Logo detection and Recognition from Scanned Images

Submitted in partial fulfillment of the requirements

Of the degree of

BACHELOR OF COMPUTER ENGINEERING

by

Vinay Billa (8)

Saikirankumar Dasari (15)

Saqib Naikwadi (55)

Aakash Tiwari (89)

Supervised by:

Prof. Sushama Khanvilkar

DEPARTMENT OF COMPUTER ENGINEERING

XAVIER INSTITUTE OF ENGINEERING

2018-2019

CERTIFICATE

This is to certify that the project entitled **“Logo detection and Recognition from Scanned Images”**is a bonafide work of “**Vinay Billa (8), Saikirankumar Dasari(15) ,Saqib Naikwadi (55), Aakash Tiwari(89)”**submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelors of Engineering”** in**“Computer Engineering”**

(Prof. Sushama Khanvilkar)	(Prof. Kunal Meher)
Supervisor/Guide	Co-Supervisor/Guide

(Prof. Sushama Khanvilkar)	(Dr.Y.D. Venkatesh)
Head of Department	Principal

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Vinay Billa	**(08)**------------------------------------
Saikirankumar Dasari	**(15)** ------------------------------------
Saqib Naikwadi	**(55)-**------------------------------------
Aakash Tiwari	**(89)-**------------------------------------

Date:

Place:

Project Report Approval

This project report entitled by **“Logo detection and Recognition from Scanned Images”** by “**Vinay Billa (8), Saikirankumar Dasari (15), Saqib Naikwadi (55), Aakash Tiwari(89)”** is approved for the degree of **Bachelors of Engineering** in **Computer Engineering.**

Examiners

1.---

2.---

Date:

Place:

ABSTRACT

We present a randomized and heuristic algorithm based on computer vision [5, 9] principles to detect and recognize logos in documents such as invoices and purchase orders. It is intended for use in enhancing document retrieval (DR) and named entity recognition (NER) tasks. Our algorithm operates in two phases: first, it detects the area in a page of a scanned document where a logo appears, and marks this area by using a bounding box; then, it recognizes the organizational entity associated with the logo by finding the best match for it against a pre-existing database of known logos. We extract distinctive image features from both logo and document using the Scale Invariant Feature Transform (SIFT [10]) algorithm and use the Fast Library for Approximate Nearest Neighbours (FLANN) for matching against the database, with a number of custom heuristics to reduce detection and recognition error. Our overall accuracy is 92.7% for detection and 91.7% for recognition. The technique is robust to image transformations such as rotation and scaling and to noise from blurriness and scanner artefacts.

Contents

List of Figures

List of Abbreviations

SIFT	Scale-Invariant Feature Transform
FLANN	Fast Library for Approximate Nearest Neighbors
DL	Deep Learning
OpenCV	Open Source Computer Vision
DR	Document Retrieval
NER	Named Entity Recognition
ER	Entity Relationship
NLP	Natural Language Processing
CRF	Conditional Random Fields
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
OCR	Optical Character Recognition
RANSAC	Random Sample Consensus

Chapter 1

Introduction

1.1 Problem Definition

Public organizations, institutes, companies and private sectors are generally interested in implementing digital mailrooms to improve the efficiency of paper-intensive workflows and to reduce the burden of manual processing of different administrative documents including incoming mails, faxes, forms, invoices, reports, employee records, health record, etc. By this digital mailroom, public and private sectors would obviously like to have an automatic indexation of their incoming documents that results in automatic classification, distribution and also easy access and retrieval of those documents in future. One possible solution is the use of textual information for automatic indexation of those administrative document. However, beside the presence of textual information, administrative documents commonly contain different salient entities such as logos, stamps, layout-structure,signatures, and bar-codes, which refer to the paradigm of corresponding organization,institute, product or personnel. These salient entities have a rich context information providing distinctive features and characteristics to deal with the problem of document image analysis. Truly speaking, these salient entities have mainly been considered as an alternative pathway for administrative document image retrieval (ADIR) and document classification.

Logo can be considered as an important and popular salient entities presented in administrative documents. The manual identification/verification of logos is not an easy task, as the documents in-flow in organizations is growing rapidly. Therefore, many research work has been carried out to automatically detect and verify logos facilitating such administrative document-based systems. Indeed, accurate detection and recognition of logo in document images provide us with a more reliable and appropriate system.

However, logo detection/recognition [8] is a challenging task, as logos are generally composed of quite complex symbols, graphical and textual components. Presence of noise and degradation makes the detection/recognition task more difficult. There are also some specific issues related to logos that make the detection/recognition more challenging.

1.2 Aims and Objectives

1.2.1 AIMS:

The main aim of this project is to present the efficient and robust framework for detection as well as recognition of logo images.

1.2.2 OBJECTIVES:

1. To present the literature review over different approaches presented over logo.
2. To present the analysis of different methods according to their detection accuracy and performances. We use SIFT [10] (Scale Invariant Feature Transform) algorithm for feature extraction and FLANN (Fast Library for Approximate Nearest Neighbors [11]) for matching against the database.
3. To present and discuss the proposed methods for logo detection and recognition.

4. To present the practical analysis of proposed work and its evaluation against the existing methods.

1.3 Scope of the Project

Financial transactions for exchange of goods or services between parties usually generate data in the form of physical documents such as invoices and purchase orders. Such documents are used to record, verify, and scrutinize transactions they may either have a rigidly structured layout like a form , or be more unstructured in nature. Use cases such as domestic and international trade finance transactions typically involve additional third parties like banks and financial institutions as intermediaries who facilitate the exchange and act as trusted parties. While such parties earn significant revenue from such trade activities, they need to avoid invalid transactions, which may have legal sanctions and lead to significant losses .Also in Administrative, communication and filing procedures, which were mostly paper-based, are driven into a digital environment by the ubiquity of different computation facilities. In all these applications, the objective is not only to preserve documents in a digital format, but also to process documents to provide an easy access and retrieval service to a wider number of users. Traditionally, document retrieval (DR) is associated to textual analysis .This retrieval process can be accelerated using technique of logo detection and the organizational entity can be identified quickly.

Features :

- User will be able to get the details of respective oraganization’s details which includes name , occupation , address , phone number etc.
- Scanned document can be in rotated (slant, inverted) form or can have incorrect borders.
- Since the document scanned is matched using FLANN [11] the processing and matching process is faster.
- Logo matching process is automatic, i.e. user just have to upload the scanned images.

Hence there are lot of research and work is done to make the project a successful one. Algorithm are used to bring out the best efficiency. So our scope is to improve the document identification and retrieval easy and less time consuming.

1.4 Existing System

Till now work on logo detection and recognition [8] was concerned with providing some automatic support to the logo registration process. The system check whether other registered logos in archives of millions, exist that have similar appearance to the new coming logo image, in order to ensure that it is sufficiently distinctive and avoid confusion . Kato's system was among the earliest ones. It converts a normalized logo image to a 64 pixel grid, and calculated a global feature vector from the frequency distributions of edge pixels. Recently, Wei et al proposed a different solution, where logos were described by global Zernike moments, local curvature and distance to centroid. Other methods have used different global descriptors of the full logo image either accounting for logo contours or exploiting shape descriptors such as shape context. This method assume that a logo picture is fully visible in the image, is not corrupted by noise and is not subjected to transformations. According to this, they cannot be applied to real world images. Hichem Sahbi, Lamberto Ballan, in their paper they presented the validity of method through extensive experiments on the challenging MICC-Logos dataset.This method overtakes, by 20%, baseline as well as the state of the art matching /recognition procedures. Sami M. Halawani1 and Ibrahim A. Albidewi in their research work concerned with the specific class of complicated objects, i.e. logo. The progress, particularly in this field, is still at extensive research work level, due to infinite varieties of shapes and classes which are used. Essentially, the algorithm proposed is based on Principle Component Analysis (PCA) approach. In this technique, the PCA is used to extract the features, kept inherent in the normalized pattern for later matching process. The experiment had shown that, the minimum number of weights needed to perform a correct recognition is seventeen. However, for the purpose of image reconstruction, this number is not enough to build a visible image.

Suma R1, Anita George introduced a method in their work is a novel logo detection and localization approach based on a new class of similarities referred to as context dependent The strength of the proposed method resides in several aspects: (i) the inclusion of the information about the spatial configuration in similarity design as well as visual features, (ii) the ability to control th influence of the context and the regularization of the solution via our energy function, (iii) the tolerance to different aspects including partial occlusion, makes it suitable to detect both near duplicate logos as well as logos with some variability in their appearance. J.Matas et al , introduced a Novel rotation invariant detector. It was coined as SURF [6]. The robustness ensures that invariants from multiple measurement regions (regions obtained by invariant constructions from external regions), some that are significantly larger than the MSERs. Guangyu Zhu:Doermann D. in their work aimed at Graphics detection and recognition are of fundamental research problems in document image analysis and retrieval. In this work, they developed an automatic logo-based document image retrieval system that handles: Logo detection and segmentation by boosting a cascade of classifiers across multiple image. David S. Doermann, Ehud Rivlin and Isaac Weiss In their work, they present a multi-level staged approach to logo recognition which uses global invariants to prune the database and local a fine invariants to obtain a more refined match. They obtain an invariant signature which can be used for matching under a variety of transformations.Their work provide a method of computing Euclidean invariants, and show how to extend them to capture similarity, a fine and projective invariants when necessary. They implement feature detection [8], feature extraction and local invariant algorithms and successfully demonstrate the approach on a small database

Chapter 2

Literature Survey

2.1 General

The literature on Logo Detection and Recognition [8] is extremely extensive. Financial transactions for exchange of goods or services between parties usually generate data in the form of physical documents such as invoices and purchase orders. Such documents are used to record, verify, and scrutinize transactions; they may either have a rigidly structured layout like a form, or be more unstructured in nature. Use cases such as domestic and international trade finance transactions typically involve additional third parties like banks and financial institutions as intermediaries who facilitate the exchange and act as trusted parties. While such parties earn significant revenue from such trade activities, they need to avoid invalid transactions, which may have legal sanctions and lead to significant losses.

To validate transactions, such intermediaries currently employ and invest in manual verification, which is painfully slow and inefficient. It is, therefore, of great interest to them to be able to leverage automated DR and NER

Techniques from such unstructured documents for use by authorized personnels or officials to validate and act on transactions. The work described here is a component of a larger platform-based solution to address this gap: to help visualize information, to improve decision making , and further minimize the processing time and increase productivity for this task.

Logos are an important element of documents, and detecting and recognizing them significantly augments the richness of the information being presented to the human operator. However, logos are not text, and are therefore not captured by traditional Optical Character Recognition (OCR) techniques. Specialized image-based techniques are therefore needed.

The approach to the logo detection [8] and recognition problem is driven by the following considerations :

- Recent developments in Deep Learning (DL) have been applied for computer vision [9] tasks such as brand logo detection. Neural network based architectures such as CNNs provide good results in practice. However, the success of any DL technique depends critically on the volume and quality of the training data and the annotation scheme [5]. For unstructured form analytics, factors such as varying scale and orientation of images, scan and/or OCR quality, and scanner artifacts present additional challenges for labeled data generation. For cases where collecting large volumes of training data is a challenge, algorithmic solutions tend to perform better and more robustly. The lack of large volumes of training data was a real-life constraint for us, leading us to approach the problem algorithmically.
- Various public cloud-based web services (such as Google Cloud Vision [2] and ClarifAI [1]) support logo detection by exposing an API over REST-style frameworks. This saves considerable investment on model building activities (gathering training data, building neural network architectures, and fitting themto available data) at the cost of requiring documents to be uploaded to the public cloud. This is not considered an acceptable tradeoff for sensitive documents relating to financial transactions, where a data breach

or information leak can result in significant losses or liability actions. This consideration ruled out for us the possibility of using pre-built models and transfer learning techniques, and led us to develop an on-premise robust algorithmic solution.

2.2 Related works on logo detection and recognition.

The problem of using logo information for document image processing basically involves two main tasks: (i) finding boundary of a logo/ on a document image irrespective of its class, and (ii) indexing/matching the detected logo/ candidate region to a database for classifying or for concluding that the region is not of interest. The former is referred to as logo/ detection/spotting, while the latter is called logo/ recognition.

Logo retrieval can be viewed as a combination of two problems that one wants to simultaneously detect and recognize a logo across a dataset based on a query image. Considering these two tasks, the literature review related to logo and may also be divided into (i) logo detection, and (ii) logo recognition. In relation to the evolutionary aspect of the technology, the recognition step has been started earlier than the detection step. However, to respect the continuity of the presentation in this paper, first, the techniques developed for logo/ detection are discussed and then methods for logo/ recognition are overviewed.

A number of techniques have been proposed in the past for logo detection [8]. These techniques can be categorized into four main groups:

(i) connected component based approaches,

(ii) window-sliding (block) based approaches,

(iii) detection based on recognition approaches,

(iv) techniques based on local descriptors.

2.3 Review on Logo detection results and discussion.

To have an overall overview of the techniques proposed for the detection of logos [8] in document images a brief description of each technique is provided in Table 3. As demonstrated in Table 3, most of the methodologies in the literature for the logo detection have been conducted on the Tobacco-800 dataset. Among the CC-based logo detection systems the method presented in has provided considerably encouraging results, however, the CC based methods for logo detection are generally sensitive to noise, degradation, scale variation [10] and skew. In the second category of logo detection methods [8], the method presented has shown quite interesting results. Nevertheless, concerning subjective idea of logo detection, the method presented is zone classification rather than logo detection. Regarding the uniformity of detection and recognition in a single framework with the use of a feedback strategy, the method presented in is the best example in the literature for logo detection/recognition. In case where the use of a priori and domain knowledge is affordable, the systems proposed in can be good choices to deal with the problem of logo detection/recognition. In terms of color/gray 10 document images the method based on bag-of-visual-word and key point detection and matching can be well adapted to the problem. Moreover, most of the methods are invariant to image transformations. The techniques based on SIFT and SURF [6] are also segmentation free techniques.

In relation to time complexity of the logo detection methods in the literature, it is noted that the time complexities of most methods are of linear complexity (*O(n)* or *O(kn)*, where *k* is a small constant value) that makes those methods suitable for practical applications.

Table 2.1. The results reported based on different techniques for logo detection in the literature. Here T, R and S denote Translation, Rotation and Scale properties respectively. Y and N are also used as the means of “Yes” and “No”.

