7



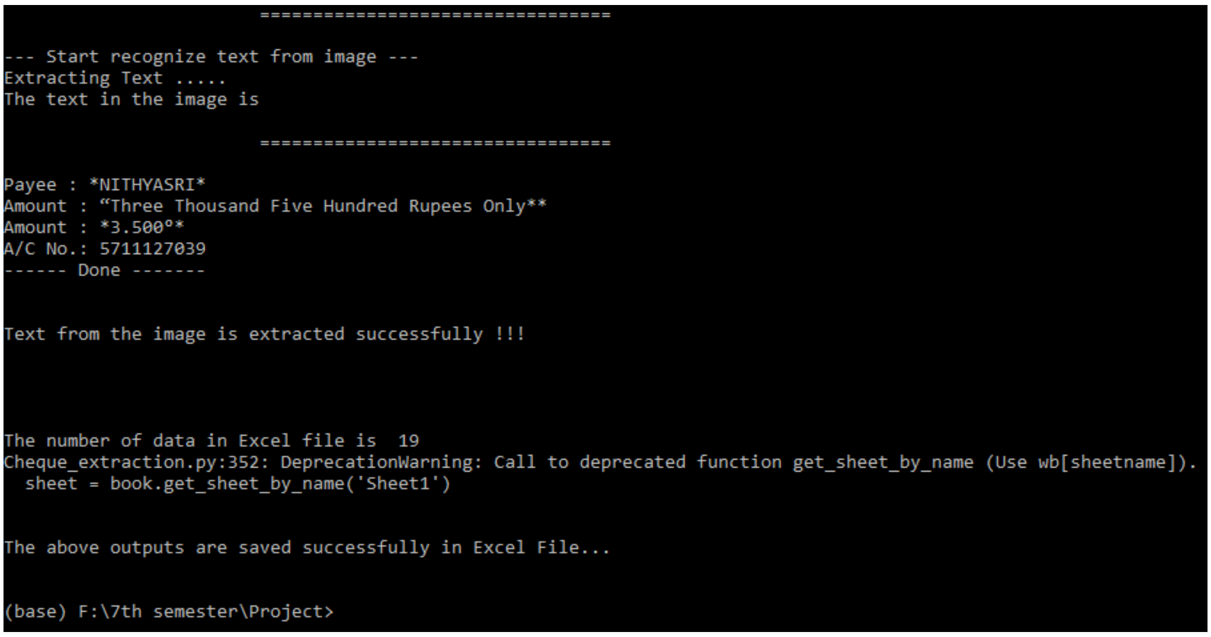
```
                        =================================

--- Start recognize text from image ---
Extracting Text .....
The text in the image is

                        =================================

Payee : **PRIYANKA**
Amount : “Nine Thousand Rupees Only**
Amount : *9 000**
A/C No.: ‘5711127039
------ Done -------


Text from the image is extracted successfully !!!



The number of data in Excel file is  19
Cheque_extraction.py:352: DeprecationWarning: Call to deprecated function get_sheet_by_name (Use wb[sheetname]).
  sheet = book.get_sheet_by_name('Sheet1')


The above outputs are saved successfully in Excel File...
```

Figure – 5.1.17 Output in anaconda prompt 8



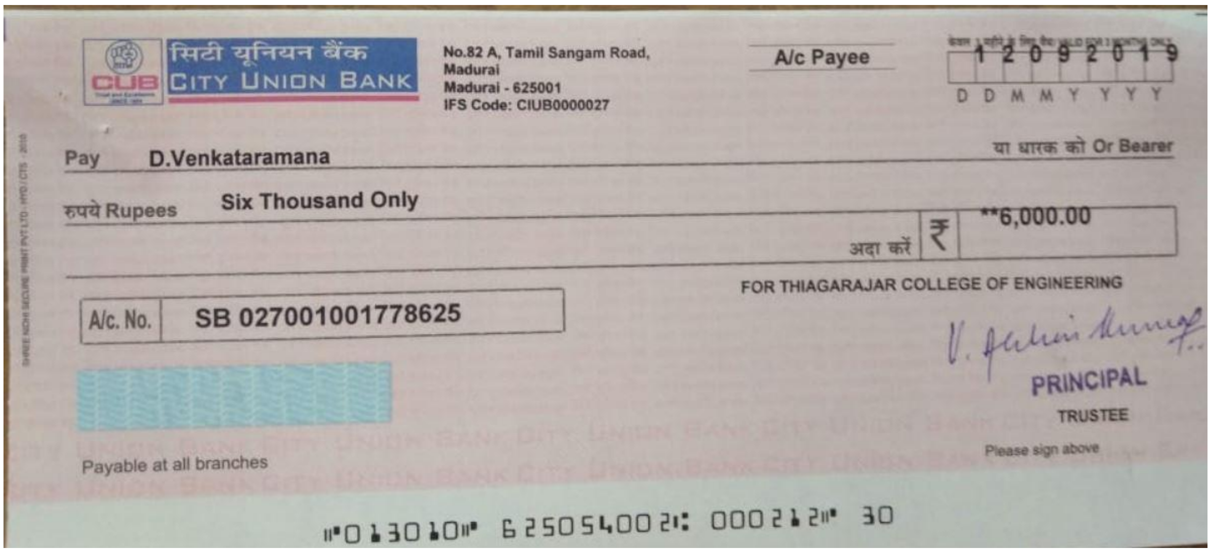
```
                        =================================

                        FEATURE EXTRACTION FROM THE IMAGE

                        =================================


Properties of the Image
**************************


RGB shape:  (320, 692, 3)
-------------------------

ARGB shape: (320, 692, 3)
-------------------------

Gray shape: (320, 692)
-------------------------

img.dtype:  uint8
-------------------------

img.size:  664320
-------------------------


Closest colour name of the image : snow

Width and height of the image
=================================

Width = 692 pixels
Height = 320 pixels
```