Method	Features	Invariant to image transformation			Dataset	No. of images	No. Logos	Training dataset	Testing datase t	Precision (%)	Acc. (%)	Time complexity
		T	R	S								
[36]	SURF features	Y	Y	Y	Tobacco -800	435	<435	NA	NA	87	NA	*$O(n^2)$*
[37]	Local descriptors, convex hull, direction	Y	Y	Y	Tobacco -800	1290	415	50	365	99.4	86.5	*$O(n)$*
[38]	Foreground spatial density	Y	Y	Y	Tobacco -800	1290	416	100	316	32.1	39.3	*$O(n)$*
[39]	Contour based features	Y	N	Y	Tobacco -800	1290	432	50	1240	44	91	*$O(kn)$*
[39]	Contour based features	Y	N	Y	Tobacco -800	426	426	50	376	92.98	90.05	*$O(kn)$*
[42]	CC based features	Y	N	Y	UMD	255	255	35	220	NA	88	*$O(n)$*
[43]	Moments, Contour features	Y	N	Y	Tobacco -800	400	400	100	300	94.7	84.2	*$O(n)$*
[44]	Feature rectangles	Y	N	N	Tobacco -800	416	435	100	316	93.3	80.4	*$O(kn)$*
[45]	Context distance	Y	Y	Y	Tobacco -800	1290	432	50	1240	73.5	84.2	*$O(kn)$*
[46]	Shape descriptors	Y	Y	Y	Tobacco -800	1290	432	432	386	82.6	78.5	*$O(2^{\log n})$*
[70]	Probability, Gaussian features	N	Y	Y	Tobacco -800	1290	432	100	316	75.25	91.50	*$O(kn)$*
[76]	Local descriptors	Y	Y	Y	Tobacco -800	1290	432	432	1240	91.15	88.78	*$O(2n+logn)$*
[77]	Local descriptors	Y	Y	Y	Tobacco -800	1290	432	432	1240	97.67	95.86	*$O(2n+logn)$*

2.4 Recognition methodologies

Logo identification can be regarded as an application of the general pattern recognition schemes. Alike any other pattern recognition systems, the problem of logo and seal [4] recognition generally involves three vital tasks:

(i) Preprocessing,

(ii) Extracting meaningful features for a logo,

(iii) Classifying/indexing a detected/spotted logo into a large database of logos or a database of documents which contain logos.

A critical assumption in most of the recognition strategies presented in the literature is the availability of logos or seals provided by manual segmentation. Pieces of works related to logo and seal recognition are reviewed below.

2.5 Related methods for logo recognition

In document image analysis applications, two analogous logo recognition tasks are of interest. First, given a document, which contains a logo, classify the logo as one of a finite set of known logos in a logo database or conclude that the logo does not present in the database. Second, given an extracted logo (known or unknown), index a database of documents and extract all the documents, which contain the extracted logo. Both problems can be viewed as indexing into a possibly large database of logos or documents based on features extracted from logos or candidate logo regions. In both cases a primary logo detection procedure is necessary to provide the required information for logo recognition. In literature, there are many research works for the recognition of logos and trademarks. Since trademark recognition is relatively close to the logo recognition problem, a few number of papers from trademark recognition literature are also reviewed here. From the literature of logo recognition techniques, it is evident that most of the techniques considered logos as entities perfectly segmented from the document. In this case, two main steps of feature extraction and then classification are commonly employed for logo recognition in the literature. For the sake of clarity, the same pipeline used in the recognition process is followed in this paper to review the logo recognition methods in the literature. The logo recognition results obtained in the literature are also reported.

The results obtained from the pieces of work in the literature of logo and trademark recognition are provided in Table 5. From Table 5, it may be noted that most of the results have been reported on the University of Maryland Logo Dataset (UMLD). Both nearest neighbor[11] and learning based classification approaches have frequently been used in the literature for the recognition of logos. However, the nearest neighbor based approaches are computationally expensive in the cases of high dimension features and a large number of instances. Hashing and indexing can be integrated in such a scenario to take care of high dimensionality and scalability. Learning based methods for the recognition are not also efficient when the number of classes is greater than 1000. In such a case, similarity based approaches seem to be more appropriate solution for the recognition purpose.

Table 2.2. Comparative analysis of logo recognition results.

Method	Feature	Invariant to image transformation	Classifier	Dataset	Train	Test	Accuracy
[3]	Higher-order spectra	Y	Nearest Neighbor	UMLD (105 logo classes)	3150=105*30 105=105*1	3150=105*30 3150=105*30	99.6% >90%
[4]	Gradient	N	Neural Network	UMLD (40 logo classes)	4000=40*100	4000=40*100	99.15%
[6, 7]	Gradient	N	Neural Network	134 images, UMLD (105 logos)+29	26800=200*134	26800=200*134	99.04
[8]	text, primitive shapes (line, circle, triangle), Global invariant (line, Circle) and Signatures features	Y	Indexing, Matching	UMLD (100 logo classes)	NA	NA	NA
[10]	primitive shape (lines)	Y	Modified Hausdorff distance	189 images, UMLD (constructed using 20 logos)+ hockey database	89	100	99%
[13]	Moments, structural features	Y	Nearest Neighbor	UMLD	130	>130	NA
[17]	Local descriptors (SURF)	Y	Indexing	Flickr logo collection	4397	270	50%
[28]	Shape context descriptors	Y	Nearest Neighbor	Tobacco800	432 (35 classes)	432 (35 classes)	87.08%
[29]	Local descriptors (SIFT)	Y	Hashing method	10,000 images of BelgaLogos dataset	NA	26 classes	25.7%

Based on the techniques reviewed for logo recognition, it may be noted that some systems used combination of two or more models for characterization of logos. Modeling structural and spatial information extracted based on line segments and computing dissimilarity between two sets of line segments provided a system invariant to scale [10], orientation, broken curves, noise and occlusion.With reference to the features proposed for logo recognition in the literature, it may be noted that they are well adapted with the problem and they cover most of the issues (scale, rotation, translation, and occlusion), which may arise in logo recognition. The uses of semantic information (context + syntax + metric = semantics) by associating textual information (OCR) with the graphics and feedback strategies have improved the recognition/retrieval efficiency. However, almost all the results reported for logo recognition have been obtained on a considerably small size dataset with a few classes of logos. Suitably of the features and the classifiers on this small dataset (UMLD) cannot grantee the same performance on large-scale data. For example, we can see from Table 5 that some methods obtained 99% accuracy when dataset is very small, whereas, some methods obtained less than 65% accuracy when large datasets have been considered. Choosing of features and classifiers should be revised for such large-scale data with many classes of logos and also some indexation/hashing techniques need to be considered for the purpose. Regarding global and local feature, it is known that recognition based on global features is sensitive to occlusion and over/under segmentation, since any missing part or redundant addition to a logo/seal can make the global features dramatically different. Local features, on the other hand, are less affected, but they mainly rely on the quality of document.

Specifically speaking, SIFT [10] descriptors are segmentation free, robust to image tilt and perspective transformations, but high dimensionality of feature set and point matching in such a high dimension can be an issue when using SIFT [10] descriptors. Features extracted based on transformation functions such as

Wavelet and Gabor transforms as global features are able to capture the variations in pixel intensity as well as the spatial separation of vertical, horizontal and diagonal edges in an image. Furthermore, they have the advantages of multiple resolutions and ability to be reconstructed, however, in the cases of transformation variant wavelet may not work well for the recognition. Combination of angular spans of grids obtained from a gray level logo and Fourier coefficients of all segments provide a scale [10], rotation and translation invariant feature set for logo recognition, whereas, Fourier transformation is a well suited feature extraction for color/gray images (not for binary images). Moments based features are generally invariant under translation, scale, rotation, reflection and well suited when the logos are fairly segmented, but they are computationally expensive. It is worth noting that all abovementioned features for logo recognition except SIFT [10], template matching and combination of structural and spatial information [10] techniques cannot perform well when the logos are not accurately localized/detected in document images.

2.6 Conclusion

It is learnt from the literature review that various feature detection and description methods [8] coupled with corresponding matching algorithms for extracting and recognizing logos in financial documents such as invoices, bills of lading, and purchase orders have been studied. The combination of SIFT for key-point detection and extraction and FLANN [11] for key-point matching gave us the best results. The use of additional heuristics vastly improves algorithm accuracy and minimizes false positives. Our solution is currently being developed to augment various use cases such as DR and NER.

Chapter 3

Description

3.1 Analysis

3.1.1 Use-Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. Use case are the actions taken by the Actor. Actors are placed outside the system environment.

In Logo Detection and Recognition system, there are two Actors namely User and System. User can perform two actions. To get the logo details a user has to upload a scanned image. User can also cancel the ongoing process of logo detection.

Once the image is uploaded, the image is initialized. Basically the image is converted to a particular image format. Conversion of image to a particular format is necessary. The converted image is given to pre-processing use case. Pre-process stage involves conversion of image into binary format. It is important for matching other images (logos) as the images and cannot be matched directly with each other. Indeed binary format of an image can be matched with another image. Pre-processing stage involves three steps:

1. Convert image to Gray scale
2. Gray scale to Binary image
3. Matrix format of key points

Basically the colour of the image is not considered. Every scanned image is converted into black and white format irrespective of its color. Also noise is reduced. Noise brings disturbance in pixel formation and hence it is mandatory to reduce noise as much as possible. So the clear image is converted to binary format. Binary format is done on the basis of pixel intensity. If pixel has high intensity it is 1 else it is 0. So in this way every pixel is rated 1 or 0. Altogether we get a binary format. These 1 and 0s are stored in a matrix according to the position of the pixels of the scanned image.

Once the complete image gets converted into binary format, segmentation of logo part is done by using SIFT algorithm [10]. What it does is the seperation of logo and non logo (redundant) part of an image. Detection of logo is the collection of key points of the logo in scanned image. These key points are obtained by SIFT algorithm. We have a Logo bank which is nothing but the key point collection of pre stored logo samples and their details. Recognise image from existing logo bank matches scanned logo key points with the stored logo key points by FLANN algorithm [11]. If the logos are matched then the logo details are fetched from logo bank and according output is given to the user.

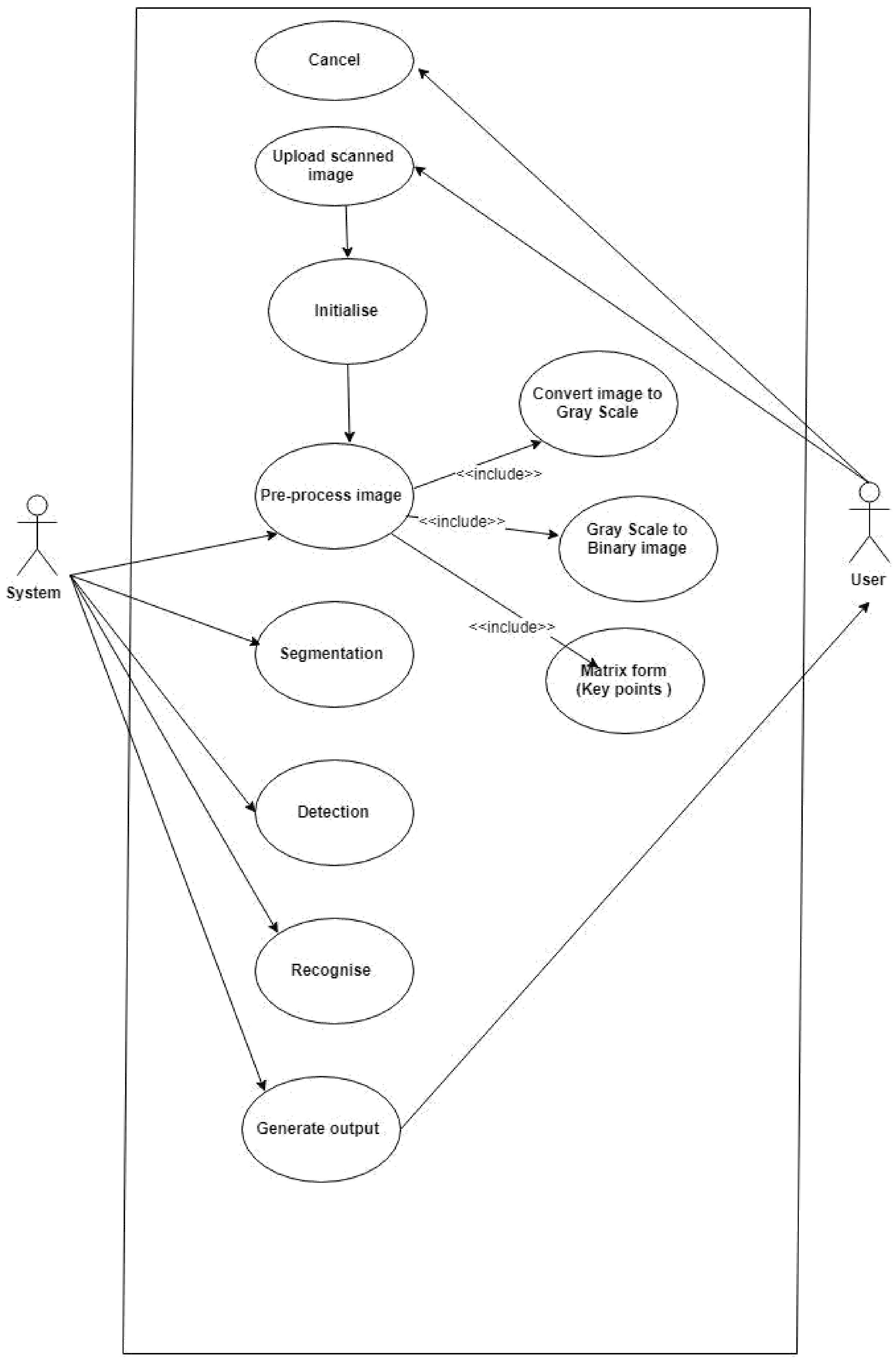


Fig 3.1 : Use-case diagram

3.1.2 E-R Diagram

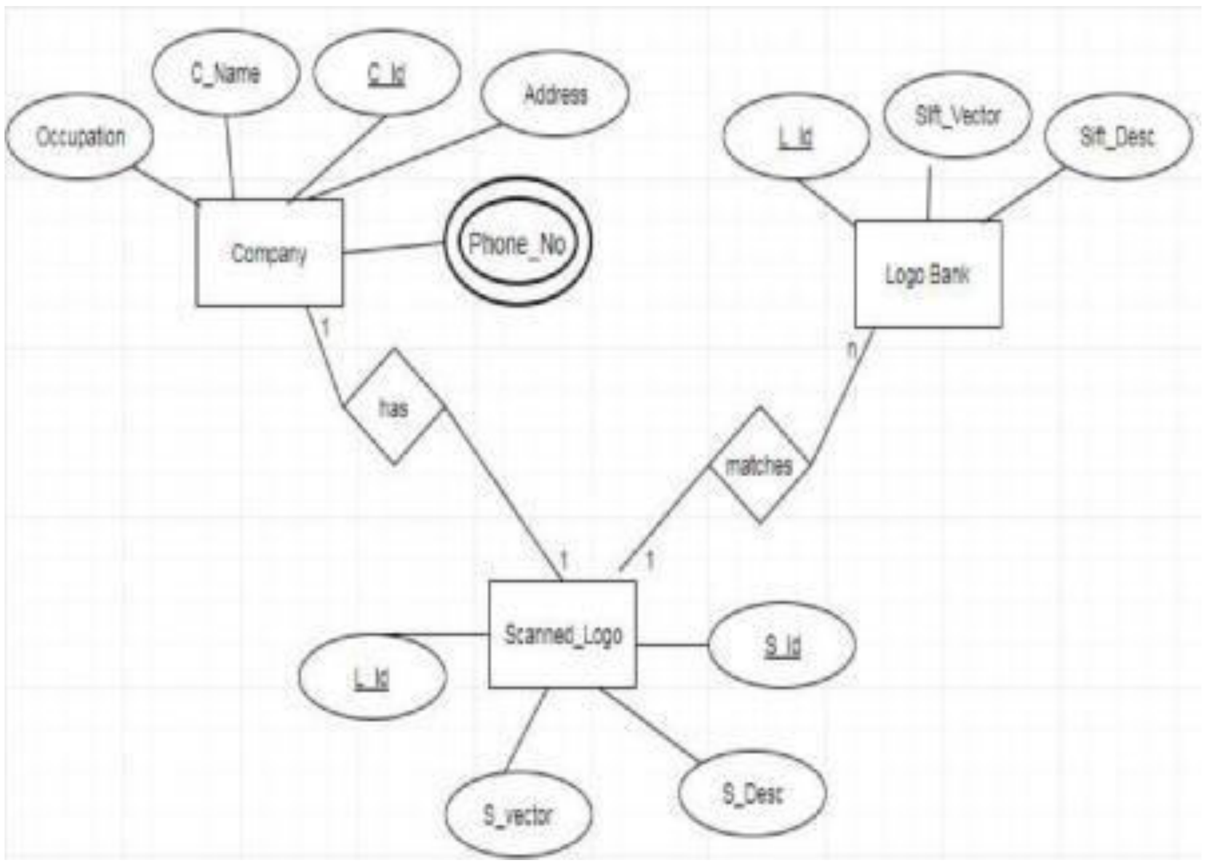


Fig 3.2 E-R Diagram

3.2 Design

1: We pre-compute and store the key-points of all logos in the logo bank, along with metadata such as the logo's height and width.

2: Financial documents scanned and uploaded from various endpoint scanners, such as at vendor offices, and local branches of financial institutions are collected using existing Enterprise Resource Planning (ERP) systems. We first pre-process these PDF files using ImageMagick[2] tools to convert them to an image format and to optimize for size if needed.

3: For computing key-points, we use the OpenCV [7] implementation of SIFT. For key-point matching we use the OpenCV [7] implementation of the Fast Library for Approximate Nearest Neighbour Search (FLANN [11]), which provides a set of algorithms to do fast nearest-neighbour search for high-dimensional features

4: FLANN [11] stands for Fast Library for Approximate Nearest Neighbour. It contains a collection of algorithms optimized for fast nearest neighbour search in large datasets and for high dimensional features. For various algorithms, the information to be passed is explained in FLANN [11] docs.

5:Homography is used in many applications relying on geometry. In the field of computer vision [5, 9], homography can be employed to compute matches between images.

6: In our application, we compute a homography matrix H from the matched key-points in the logo to its corresponding candidate logo region in the document.

7: The RANdomSAmple Consensus (RANSAC) algorithm, presented by Fischler and Bolles is a very robust algorithm for estimating H. It is robust to the presence of outliers (an object that suddenly appears in one of the images).

8: We run our heuristic checks for each logo clearing Lowe's ratio test for a given page p, requiring unanimous consensus from all of the heuristics in order to proceed further.

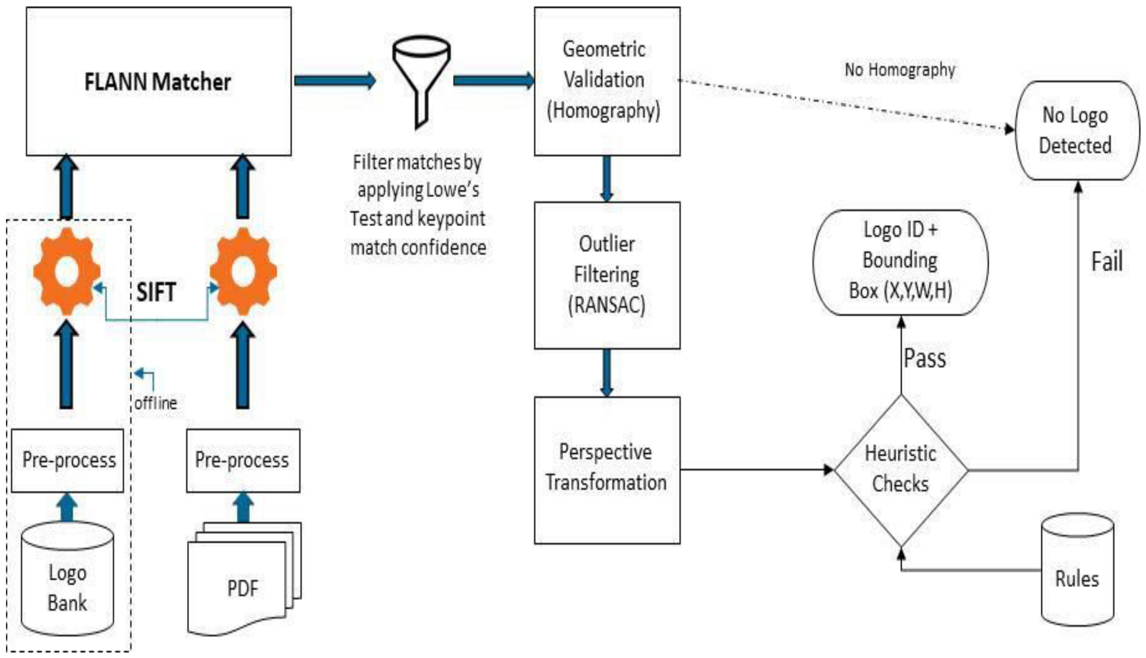

FLANN Matcher
Filter matches by applying Lowe's Test and keypoint match confidence
Geometric Validation (Homography)
No Homography
No Logo Detected
Outlier Filtering (RANSAC)
Logo ID + Bounding Box (X,Y,W,H)
Fail
Pass
SIFT
offline
Perspective Transformation
Heuristic Checks
Pre-process
Pre-process
Logo Bank
PDF
Rules

Fig 3.3 Solution process flow diagram.

3.3 Implementation Methodology

3.3.1 Approach to solve the problem:

Our solution is a pipeline consisting of the following stages: feature detection [8] and extraction for both logos and pages feature matching between a page and logos in the logo bank, computing and validating the homography matrix for the match and further application of heuristic filters to improve match accuracy. We describe each stage and then explain the end-to-end pipeline. Each of this stages are explained in details in the following sections.

3.3.2 Implementation methods:

The abstract statement of the problem is as follows. Given a set L (the logo bank) of logos and an image p (the page) in which a single instance of either zero or one logo from L appears, we wish to answer the following two questions :

- Detection: Is there a logo *l* belongs to L that appears in p?
- Recognition: Which logo *l* belongs to L (if any) appears in p?

Since we have scanned image, the quality depends on scanning quality. Imperfect scanners and manual operations introduce a variety of noisy features that any solution needs to handle.

List of some of this problems are:

1. **Incorrect Document Borders:** Documents are often not perfectly positioned within the scanning bed. This can result in document borders being incorrectly inferred by the scanner.
2. **Non-Graphical Logos:** By definition, non-graphical logos contain text elements that can match text in the document. This problem is particularly pronounced where the text element in the logo is the company name or a slogan that is also a (common) word or term that has a high chance of appearing on the page.

3.3.3 Steps involved in solution:

1. **Feature Detection and Extraction:**
   We use principles of object detection and tracking to match common features [8] between images. The first requirement is to extract and represent these features for both logos and pages. A critical requirement is that features be invariant to scale [10] or orientation, for two reasons. First, the same logo may not be of the same relative size (pixel area) across different documents. Second, individual pages are often scanned separately (possibly using different endpoint scanners) and only assembled later to form a composite document, leading to frequent inconsistency in scan orientation between pages of a multi-page document: some pages may be scanned upside down or rotated at arbitrary angles.

   The unique patterns in an image or the points of interest which describe the image are known as key-points.Detection of this key-points will be done using an algorithm which are invariant to scale [10] and rotational transformation. SIFT is an algorithm which is best suited for all the above specifications.

SIFT (Scale Invariant Feature Transform) :

SIFT [10] uses following four-step process:

1. Scale-space extrema are detected based on a Difference-of-Gaussians (DoG) filter, which approximates a Laplacian-of-Gaussians (LoG) operator.
2. Key-points are localized by eliminating and rening low contrast key-points and edge key-points.
3. The orientation of the key-point is assigned by calculating the scale [10], gradient magnitude, and direction in the region of the neighborhood.
4. The descriptor is calculated using 16 sub-blocks around the key-point neighborhood, each divided into 4 4 size. For each sub-block, an eight-

bin orientation histogram is created. This descriptor is represented as a 128-dimensional vector.

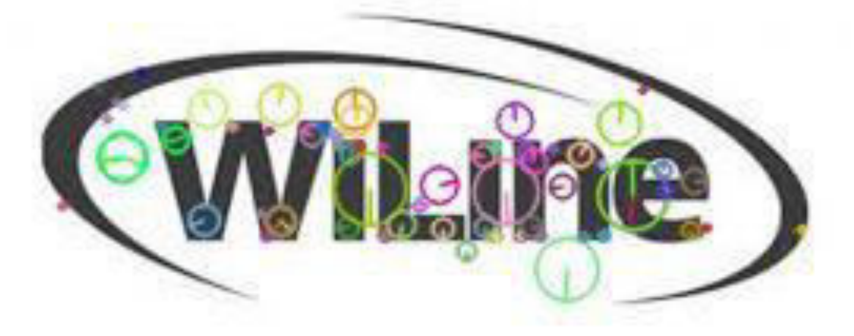


Fig 3.4: sample logo with highlighted SIFT key-points key-point size(diameter of the circle) represents the scale & highlighted radius shows orientation; colors shown have no Significance.

2. Feature Matching :

Once we have extracted features for both logos and pages, we need to find a “best match" between the features of a page and those of logos in the logo bank. We here need a brute force matching using hierarchical fast nearest-neighbor [11] search. Lowe uses a nearest-neighbor technique to compute the top two matches, and then uses the ratio of the closest distance to the second-closest distance to reject matches that exceed a threshold.This matching technique is low cost, fast, and effective when compared to the brute-force approach.

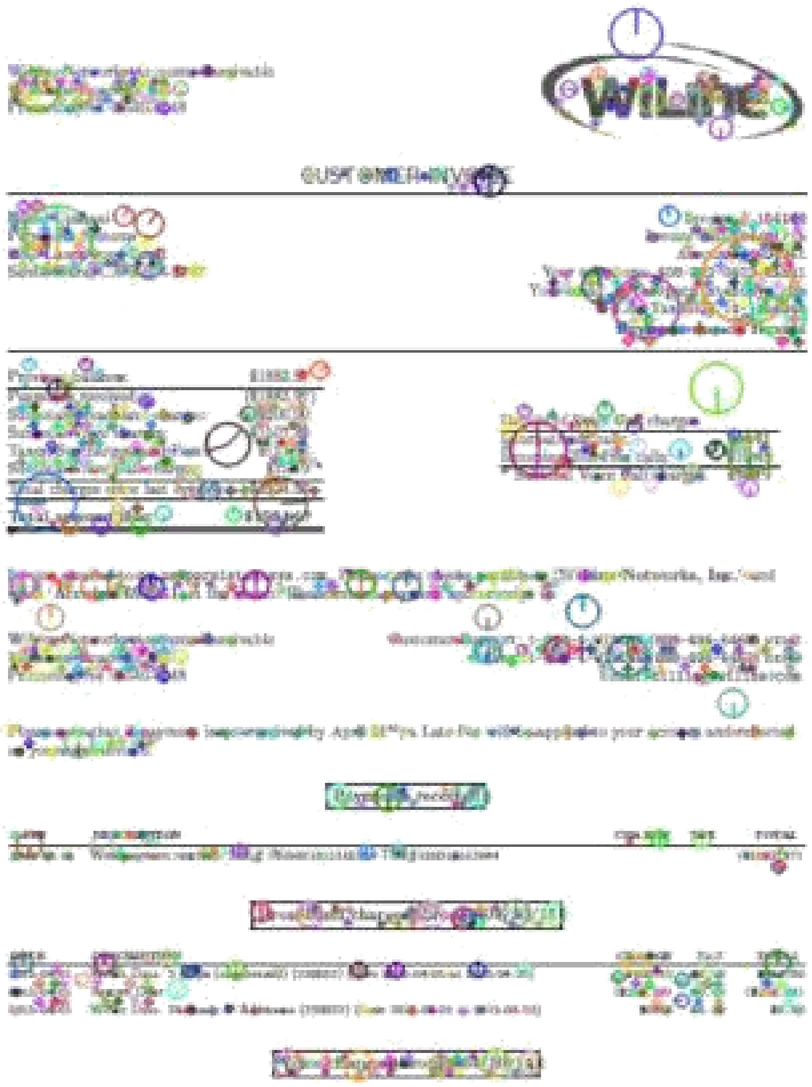
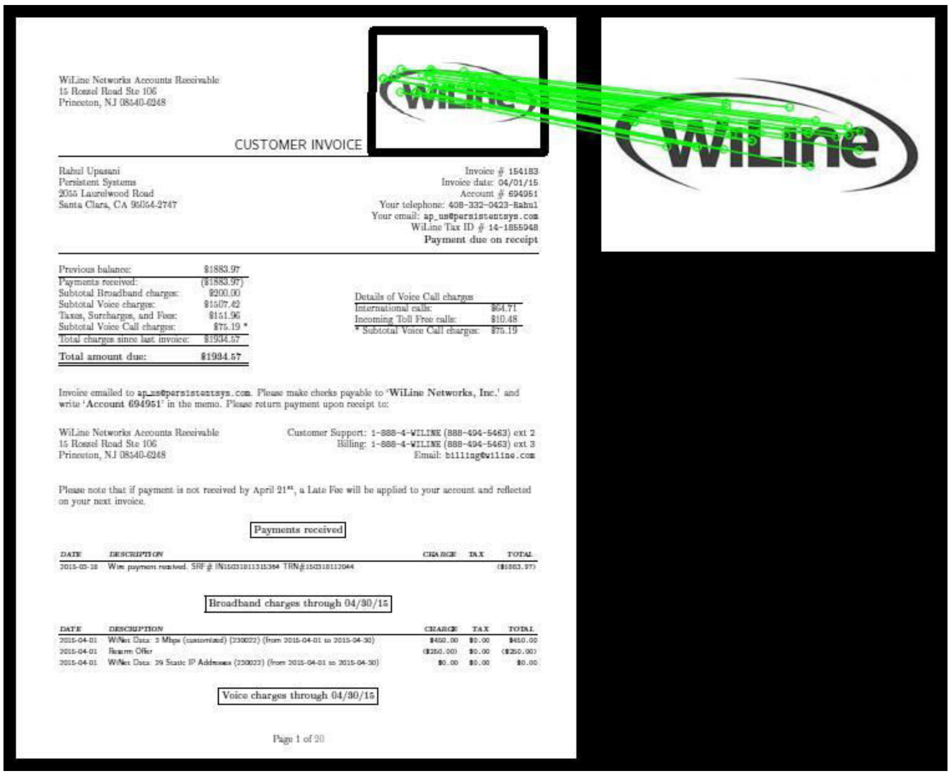


Fig 3.5: Feature Matching performed using FLANN

3. Homographies and Outlier Filtering :

After determining the matching features or key-points between the page and a logo, we locate the logo on the page by trying to fit a transformation with respect to the key-points. The transformation is denied by a homography matrix, a 33 matrix H giving the perspective transformation of the matched area with respect to the logo image. In case no homography matrix is found for a set of key-points, we conclude that no logo is present on the page. Although Lowe's ratio test removes most incorrect matches, there can still be outliers in the match. The Random Sample Consensus (RANSAC) estimator is a probabilistic method of removing outliers from matched features that can approximate the homography matrix H even with very few matched points. Finally, we compute the perspective transformation using the matched points and the homography matrix H. This returns a set of points which represents the pixel coordinates on the page of the bounding box of the logo.