Figure – 5.1.18 Output in anaconda prompt 9



```
                        =================================

--- Start recognize text from image ---
Extracting Text .....
The text in the image is

                        =================================

Payee : *NITHYASRI*
Amount : “Three Thousand Five Hundred Rupees Only**
Amount : *3.500°*
A/C No.: 5711127039
------ Done -------


Text from the image is extracted successfully !!!



The number of data in Excel file is  19
Cheque_extraction.py:352: DeprecationWarning: Call to deprecated function get_sheet_by_name (Use wb[sheetname]).
  sheet = book.get_sheet_by_name('Sheet1')


The above outputs are saved successfully in Excel File...


(base) F:\7th semester\Project>
```

Figure – 5.1.19 Output in anaconda prompt 10



सिटी यूनियन बैंक
CITY UNION BANK

No.82 A, Tamil Sangam Road,
Madurai
Madurai - 625001
IFS Code: CIUB0000027

A/c Payee

1 2 0 9 2 0 1 9
D D M M Y Y Y Y

Pay D.Venkataramana या धारक को Or Bearer

रुपये Rupees Six Thousand Only

₹ **6,000.00

अदा करें

FOR THIAGARAJAR COLLEGE OF ENGINEERING

A/c. No. SB 027001001778625

[illegible]

PRINCIPAL

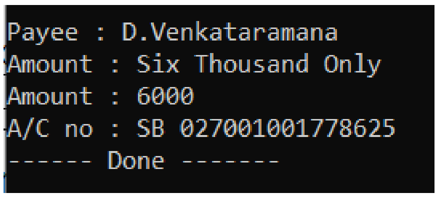
TRUSTEE

Please sign above

Payable at all branches

⑈013010⑈ 625054002⑆ 000212⑈ 30

Figure – 5.1.20 Demand Draft



```
Payee : D.Venkataramana
Amount : Six Thousand Only
Amount : 6000
A/C no : SB 027001001778625
------ Done -------
```