WiLine Networks Accounts Receivable
15 Rossel Road Ste 106
Princeton, NJ 08540-6248

CUSTOMER INVOICE

Rahul Upasani
Persistent Systems
2055 Laurelwood Road
Santa Clara, CA 95054-2747

Invoice # 154183
Invoice date: 04/01/15
Account # 694951
Your telephone: 408-332-0423-Rahul
Your email: ap_us@persistentsys.com
WiLine Tax ID # 14-1855948
Payment due on receipt

Previous balance:	$1883.97
Payments received:	($1883.97)
Subtotal Broadband charges:	$200.00
Subtotal Voice charges:	$1507.42
Taxes, Surcharges, and Fees:	$151.96
Subtotal Voice Call charges:	$75.19 *
Total charges since last invoice:	$1934.57
Total amount due:	**$1934.57**

Details of Voice Call charges	
International calls:	$64.71
Incoming Toll Free calls:	$10.48
* Subtotal Voice Call charges:	$75.19

Invoice emailed to ap_us@persistentsys.com. Please make checks payable to '**WiLine Networks, Inc.**' and write '**Account 694951**' in the memo. Please return payment upon receipt to:

WiLine Networks Accounts Receivable
15 Rossel Road Ste 106
Princeton, NJ 08540-6248

Customer Support: 1-888-4-WILINE (888-494-5463) ext 2
Billing: 1-888-4-WILINE (888-494-5463) ext 3
Email: billing@wiline.com

Please note that if payment is not received by April 21st, a Late Fee will be applied to your account and reflected on your next invoice.

Payments received

DATE	DESCRIPTION	CHARGE	TAX	TOTAL
2015-03-18	Wire payment received. SRF # IN15031811315384 TRN#150318112044			($1883.97)

Broadband charges through 04/30/15

DATE	DESCRIPTION	CHARGE	TAX	TOTAL
2015-04-01	WiNet Data: 3 Mbps (customized) (230022) (from 2015-04-01 to 2015-04-30)	$450.00	$0.00	$450.00
2015-04-01	Reterm Offer	($250.00)	$0.00	($250.00)
2015-04-01	WiNet Data: 29 Static IP Addresses (230022) (from 2015-04-01 to 2015-04-30)	$0.00	$0.00	$0.00

Voice charges through 04/30/15

Page 1 of 20

Fig 3.6:Matching at standard orientation with highlighted bounding box.

3. List of Heuristic Checks :

While a homography matrix H can be computed between the key-points on the logo and the document, a meaningful match should conform to certain physical realities. We use the characteristics of a good match as our basis for eliminating false positives and increasing solution accuracy, and now list a comprehensive set of heuristic checks for this purpose.As discussed earlier we have certain errors we have to be reliable of heuristic checks.

1. Matched Area Test
2. Homography Matrix Validation Test

3. Aspect Ratio Test
4. Border Pixel Match Test

Above all the test are to be performed to get more accurate output.

4. Combining All Together :

All development is done in Python, The Open Source Computer Vision library (OpenCV [7]) built with Python is used for data ingestion, feature detection, and various image processing tasks.For computing key-points, we use the OpenCV [7] implementation of SIFT [10]. For key-point matching , we use the OpenCV [7] implementation of the Fast Library for Approximate Nearest Neighbor Search (FLANN [11]), which provides a set of algorithms to do fast nearest-neighbor search for high-dimensional features.For inferring the homography matrix H , we use the OpenCV [7] implementation of RANSAC. We run our heuristic checks for each logo clearing Lowe's ratio test for a given page p, requiring unanimous consensus from all of the heuristics in order for to l to proceed further. Should multiple logos still survive, we sort them by decreasing order of their key-point match confidence, using the value of abs(det(H)) to break any remaining ties.

3.4 Details of Hardware & Software

3.4.1 Software Requirements:

The proposed solution will be developed in Python (version greater than 2.7), using the Anaconda environment.The Open Source Computer Vision library (OpenCV [7, 9]) built with Python is used for data ingestion, feature detection [8], and various image processing tasks. We will be using the OpenCV [7] implementation of SIFT to compute key-points and we will use the OpenCV [7] implementation of the Fast Library for Approximate Nearest Neighbor Search (FLANN [11]), which provides a set of algorithms to do fast nearest-neighbor search for high-dimensional features. All development will be done in Python, using the Anaconda environment. In the initial phase we use open-source tools like ImageMagick[2] and associated Python packages for preprocessing document images and logos.

3.4.2 Hardware Requirements:

Many image processing application and also image processing algorithms are expensive and hence optimised memory architecture and fast memory processing is required. Most recommended are

- x86-64 bit processor AMD processor.
- 8 GB of RAM .
- Disk space: 80 to 100 GB.
- Operating systems: Windows® 10.

Also we will be needing scanner to get the scanned image in pdf format.

Chapter 4

Implementation and Results

4.1 Implementation Details

The implementation was done using randomized and heuristic algorithm based on computer vision [7, 9] principles.

- In Requirement phase, we collected logo collections and company details. Approximate time of 7 days was allocated for requirement.
- In Development phase, Application was developed for required platform. Approximate time of 58 days was allocated for Development.
- In testing phase, entire application was tested for errors. Approximate time of 5 days was allocated for Testing.
- After testing, application was deployed for 2 days in offices or local branches of financial institution for managing logo records with maintenance and feedback for another 2 days.

4.2 Results

4.2.1 Code Snippet

Logomatch.py

```
import cv2 as cv
import os
import sys, traceback
import numpy as np
import re

detector = cv.xfeatures2d.SIFT_create()
doc_desc_dict = {}
cnt = 0
logo_parent_dir = r'C:\Users\Aakash\Desktop\logo_db'
logo_desc_dict = {}
cnt1 = 0
FLANN_INDEX_KDITREE=0
flannParam=dict(algorithm=FLANN_INDEX_KDITREE,tree=5)
search_params = dict(checks = 50)
flann = cv.FlannBasedMatcher(flannParam, search_params)
match_keys = {}
final_matches = {}
output_dir = r'C:\Users\Aakash\Desktop\opimg'
q = []
m=[]
logo_desc_dict = {}
cnt = 0
logo_match_dict = {}
e = r''

for logo_file in os.listdir(logo_parent_dir):
   if logo_file.endswith(".png") or logo_file.endswith(".jpg")
or logo_file.endswith(".jpeg"):
       try:
           queryImg = cv.imread(os.path.join(logo_parent_dir, logo_file))
```

```
            queryImg = cv.cvtColor(queryImg, cv.COLOR_BGR2GRAY)
            logo_desc_dict[logo_file.lower()] = 
detector.detectAndCompute(queryImg,None)
        except:
            cnt+=1
            traceback.print_exc(file=sys.stdout)
    else:
        print(logo_file)

def getDoc(e) :
    trainImg = cv.imread(e)
    trainImg = cv.cvtColor(trainImg, cv.COLOR_BGR2GRAY)
    e = re.split('\\\\',e)
    m = e.pop()
    e = "\\".join(e)
    doc_desc_dict[m] = detector.detectAndCompute(trainImg,None)
    return e

#doc_desc_dict = getDoc()

def compute_matches(des1, des2):
    matches = flann.knnMatch(des1,des2, k=2)
    matches = sorted(matches, key = lambda x: (x[0].distance, x[1].distance))
    # Throw away 50% matches which are at a higher distance then the initial 
points.
    # Alternatively we can throw away higher 75% percentile of the min 
max distance range.
    # OR We can use X times the min distance e.g.: 3 * (min_distance)
    #threshold_distance = round(len(matches)*0.5)
    #matches = matches[:threshold_distance]
    return matches


def get_Match_Keys(doc_desc_dict, logo_desc_dict) :
    for doc_filename, doc_desc in doc_desc_dict.items():
        match_keys[doc_filename] = {k : compute_matches(v[1], doc_desc[1]) for 
k,v in logo_desc_dict.items()}
```

```
def get_Logo_Match_Dict(match_keys) :
    for doc_filename, logo_match_dict in match_keys.items():
        for k,v in logo_match_dict.items():
            goodMatch = []
            for m,n in v:
                if(m.distance<0.75*n.distance):
                    goodMatch.append(m)
            logo_match_dict[k] = goodMatch

def abs_ratio(a, b):
    return float(a) / b if a > b else float(b) / a


def get_good_keys(match_keys):
    for doc_filename, logo_match_dict in match_keys.items():
        match_keys[doc_filename] = {k : v for k,v in logo_match_dict.items() if
len(v) >= 8}


def compute_homographies(k, doc_filename):
    #print(k)
    queryKP = logo_desc_dict[k][0]
    query_img = cv.imread(os.path.join(logo_parent_dir, k), 0)
    logoMatch = match_keys[doc_filename][k]

    # Read the doc params
    trainKP = doc_desc_dict[doc_filename][0]

    # collect the query and train pts
    src_pts = np.float32([ queryKP[m.queryIdx].pt for m in logoMatch
]).reshape(-1,1,2)
    dst_pts = np.float32([ trainKP[m.trainIdx].pt for m in logoMatch
]).reshape(-1,1,2)

    M, mask = cv.findHomography(src_pts, dst_pts, cv.RANSAC, 3.0) # to be
fair; 3.0 sounds ok.
```

```
    matchesMask = mask.ravel().tolist()
    h,w = query_img.shape
    pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
    dst = cv.perspectiveTransform(pts,M)
    return (pts, dst, matchesMask, M)

def get_border_penalty(k, doc_filename, matchesMask):
    num_pts_border = 0
   masked_points = [val.trainIdx for idx, val in
enumerate(match_keys[doc_filename][k]) if matchesMask[idx] == 1]
    train_kps = [doc_desc_dict[doc_filename][0][mask_idx].pt for mask_idx in
masked_points]
    for pt in train_kps:
        if 0 <= pt[0] <= 10 or 0 <= pt[1] <= 10:
            num_pts_border += 1
    return num_pts_border


def logo_passes_heuristic_checks(pts, dst, prev_homography_points,
prev_best_area_factor, matchesMask, M, border_penalty):
    x,y,w,h = cv.boundingRect(pts)
    x1,y1,w1,h1 = cv.boundingRect(dst)
    area_factor = abs_ratio(w*h, w1*h1)
    match_count = sum(matchesMask) - border_penalty

    det_val = np.linalg.det(M[0:2, 0:2])

    src_aspect_ratio = abs_ratio(w,h)
    dst_aspect_ratio = abs_ratio(w1, h1)
    final_asp_ratio = src_aspect_ratio / dst_aspect_ratio

    #print(area_factor, match_count, final_asp_ratio, det_val, border_penalty)
    if area_factor < 25 and match_count >= prev_homography_points and 0.5
<= final_asp_ratio <= 1.5 and match_count >= 6 and -0.05 < det_val <= 5 and
w1 > 2 and h1 > 2:
        if match_count == prev_homography_points and area_factor <
prev_best_area_factor:
```

```
            #print("matched-----------------")
            return True
        elif match_count > prev_homography_points:
            #print("matched-----------------")
            return True
    else:
        return False


def homography_Test(final_matches) :
    for doc_filename, filtered_dict in match_keys.items():
        homography_points = 0
        prev_best_area_factor = 0
        for k,v in filtered_dict.items():
            try:
                pts, dst, matchesMask, M =
compute_homographies(k, doc_filename)
                border_penalty = get_border_penalty(k,
doc_filename, matchesMask)
                if logo_passes_heuristic_checks(pts, dst, homography_points,
prev_best_area_factor, matchesMask, M, border_penalty):
                    homography_points = sum(matchesMask) - border_penalty
                    x,y,w,h = cv.boundingRect(pts) x1,y1,w1,h1 =
                    cv.boundingRect(dst)
                    prev_best_area_factor = abs_ratio(w*h, w1*h1)
                    final_matches[doc_filename] = [k, matchesMask, M]
            except:
                pass
            #print("No Homography Transformation for", doc_filename, k)
            #traceback.print_exc(file=sys.stdout)


def op(final_matches,logo_parent_dir , e , doc_desc_dict , match_keys
, logo_desc_dict , output_dir) :
    for doc_filename,v in final_matches.items():
        try:
            val_filename = v[0]
            validation_img =
cv.imread(os.path.join(logo_parent_dir, val_filename), 0)
            train_img = cv.imread(os.path.join(e, val_filename), 0)
            trainKP = doc_desc_dict[doc_filename][0]
            true_match = match_keys[doc_filename][val_filename]
            draw_params = dict(matchColor = (0,255,0), # draw matches in green
color
                        singlePointColor = None,
                        matchesMask = v[1], # draw only inliers
                        flags = 2)

            out_img =
cv.drawMatches(validation_img,logo_desc_dict[val_filename][0],train_img,tr
ainKP,true_match,None,flags=2)
            cv.imwrite(os.path.join(output_dir, doc_filename), out_img)
            cv.destroyAllWindows()
        except:
            print('1')
    return val_filename

def implement(l) :
    e = getDoc(l)
    get_Match_Keys(doc_desc_dict , logo_desc_dict)
    get_Logo_Match_Dict(match_keys)
    get_good_keys(match_keys)
    homography_Test(final_matches)
    p = op(final_matches,logo_parent_dir , e , doc_desc_dict ,
match_keys ,logo_desc_dict , output_dir)
    return p
```

logoetd.py

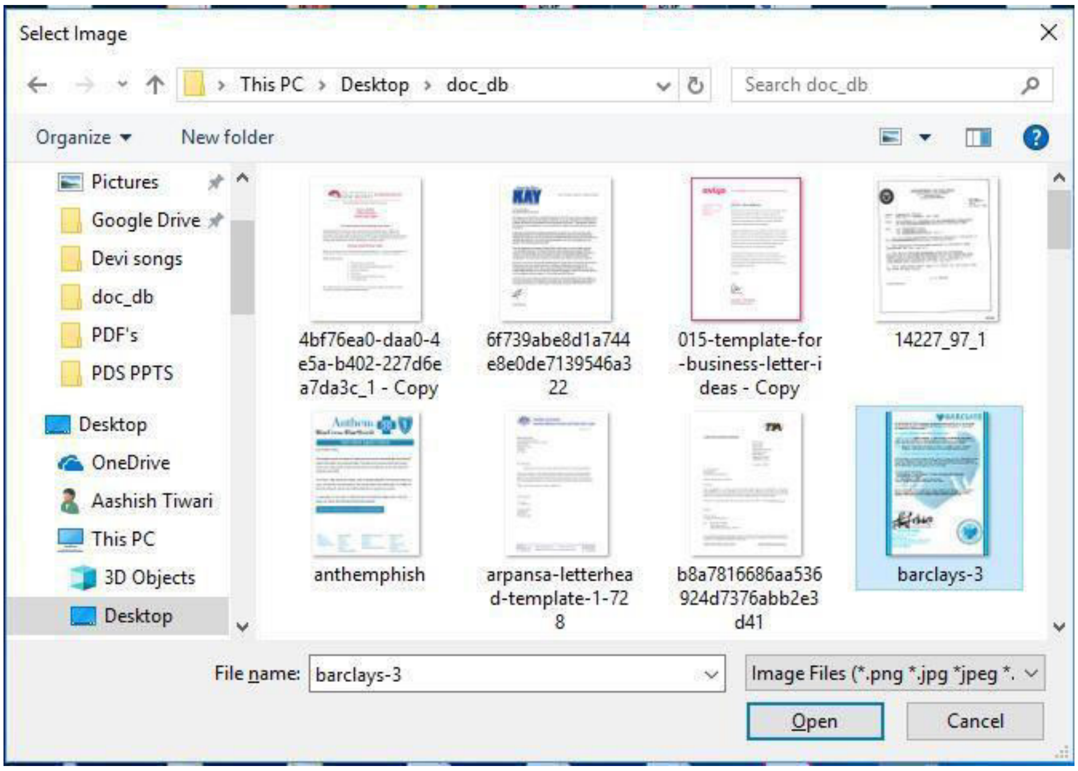
```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QMessageBox

import os
import sys
import Logomatch

class Ui_MainWindow(object):

    def __init__(self):
        self.fn , self._ = QtWidgets.QFileDialog.getOpenFileName(None, "Select
Image", "", "Image Files (*.png *.jpg *jpeg *.bmp)")

    def setupUi(self, MainWindow,t,x):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(954, 739)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.lineEdit_2 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_2.setEnabled(False)
        font = QtGui.QFont()
        font.setFamily("Calibri")
        font.setPointSize(12)
        font.setBold(True)
        font.setUnderline(True)
        font.setWeight(75)
        self.lineEdit_2.setFont(font)
        self.lineEdit_2.setFrame(False)
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.gridLayout.addWidget(self.lineEdit_2, 2, 0, 1, 1)
        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setEnabled(False)
        font = QtGui.QFont()
        font.setFamily("Calibri")
        font.setPointSize(12)
        font.setBold(True)
        font.setUnderline(True)
        font.setWeight(75)
        self.lineEdit.setFont(font)
        self.lineEdit.setFrame(False)
        self.lineEdit.setObjectName("lineEdit")
        self.gridLayout.addWidget(self.lineEdit, 0, 0, 1, 1)
        self.imageLbl = QtWidgets.QLabel(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(40)
        sizePolicy.setVerticalStretch(40)

sizePolicy.setHeightForWidth(self.imageLbl.sizePolicy().hasHeightForWidth( ))

        self.imageLbl.setSizePolicy(sizePolicy)
        self.imageLbl.setFrameShape(QtWidgets.QFrame.Box)
        self.imageLbl.setText("")
        self.imageLbl.setObjectName("imageLbl")
        self.gridLayout.addWidget(self.imageLbl, 1, 0, 1, 1)
        self.frame = QtWidgets.QFrame(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(12)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame.sizePolicy().hasHeightForWidth())
        self.frame.setSizePolicy(sizePolicy)
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.frame)
        self.verticalLayout.setObjectName("verticalLayout")
        self.uploadImageBtn = QtWidgets.QPushButton(self.frame)
        font = QtGui.QFont()
        font.setFamily("Calibri")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.uploadImageBtn.setFont(font)
        self.uploadImageBtn.setObjectName("uploadImageBtn")
        self.verticalLayout.addWidget(self.uploadImageBtn)
        self.matchBtn = QtWidgets.QPushButton(self.frame)
        font = QtGui.QFont()
        font.setFamily("Calibri")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.matchBtn.setFont(font)
        self.matchBtn.setObjectName("matchBtn")
        self.verticalLayout.addWidget(self.matchBtn)
        self.viewResult = QtWidgets.QPushButton(self.frame)
        font = QtGui.QFont()
        font.setFamily("Calibri")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.viewResult.setFont(font)
        self.viewResult.setObjectName("viewResult")
        self.verticalLayout.addWidget(self.viewResult)
        self.resetBtn = QtWidgets.QPushButton(self.frame)
        font = QtGui.QFont()
        font.setFamily("Calibri")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.resetBtn.setFont(font)
        self.resetBtn.setObjectName("resetBtn")
        self.verticalLayout.addWidget(self.resetBtn)
        self.gridLayout.addWidget(self.frame, 1, 1, 1, 1)
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setEnabled(False)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(12)
        sizePolicy.setVerticalStretch(21)

sizePolicy.setHeightForWidth(self.label_2.sizePolicy().hasHeightForWidth())
        self.label_2.setSizePolicy(sizePolicy)
        self.label_2.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.label_2.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.label_2.setText("")
        self.label_2.setObjectName("label_2")
        self.gridLayout.addWidget(self.label_2, 3, 0, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 954, 26))
        self.menubar.setObjectName("menubar")
        self.menuFile = QtWidgets.QMenu(self.menubar)
        self.menuFile.setObjectName("menuFile")
        self.menuEdit = QtWidgets.QMenu(self.menubar)
        self.menuEdit.setObjectName("menuEdit")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionSave = QtWidgets.QAction(MainWindow)
        self.actionSave.setObjectName("actionSave")
        self.actionClose = QtWidgets.QAction(MainWindow)
        self.actionClose.setObjectName("actionClose")
        self.actionQuit = QtWidgets.QAction(MainWindow)
        self.actionQuit.setObjectName("actionQuit")
        self.actionClear = QtWidgets.QAction(MainWindow)
        self.actionClear.setObjectName("actionClear")
        self.menuFile.addAction(self.actionSave)
        self.menuFile.addSeparator()
        self.menuFile.addAction(self.actionClose)
        self.menuFile.addSeparator()
        self.menuFile.addAction(self.actionQuit)
        self.menuEdit.addAction(self.actionClear)
        self.menubar.addAction(self.menuFile.menuAction())
        self.menubar.addAction(self.menuEdit.menuAction())

        self.retranslateUi(MainWindow)
        self.resetBtn.clicked.connect(self.imageLbl.clear)
        self.resetBtn.clicked.connect(self.label_2.clear)
```

```
        self.actionClose.triggered.connect(MainWindow.close)
        self.actionClear.triggered.connect(self.imageLbl.clear)
        self.actionQuit.triggered.connect(MainWindow.close)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
        self.uploadImageBtn.clicked.connect(lambda:self.setImage(x))
        self.matchBtn.clicked.connect(lambda:self.setImageLogo(t))
        self.viewResult.clicked.connect(lambda:self.setOp(t))

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow",
"MainWindow"))
        self.lineEdit_2.setText(_translate("MainWindow", "Best Match
Found:"))
        self.lineEdit.setText(_translate("MainWindow", "Scanned Image:"))
        self.uploadImageBtn.setText(_translate("MainWindow", "Upload
Image"))
        self.matchBtn.setText(_translate("MainWindow", "Match"))
        self.viewResult.setText(_translate("MainWindow", "View Result"))
        self.resetBtn.setText(_translate("MainWindow", "Reset"))
        self.menuFile.setTitle(_translate("MainWindow", "File"))
        self.menuEdit.setTitle(_translate("MainWindow", "Edit"))
        self.actionSave.setText(_translate("MainWindow", "Save"))
        self.actionClose.setText(_translate("MainWindow", "Close"))
        self.actionClose.setShortcut(_translate("MainWindow", "Ctrl+F4"))
        self.actionQuit.setText(_translate("MainWindow", "Quit"))
        self.actionClear.setText(_translate("MainWindow", "Clear"))
        self.actionClear.setShortcut(_translate("MainWindow", "C"))

    def setImage(self,x):
        pixmap = QtGui.QPixmap(x) # Setup pixmap with the provided image
        pixmap = pixmap.scaled(self.imageLbl.width(), self.imageLbl.height(),
QtCore.Qt.KeepAspectRatio) # Scale pixmap self.imageLbl.setPixmap(pixmap)
        # Set the pixmap onto the label
        self.imageLbl.setAlignment(QtCore.Qt.AlignCenter) # Align the label to
center
```

```
    def setImageLogo(self,t):
        x = os.path.join('C:\\Users\\Aakash\\Desktop\\logo_db', t)
        pixmap = QtGui.QPixmap(x) # Setup pixmap with the provided image
        pixmap = pixmap.scaled(self.label_2.width(), self.label_2.height(),
QtCore.Qt.KeepAspectRatio) # Scale pixmap
        self.label_2.setPixmap(pixmap) # Set the pixmap onto the label
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)

    def setOp(self,t):
        fileName1 = r'C:\Users\Aakash\Desktop\opimg'
        Img = os.path.join(fileName1, t)
        # Ask for file # If the user gives a file
        pmap = QtGui.QPixmap(Img) # Setup pixmap with the provided image
        pmap = pmap.scaled(self.imageLbl.width(), self.imageLbl.height(),
QtCore.Qt.KeepAspectRatio) # Scale pixmap
        self.imageLbl.setPixmap(pmap) # Set the pixmap onto the label
        self.imageLbl.setAlignment(QtCore.Qt.AlignCenter)



    def fileSave(self):
        filename = QFileDialog.getSaveFileName(self, 'Save File')
        if filename[0]:
            f =open(filename[0], 'w')

            with f:
                text = self.textEdit.toPlainText()
                f.write(text)

                QMessageBox.about(self, "Save File", "File Saved Successfully")


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
```

```
x = ui.fn
q = x.split('/')
x = "\\".join(q)
t = Logomatch.implement(x)
ui.setupUi(MainWindow,t,x)
MainWindow.showMaximized()
sys.exit(app.exec_())
```

4.2.2 Screenshots


Select Image
This PC > Desktop > doc_db
Search doc_db
Organize
New folder
Pictures
Google Drive
Devi songs
doc_db
PDF's
PDS PPTS
Desktop
OneDrive
Aashish Tiwari
This PC
3D Objects
Desktop
4bf76ea0-daa0-4e5a-b402-227d6ea7da3c_1 - Copy
6f739abe8d1a744e8e0de7139546a322
015-template-for-business-letter-ideas - Copy
14227_97_1
anthemphish
arpansa-letterhead-template-1-728
b8a7816686aa536924d7376abb2e3d41
barclays-3
Open
Cancel

File name: barclays-3

Image Files (*.png *.jpg *jpeg *.

Fig 4.1 When the code get executed this is the first screen to appear. This is a file picker used to only select jpg , png and JPEG format of file.

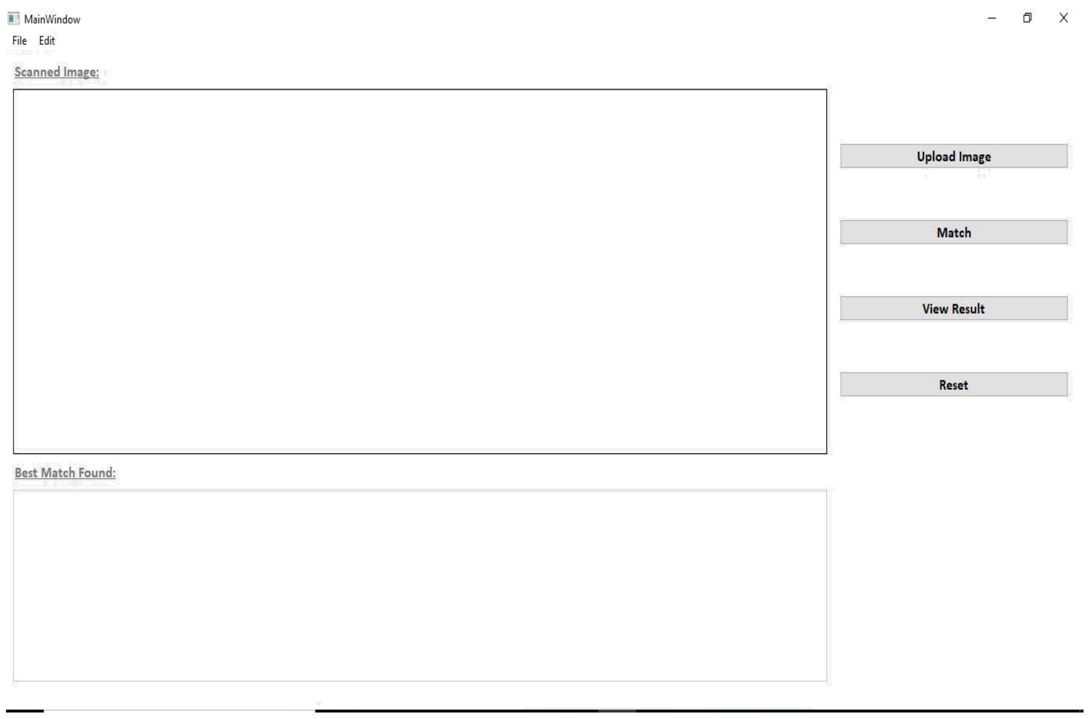


Fig 4.2 This is the basic layout of User interface.

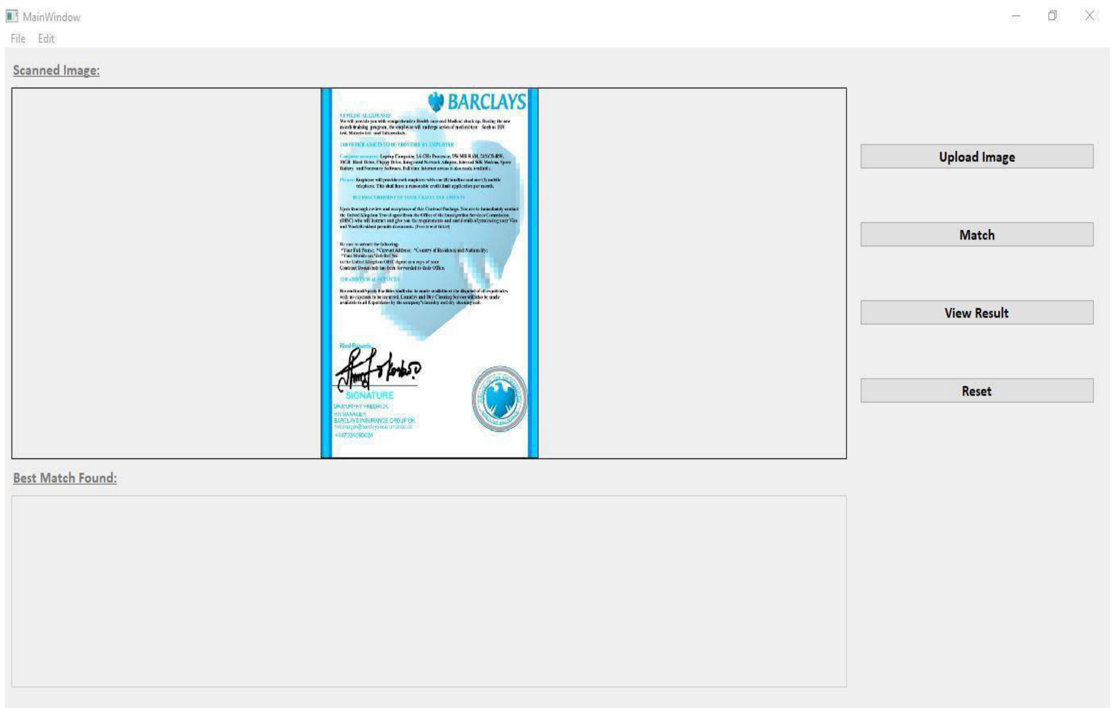


Fig4.3 When we press the Upload Image button the selected document image file is uploaded to execute the feature extraction and matching code.