Figure – 5.1.21 Demand Draft output

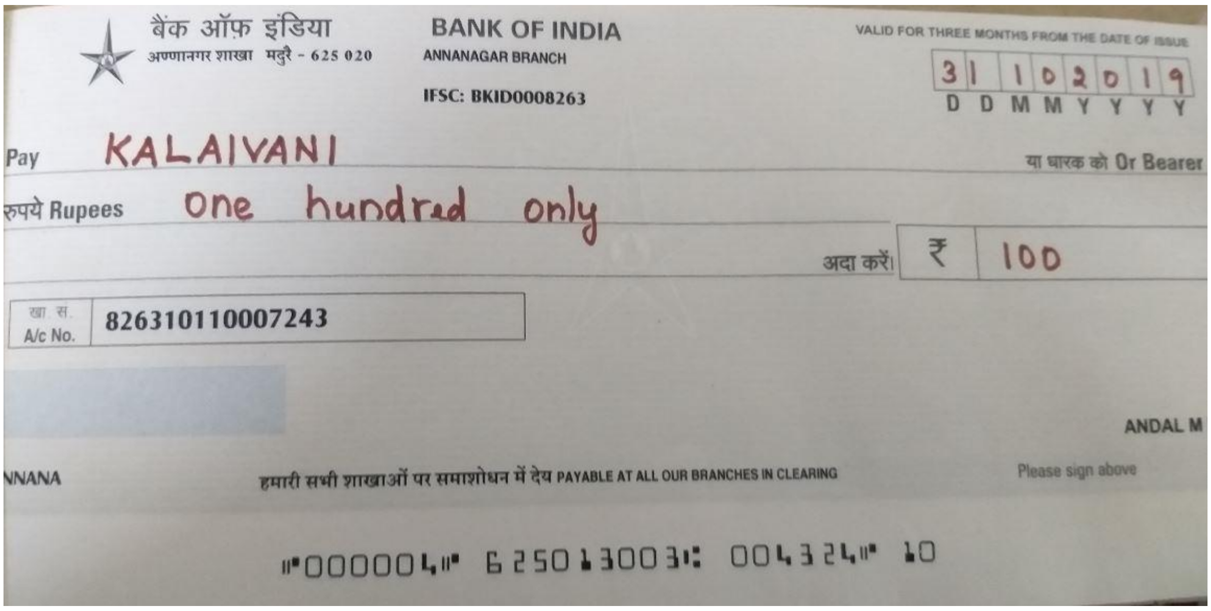

बैंक ऑफ़ इंडिया
अण्णानगर शाखा मदुरै - 625 020
BANK OF INDIA
ANNANAGAR BRANCH
IFSC: BKID0008263
VALID FOR THREE MONTHS FROM THE DATE OF ISSUE
31 10 2019
D D M M Y Y Y Y
Pay KALAIVANI
या धारक को Or Bearer
रुपये Rupees One hundred only
अदा करें ₹ 100
खा. सं. A/c No. 826310110007243
ANDAL M
NNANA
हमारी सभी शाखाओं पर समाशोधन में देय PAYABLE AT ALL OUR BRANCHES IN CLEARING
Please sign above
⑈000004⑈ 625013003⑆ 004324⑈ 10

Figure – 5.1.22 Handwritten Cheque

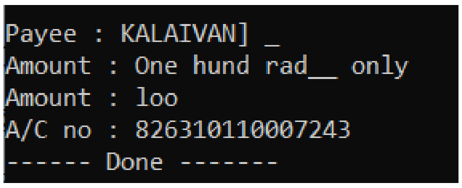

```
Payee : KALAIVAN] _
Amount : One hund rad__ only
Amount : loo
A/C no : 826310110007243
------ Done -------
```

Figure – 5.1.23 Handwritten Cheque output

5.2 PERFORMANCE ANALYSIS

Our application can recognize the text in digitally printed cheque images. Pytesseract has been used to extract the text from image. Pytesseract can recognize over 130 languages. Since we have used pytesseract – OCR engine, the performance measure of our application solely depends on the accuracy of OCR engine.

Accuracy of OCR Engine

After being open-sourced by HP in 2005, Tesseract is now developed and maintained by Google. The code has changed a lot since 2005, including conversion to Unicode and retraining. Experiments show that Tesseract is capable of achieving high accuracy such as 95% but in the case of some complex images having multi layered backgrounds or fancy text. Tesseract provides better accuracy in results if the pictures are in the gray scale mode as instead of color. The more time you spend on pre-processing gets more accuracy but it increases the runtime too. Different bank cheque images with different fields such as payee, amount in words, amount in rupees, account number and date are recognized successfully with success rate of about 93% to 95%. The proposed method is efficient in terms of speed and accuracy. A misdetection occurs if the input image is distorted, damaged, crossed, low illumination, low resolution and oblique view of cheque having one segment missing or crossed line or cancelled cheque with oblique line.

Our system can't be used to recognize hand written text. Since, the system only able to extract the details partially from the bank cheque. In order to extract hand written text, deep learning method should be adopted for more accuracy.

5. CONCLUSION AND FUTURE WORK

Automatic cheque processing is an interesting field of research from both scientific and commercial points of view. The variability of the size, structure and background of bank cheques, together with the complexity of the character recognition, makes the development of universal algorithms and strategies extremely challenging for automatic bank cheque processing. The misdetection occurs if the details of the cheque printed in out of index bound i.e. region of interest. The time required for the entire process of recognition by the proposed method also needs to be studied further, and it should be lesser than the time required for visual recognition, typing and verification. In future, the proposed system shall be converted to an entire web application or mobile application for use.

REFERENCES

- Xianlin Zhang , Xueming Li, Yang Liu, Fangxiang Feng (2019). A SURVEY ON FREE HAND SKETCH RECOGNITION AND RETRIEVAL. Retrieved from https://www.sciencedirect.com/science/article/pii/S0262885619300939

- Raghavendra SP, Ajit Danti , Suresha M (JAN, 2019). CORRELATION BASED TEMPLATE MATCHING FOR RECOGNITION OF BANK CHEQUE NUMBER. Retrieved from http://www.ijcea.com/correlation-based-template-matching-recognition-bank-cheque-number/

- R Jayadevan ,SR Kolhe, PM Patil ,U Pal (JUL, 2018). AUTOMATIC PROCESSING OF HANDWRITTEN BANK CHEQUE IMAGES . Retrieved from https://link.springer.com/article/10.1007/s10032-011-0170-8

- Karthick Kanagarathinam, Kavaskar Sekar (2019). TEXT DETECTION AND RECOGNITION IN RAW IMAGE DATASET OF SEVEN SEGMENT DIGITAL ENERGY METER DISPLAY. Retrieved from https://www.sciencedirect.com/science/article/pii/S0167739X18322180

- L. Minh Dang, Syed Ibrahim Hassan, Suhyeon Im, Irfan Mehmood, Hyeonjoon Moon (2018). TEXT RECOGNITION FOR THE DEFECTS EXTRACTION IN SEWERS CCTV INSPECTION VIDEOS. Retrieved fromhttps://www.sciencedirect.com/science/article/abs/pii/S0166361517304633

- Leena Mary Francis , N. Sreenath (2019). DETECTION OF TEXT IN THE NATURAL ENVIRONMENT. Retrieved from https://www.sciencedirect.com/science/article/pii/S0167739X18322180