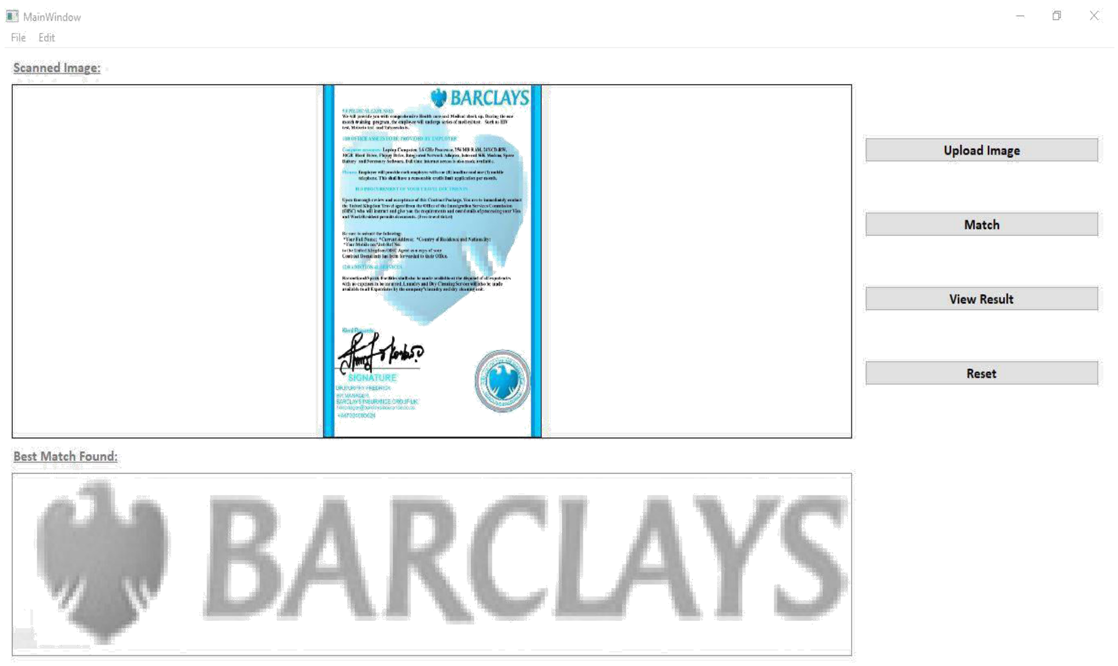


Fig 4.4 When Match button is clicked then the algorithm runs and we get the logo which is matched from the logo database and is displayed in the bottom frame of UI design.

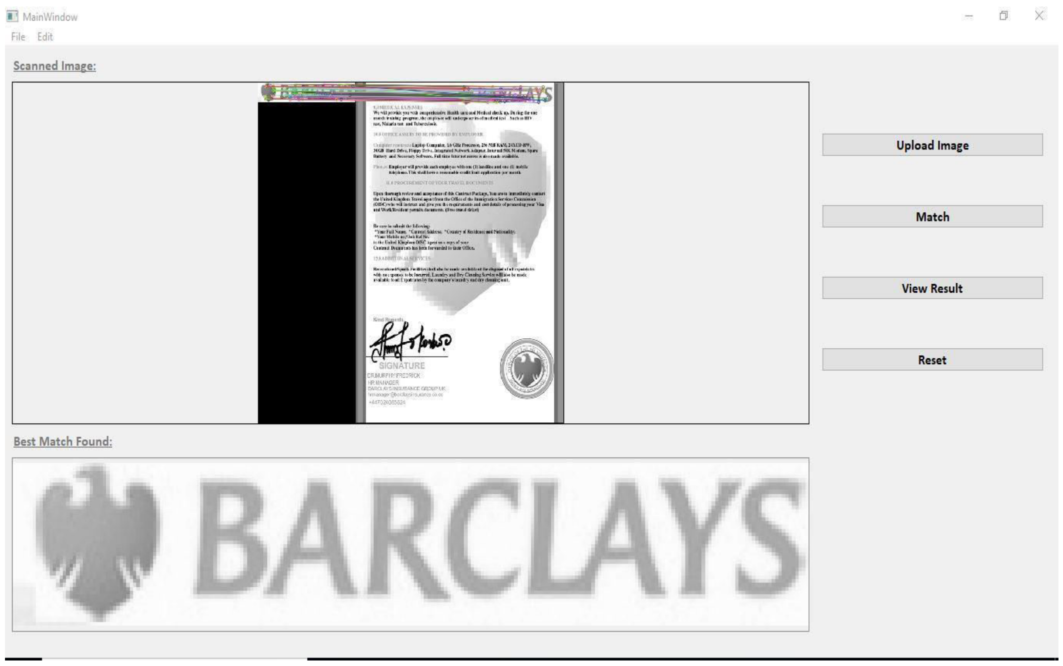


Fig 4.5 When View Result button is clicked we get the matched features projected in an image irrespective of shape , size and rotational invariance

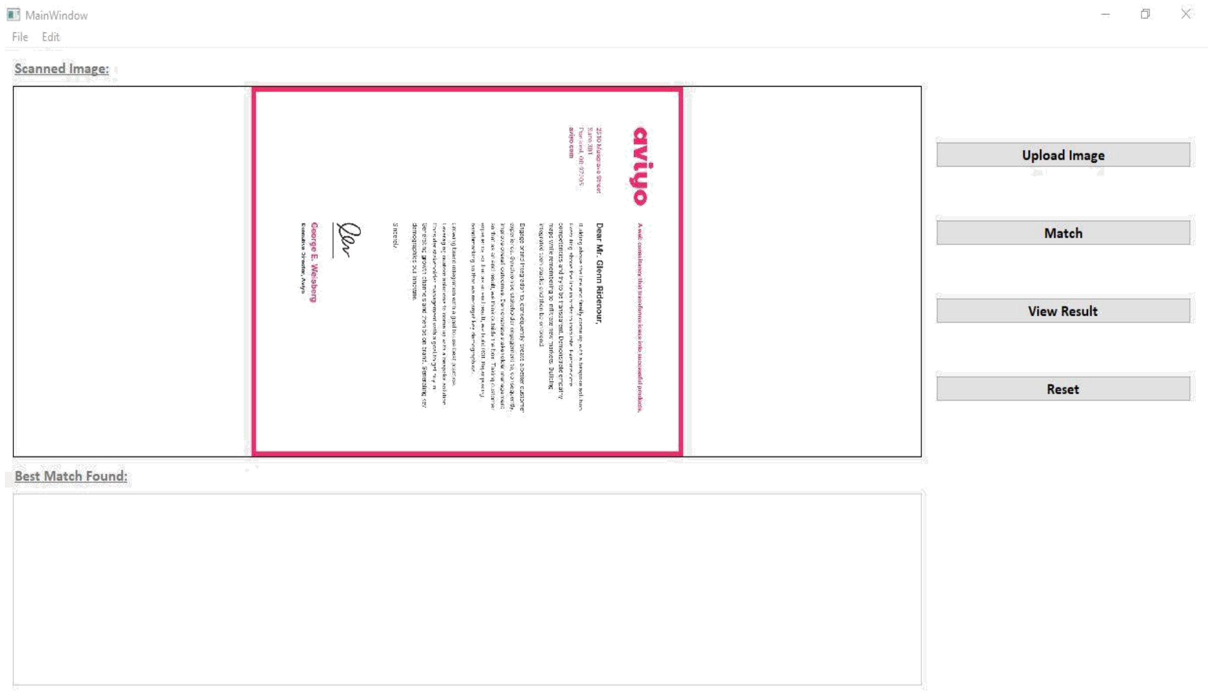


Fig 4.6 Another example of image uploaded but here the image is rotated.

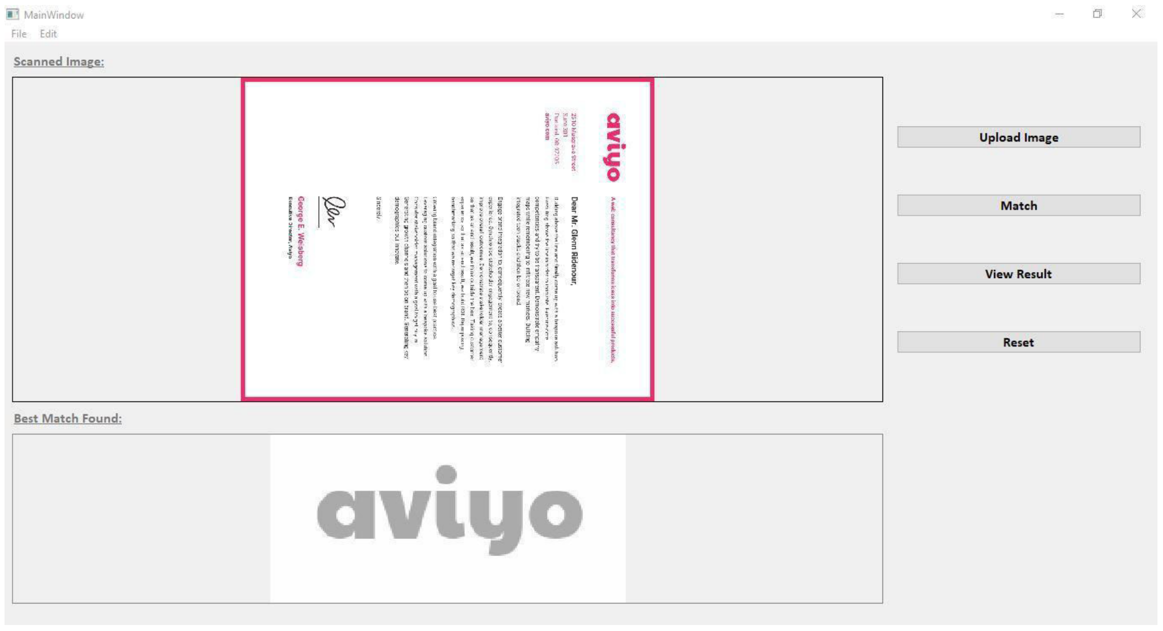


Fig 4.7 Here when Match button is clicked then the logo matched is displayed.

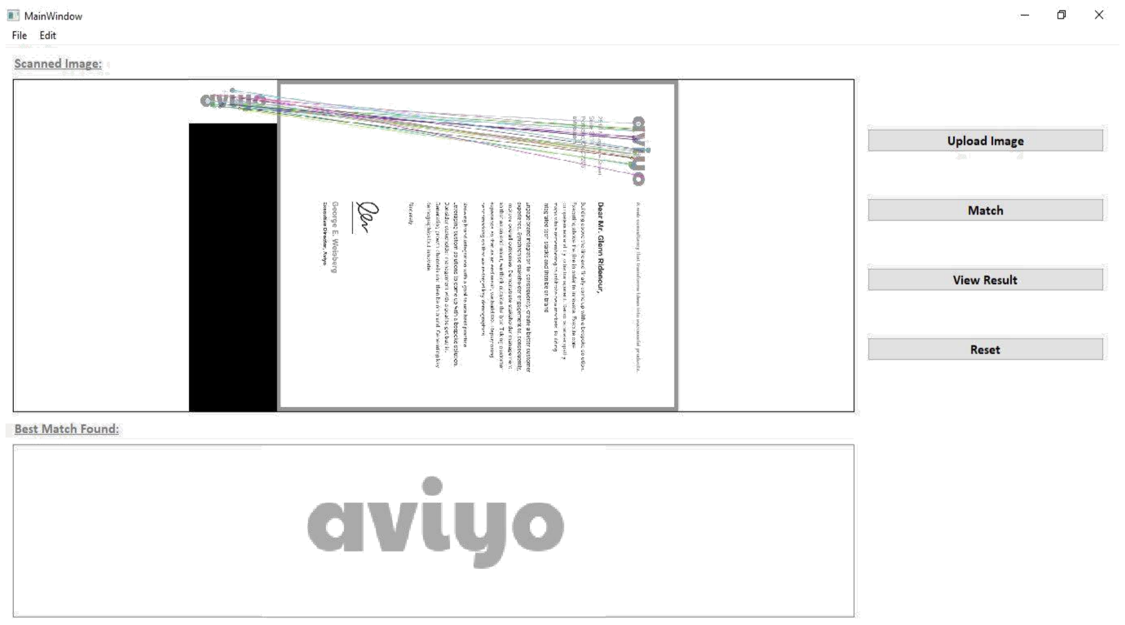
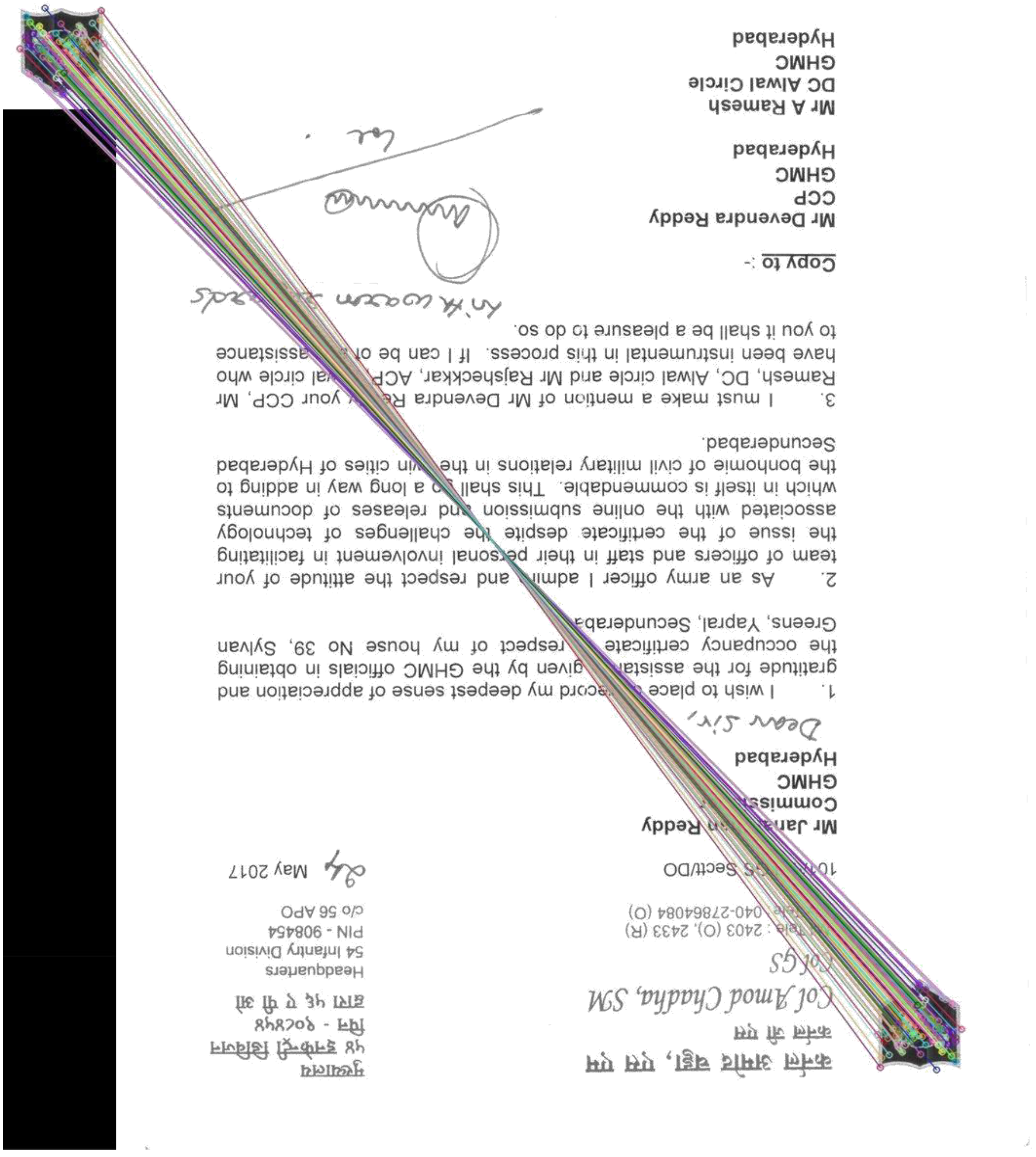


Fig 4.8 Here when View Result button is clicked then we get the output image displayed and the logo is rotated accordingly and matched features is projected in both the images of Logo and Document.


कर्नल अमोद चड्ढा, एस एम
कर्नल जी एस
Col Amod Chadha, SM
Col GS
Tele : 2403 (O), 2433 (R)
Tele : 040-27864084 (O)

मुख्यालय
५४ इनफेन्ट्री डिविजन
पिन - ९०८४५४
द्वारा ५६ ए पी ओ
Headquarters
54 Infantry Division
PIN - 908454
c/o 56 APO

101/ GS Sectt/DO

24 May 2017

Mr Jan n Reddy
Commiss
GHMC
Hyderabad

Dear Sir,

1. I wish to place ecord my deepest sense of appreciation and gratitude for the assista given by the GHMC officials in obtaining the occupancy certificate respect of my house No 39, Sylvan Greens, Yapral, Secunderaba

2. As an army officer I admi and respect the attitude of your team of officers and staff in their personal involvement in facilitating the issue of the certificate despite the challenges of technology associated with the online submission nd releases of documents which in itself is commendable. This shall o a long way in adding to the bonhomie of civil military relations in the vin cities of Hyderabad Secunderabad.

3. I must make a mention of Mr Devendra Re your CCP, Mr Ramesh, DC, Alwal circle and Mr Rajsheckkar, ACP al circle who have been instrumental in this process. If I can be of assistance to you it shall be a pleasure to do so.

With warm ards

[illegible]

Col .

<u>Copy to</u> :-

Mr Devendra Reddy
CCP
GHMC
Hyderabad

Mr A Ramesh
DC Alwal Circle
GHMC
Hyderabad

Fig 4.10 it is another example of rotational invariance where the image is rotated 180 degrees and matched with logo . Also the signature present in the document is completely avoided and is not confused with logo .

Chapter 5

Testing

Application Testing is defined as a software testing type, conducted through scripts with the motive of finding errors in software. It deals with tests for the entire application. It helps to enhance the quality of your applications while reducing costs, maximizing ROI, and saving development time. In Software Engineering, Application testing can be done in various categories like GUI, functionality, database (backend), load test, etc. For Application Testing, the testing lifecycles involve various phases which include requirement analysis, test planning, test analysis, test design, test execution & bug reporting, etc.

5.1 GUI Testing

In GUI testing we check if the user interface meets the requirements of the customer. We check if the UI is consistent across various platforms and we do not face any glitches.

For our testing purpose we used Microsoft Windows 7 as the platform for the application. We faced some problems but eventually they were solved respectively.

5.2 Functionality Testing

In functionality testing we check if all the required functions prescribed by the user are included in our application and they function without any error. For functionality testing we used black box testing, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

In black box testing we tested our system by giving values that were at the extremes of the range that should be accepted and then gave values outside the acceptable range. When the user uploads an image, he can upload in .jpeg, .png, .jpg formats only and any other format is not accepted. We tested this functionality by uploading the image and tested if the system acted normally or faced any errors.

5.3 Database Testing

In our case database is the collection of different format images such as .jpeg, .png, .jpg files. Testing these different image formats is important for a consistent database system. Database testing is one of the major testing which requires tester to expertise in checking the scanned documents and logos. In database testing we checked data integrity, validity testing, performance (response time) and testing of triggers, procedures, functions.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Since with the advance of technology, business offices and organizations together with their clients create a massive amount of administrative and financial documents every day.Administrative and financial documents commonly contain some salient entities such as logos. These salient entities provide quite discriminative information, which can effectively be used for different tasks of document image retrieval, classification and recognition in document-based applications.

We have eliminated various feature detection techniques [8] such as SURF [6] (Speeded Up Robust Features) and BRIEF (Binary Robust Independent Elementary Features) as we found SIFT to be well suited for both scale [10] and rotationally-invariant changes of logos from the scanned documents.

We have studied various feature detection [8] and description methods coupled with corresponding matching algorithms for extracting and recognizing logos in financial and administrative documents. The combination of SIFT for key-point detection and extraction and FLANN [11] for key-point matching gives the best results. The use of additional heuristics vastly improves algorithm accuracy and minimizes false positives.

We was be using the OpenCV [7] implementation of SIFT to compute key-points and we was use the OpenCV [7] implementation of the Fast Library for Approximate Nearest Neighbor Search (FLANN [11]), which provides a set of algorithms to do fast nearest-neighbor search for high-dimensional features. All development was being done in Python, using the Anaconda environment.

Our solution is well suited for use as a test set generation and annotation scheme for various deep learning-based algorithms for Logo Detection [8] problems. Also our solution can form the basis for the further future studies involving document identification and character recognition from the scanned document images.

6.2 Future Work

The following are directions for future extensions to this work:

1. Our various heuristics (especially their threshold values) are sensitive to various characteristics of the logos and the documents. Rather than setting their values by trial-and-error as we have currently done, they can and should be studied as data-driven parameter search problems. Hyperparameter search techniques used in machine learning should be used to and the best values for the thresholds TM , TN , TL, TU , TA, and TB, assuming availability of sufficient training examples.
2. The extension to multiple logos on a single page is not conceptually difficult in the case where we have multiple matches in which we have good confidence and whose homography matrices place the logos in non-overlapping areas of the page. The efficacy of this approach needs to be validated empirically.
3. The current solution is not closed-loop, in the following sense: if a page contains a previously unseen logo (i.e., one currently not existing in the logo bank), this can result either in a false positive or in no match, but there is no way to distinguish these outcomes and use it to provide (possibly human) feedback that would allow the system to bootstrap itself. The problem of incremental corpus enlargement is not unique to this use case.
4. We can consider using information from color channels, where available, to improve accuracy. The additional information could be useful for

creating more discriminating descriptors, and could further mitigate the problem such as incorrect borders and occlusions by seals [4].

5. Seal detection, which has to address a very different set of problems (see [4]), can provide additional information on entities such as government organizations, ports, and processing entities, which would all be useful in this application domain.
6. Given its high level of accuracy, our solution is well-suited for use as a test set generation and annotation scheme for various deep learning-based algorithms for Logo Detection [8] problems.

References

[1] ClarifAI Logo Model. https://www.clarifai.com/models/.Accessed on 2018-08-18.

[2] ImageMagick. https://github.com/ImageMagick/ImageMagick. Ac- cessed on 2018-08-18.

[3] Jupyter Notebook. http://jupyter.org/. Accessed on 2018-08-18.

[4] Alaei, A., Roy, P., and Pal, U. Logo and seal based administrative document image retrieval: A survey. Computer Science Review Vol. 22 (09 2016), 47–63.

[5] Alcantarilla, P. F., Bartoli, A., and Davison,A.J. Kaze features. In Proceedings of the 12th European Conference on Computer Vision - Volume Part
VI (Berlin, Heidelberg, 2012), ECCV'12, Springer-Verlag, pp. 214–227.

[6] Bay, H., Ess, A., Tuytelaars, T., and Van Gool,L. Speeded-up robust features (SURF). Comput. Vis. Image Underst. 110, 3 (June 2008), 346–359.

[7] Bradski, G. The opencv library. Dr. Dobb's Journal of Software Tools (2000).

[8] Hassaballah, M., Ali, A., and Alshazly, H. Image Features Detection, Description and Matching, vol. 630. Springer, Cham, 02 2016.

[9] Leutenegger, S., Chli, M., and Siegwart, R. Y. Brisk: Binary robust invariant scalable keypoints. In Proceedings of the 2011 International Conference on Computer Vision (Washington, DC, USA, 2011), ICCV '11, IEEE Computer Society, pp. 2548–2555.

[10] Lowe, D. G. Distinctive image features from scale invariant keypoints. Int. J. Compute. Vision 60, 2 (Nov. 2004), 91–110.

[11] Muja, M., and Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. In International Conference on Computer Vision Theory and Application VISSAPP'09) (2009), INSTICC Press, pp. 331–340.

Logo Detection and Recognition from Scanned Documents

Sushama Khanvilkar[1], Vinay Billa[2], Saikirankumar Dasari[3], Saqib Naikwadi[4] , Aakash Tiwari[5]

Computer Department, Xavier Institute of Engineering, Mumbai, India

[1]sushma.k@xavierengg.com
[2]billavinay2012@gmail.com
[3]saikiran.dasari2011@gmail.com
[4]sinxie007@gmail.com
[5]aakash2997@gmail.com

***Abstract*--We present a randomized and heuristic algorithm based on computer vision principles to detect and recognize logos in documents such as invoices and purchase orders. It is intended for use in enhancing document retrieval (DR) and named entity recognition (NER) tasks. Our algorithm operates in two phases: first, it detects the area in a page of a scanned document where a logo appears, and marks this area by using a bounding box; then, it recognizes the organizational entity associated with the logo by finding the best match for it against a pre-existing database of known logos. We extract distinctive image features from both logo and document using the Scale Invariant Feature Transform (SIFT) algorithm and use the Fast Library for Approximate Nearest Neighbors (FLANN) for matching against the database, with a number of custom heuristics to reduce detection and recognition error. The technique is robust to image transformations such as rotation and scaling and to noise from blurriness and scanner artifacts.**

***Keywords*— FLANN, SIFT, Document Retrieval(DR), Named entity recognition(NER)**

I. INTRODUCTION

Public organizations, institutes, companies and private sectors are generally interested in implementing digital mail rooms to improve the efficiency of paper-intensive workflows and to reduce the burden of manual processing of different administrative documents including incoming mails, faxes, forms, invoices, reports, employee records, health record, etc. Logo can be considered as an important and popular salient entities presented in administrative documents. The manual identification/verification of logos is not an easy task, as the documents in-flow in organizations is growing rapidly.

Indeed, accurate detection and recognition of logo in document images provide us with a more reliable and appropriate system. However, logo detection/recognition is a challenging task, as logos are generally composed of quite complex symbols, graphical and textual components.

The main aim of this project is to present the efficient and robust framework for detection as well as recognition of logo images.

A. Aim and Objective :

The aim is to provide efficient and robust framework for detection as well as recognition of logo images

.

The objectives are:

1) To present the literature review over different approaches presented over logo.
2) To present the analysis of different methods according to their detection accuracy and performances.
3) To present and discuss the proposed methods for logo detection and recognition.
4) To present the practical analysis of proposed work and its evaluation against the existing methods.

II. RELATEDWORK

The literature on Logo Detection and Recognition is extremely extensive. Financial transactions for exchange of goods or services between parties usually generate data in the form of physical documents such as invoices and purchase orders. Such documents are used to record, verify, and scrutinize transactions; they may either have a rigidly structured layout like a form, or be more unstructured in nature. Use cases such as domestic and international trade finance transactions typically involve additional third parties like banks and financial institutions as intermediaries who facilitate the exchange and act as trusted parties. While such parties earn significant revenue from such trade activities, they need to avoid invalid transactions, which may have legal sanctions and lead to significant losses.

To validate transactions, such intermediaries currently employ and invest in manual verification, which is painfully slow and inefficient. It is, therefore, of great interest to them to be able to leverage automated DR and NER. Techniques from such unstructured documents for use by authorized personnels or officials to validate and act on transactions. The work described here is a component of a larger platform-based solution to address this gap: to help visualize information, to improve decision making, and further minimize the processing time and increase productivity for this task.

Logos are an important element of documents, and detecting and recognizing them significantly augments the richness of the information being presented to the human operator. However, logos are not text, and are therefore not captured by traditional Optical Character Recognition (OCR) techniques. Specialized image-based techniques are therefore needed.

Various public cloud-based web services (such as Google Cloud Vision and ClarifAI [1]) support logo detection by exposing an API over REST-style frameworks. This saves considerable investment on model building activities (gathering training data, building neural network architectures, and fitting them to available data) at the cost of requiring documents to be uploaded to the public cloud. This is not considered an acceptable tradeoff for sensitive documents relating to financial transactions, where a data breach or information leak can result in significant losses or liability actions. This consideration ruled out for us the possibility of using pre-built models and transfer learning techniques, and led us to develop an on-premise robust algorithmic solution.

The problem of using logo information for document image processing basically involves two main tasks:

1) Finding boundary of a logo/ on a document image irrespective of its class.

2) Indexing/matching the detected logo/ candidate region to a database for classifying or for concluding that the region is not of interest. The former is referred to as logo/ detection/spotting, while the latter is called logo/ recognition.

III. DESIGN AND IMPLEMENTATION

The project aims at developing a Logo Detection and Recognition system from Scanned Documents which can be used in various organizations, institutes, etc. The technology is rapid efficient and robust.

For computing key-points, we use the OpenCV [7] implementation of SIFT. For key-point matching we use the OpenCV implementation of the Fast Library for Approximate Nearest Neighbour Search (FLANN) [11], which provides a set of algorithms to do fast nearest-neighbour search for high-dimensional features.

A. Feature Detection and Extraction:

We use principles of object detection and tracking to match common features between images. The unique patterns in an image or the points of interest which describe the image are known as key-points [8]. Detection of this key-points will be done using an algorithm which are invariant to scale and rotational transformation. We have compared SURF [6], SIFT, BRISK [9] and KAZE [5] algorithms and we found SIFT algorithm is best suited for all the above specifications. SIFT (Scale Invariant Feature Transform): SIFT uses following four-step process: Scale-space extrema are detected based on a Difference-of-Gaussians (DoG) filter, which approximates a Laplacian-of-Gaussians (LoG) operator. Key-points are localized by eliminating and rening low contrast key-points and edge key-points. The orientation of the key-point is assigned by calculating the scale, gradient magnitude, and direction in the region of the neighbourhood. The descriptor is calculated using 16 sub-blocks around the key-point neighbourhood, each divided into 4x4 size. For each sub-block, an eight-bin orientation histogram is created. This descriptor is represented as a 128-dimensional vector.

B. Feature Matching:

Once we have extracted features for both logos and pages, we need to find a “best match " between the features of a page and those of logos in the logo bank. We here need a brute force matching using hierarchical fast nearest-neighbour search. Lowe [10] uses a nearest-neighbor technique to compute the top two matches, and then uses the ratio of the closest distance to the second-closest distance to reject matches that exceed a threshold. This matching technique is low cost, fast, and effective when compared to the brute-force approach.

C. Homographies and Outlier Filtering:

After determining the matching features or key-points between the page and a logo, we locate the logo on the page by trying to fit a transformation with respect to the key-points. The transformation is denied by a homography matrix, a 33 matrix H giving the perspective transformation of the matched area with respect to the logo image. In case no homography matrix is found for a set of key-points, we conclude that no logo is present on the page. Although Lowe's ratio test removes most incorrect matches, there can still be outliers in the match. The Random Sample Consensus (RANSAC) estimator is a probabilistic method of removing outliers from matched features that can approximate the homography matrix H even with very few matched points. Finally, we compute the perspective transformation using the matched points and the homography matrix H. This returns a set of points which represents the pixel coordinates on the page of the bounding box of the logo.

D. *List of Heuristic Checks:*

The data challenges discussed earlier introduce errors which are not solved by techniques like homography estimation. While a homography matrix H can be computed between the key-points on the logo and the document, a meaningful match should conform to certain physical realities. We use the characteristics of a good match as our basis for eliminating false positives and increasing solution accuracy, and now list a comprehensive set of heuristic checks for this purpose. All of these checks are Boolean in nature.

1) *Matched Area Test:*

We do not expect the matched logo to occupy the majority of the space on the target page. The ratio of these areas must be below a threshold TM.

2) *Homography Matrix Validation Test:*

The homography matrix which defines a transformation can be computed mathematically but may not be valid as per traditional image transformation rules. One such way of validating a homography matrix is to compute its determinant. If the determinant value approaches zero, then the homography matrix is singular, which means we are seeing the plane object at 90 degrees, which is impossible. Likewise, the ratio of the matched area on the page to the matched logo area (given by the square of the determinant) should be a finite number and not exceed a threshold. Intuitively, the transformation can expand or shrink the logo in document, but not by a very large factor. Additionally, if the determinant of the homography matrix is negative, then the homography is not conserving the orientation, except for mirror images, which is outside the scope of this paper. SIFT and similar algorithms are not known to be mirror-image invariant.

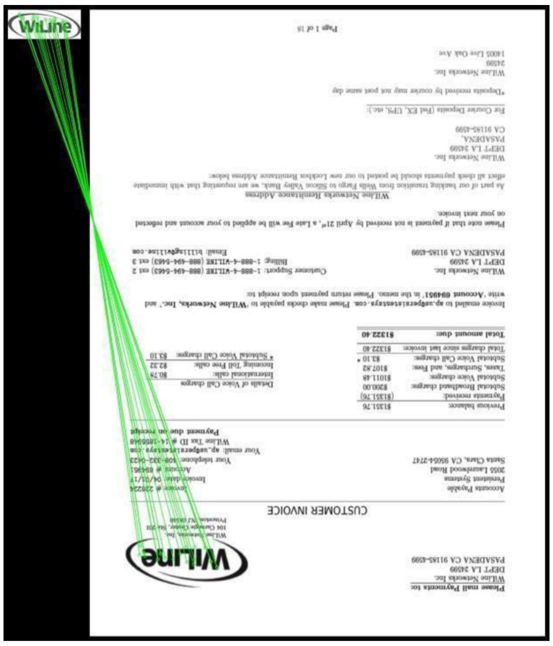


Fig. 1 Example match result on the same document scanned up-side down

3) *Aspect Ratio Test:*

The aspect ratio should ideally be preserved by the homography transformation, for a good match. The computed ratio of the aspect-ratios of the logo and its image on the page should not exceed a threshold TA. We allow the tolerance to take into consideration the changes caused by perspective transformation and introduced noise.

4) *Border Pixel Match Test:*

The key-point match is struck down if the matching key-points are at or within TB pixels of the border of the page. This is equivalent to shrinking the match area by adding margins.

E. *Putting It Together*:

We use open-source tools like ImageMagick [2] and associated Python packages for preprocessing document images and logos. All development is done in Python, using the Anaconda environment with Jupyter Notebooks [3]. The Open Source Computer Vision library (OpenCV) built with Python is used for data ingestion, feature detection, and various image processing tasks. We pre-compute and store the key-points of all logos in the logo bank, along with metadata such as the logo's height and width. Logos are resized for maintaining correct aspect ratios. An arbitrary logo downloaded from the internet or captured from documents may contain uninformative whitespaces, so that even a rectangular shaped logo would appear square. The logos are cropped to maintain the correct aspect ratio as needed for the match. If needed, logos are compressed as well for optimizing size. We then convert them or standardize them to gray scale formats from color (RGB) formats. We use the Mogrify program of the ImageMagick suite to resize and strip useless information and plane-interlace the color channel values.

Financial documents scanned and uploaded from various endpoint scanners, such as at supplier locations, vendor offices, and local branches of financial institutions are collected using existing Enterprise Resource Planning (ERP) systems. We first preprocess these PDF files using ImageMagick tools to convert them to an image format and to optimize for size if needed. For some cases we also apply a very small Gausian blur (radius = 0.05), as this enhances the detection of logo features, while making it less likely to detect them in textual content area. The page images are then converted into gray scale format from color (RGB) formats.

For computing key-points, we use the OpenCV implementation of SIFT. For key-point matching, we use the OpenCV implementation of the Fast Library for Approximate Nearest Neighbor Search (FLANN) [17], which provides a set of algorithms to do fast nearest-neighbor search for high-dimensional features. We run our heuristic checks for each logo A clearing Lowe's ratio test for a given page p, requiring unanimous consensus from all of the heuristics in order for to A to proceed further. Should multiple logos still survive, we sort them by decreasing order of their key-point match confidence, using the value of abs(det(H)) to break any remaining ties.

IV. RESULTS

We now discuss the performance of our solution on a set of multi-page PDF documents like invoices and purchase orders containing distinct logos. Of these, logos were graphical, textual,
and mixed. No page contained more than a single logo. We selected pages from the documents, each containing either zero or one logo from our logo bank. The rise in accuracies shows the critical role of the heuristics in the solution. As shown in Figure 1, the matches are rotational invariant and independent of the degree of rotation.

We emphasize that these results are necessarily specific to the data bank and the set of pages, and extrapolation to other scenarios should be performed only with great caution.

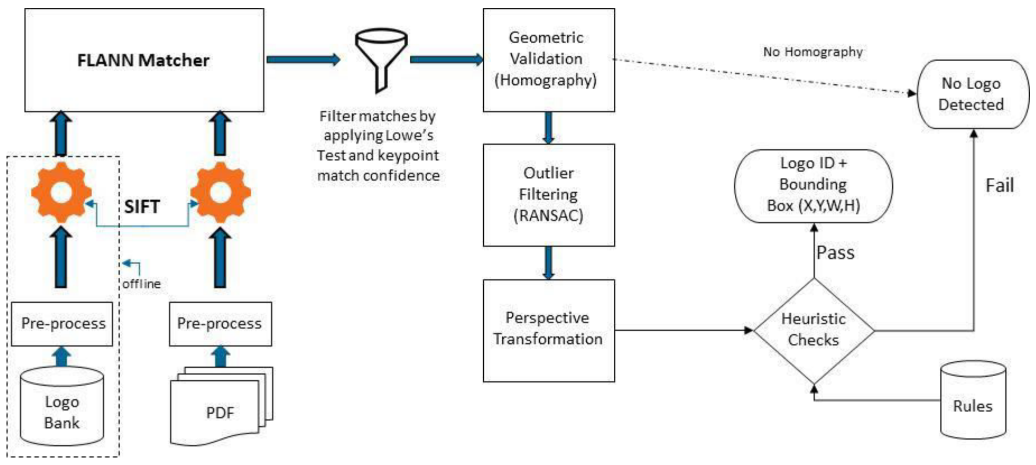
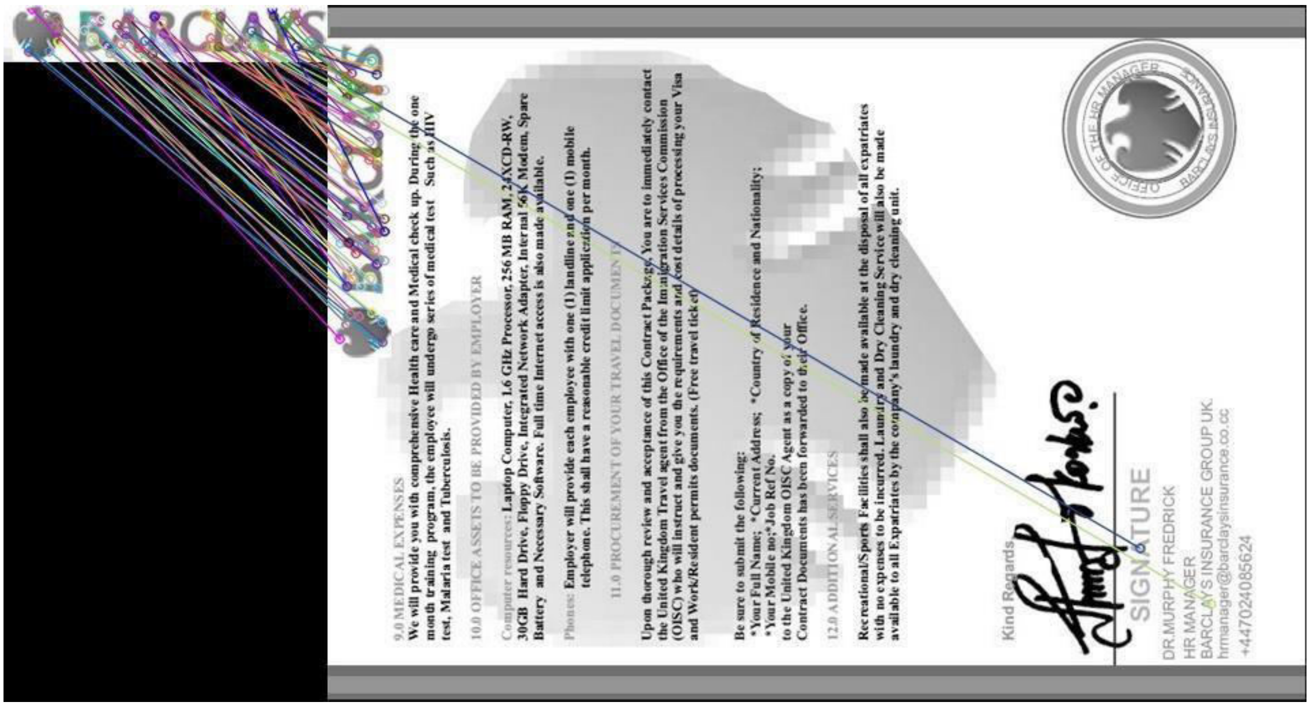

FLANN Matcher
Filter matches by applying Lowe's Test and keypoint match confidence
Geometric Validation (Homography)
No Homography
No Logo Detected
SIFT
offline
Outlier Filtering (RANSAC)
Logo ID + Bounding Box (X,Y,W,H)
Fail
Pass
Pre-process
Pre-process
Perspective Transformation
Heuristic Checks
Logo Bank
PDF
Rules

Fig. 2 Solution process flow diagram.

The model is fed with the input and the logo is retrieved. Fig. 3 shows the output for a given document.



9.0 MEDICAL EXPENSES
We will provide you with comprehensive Health care and Medical check up. During the one month training program, the employee will undergo series of medical test Such as HIV test, Malaria test and Tuberculosis.

10.0 OFFICE ASSETS TO BE PROVIDED BY EMPLOYER

Computer resources: Laptop Computer, 1.6 GHz Processor, 256 MB RAM, 24XCD-RW, 30GB Hard Drive, Floppy Drive, Integrated Network Adapter, Internal 56K Modem, Spare Battery and Necessary Software. Full time Internet access is also made available.

Phones: Employer will provide each employee with one (1) landline and one (1) mobile telephone. This shall have a reasonable credit limit application per month.

11.0 PROCUREMENT OF YOUR TRAVEL DOCUMENTS

Upon thorough review and acceptance of this Contract Package, You are to immediately contact the United Kingdom Travel agent from the Office of the Immigration Services Commission (OISC) who will instruct and give you the requirements and cost details of processing your Visa and Work/Resident permits documents. (Free travel ticket)

Be sure to submit the following:
*Your Full Name; *Current Address; *Country of Residence and Nationality;
*Your Mobile no;*Job Ref No.
to the United Kingdom OISC Agent as a copy of your Contract Documents has been forwarded to their Office.

12.0 ADDITIONAL SERVICES

Recreational/Sports Facilities shall also be made available at the disposal of all expatriates with no expenses to be incurred. Laundry and Dry Cleaning Service will also be made available to all Expatriates by the company's laundry and dry cleaning unit.

Kind Regards

[illegible]

SIGNATURE
DR.MURPHY FREDRICK
HR MANAGER
BARCLAYS INSURANCE GROUP UK.
hrmanager@barclaysinsurance.co.cc
+447024085624

OFFICE OF THE HR MANAGER
BARCLAYS INSURANCE

Fig. 3 Logo detection

V. CONCLUSION

We have eliminated various feature detection techniques such as SURF [6] (Speeded Up Robust Features) and BRIEF (Binary Robust Independent Elementary Features) as we found SIFT to be well suited for both scale and rotationally-invariant changes of logos from the scanned documents. We have studied and evaluated various feature detection and description methods coupled with corresponding matching algorithms for extracting and recognizing logos in financial documents such as invoices, bills of lading, and purchase orders. The combination of SIFT for key-point detection and extraction and FLANN for key-point matching gave us the best results. The use of additional heuristics vastly improves algorithm accuracy and minimizes false positives. Our solution is currently being developed to augment various use cases such as DR and NER.

VI. FUTURE WORK

A. Our various heuristics (especially their threshold values) are sensitive to various characteristics of the logos and the documents. Rather than setting their values by trial-and-error as we have currently done, they can and should be studied as data-driven parameter search problems. Hyperparameter search techniques used in machine learning should be used to find the best values for the thresholds T_M, T_N , T_L, T_U , T_A, and T_B , assuming availability of sufficient training examples.

B. The extension to multiple logos on a single page is not conceptually difficult in the case where we have multiple matches in which we have good confidence and whose homography matrices place the logos in non- overlapping areas of the page. The efficacy of this approach needs to be validated empirically.

C. The current solution is not closed-loop, in the following sense: if a page contains a previously unseen logo (i.e., one currently not existing in the logo bank), this can result either in a false positive or in no match, but there is no way to distinguish these outcomes and use it to provide (possibly human) feedback that would allow the system to bootstrap itself. The problem of incremental corpus enlargement is not unique to this usecase.

D. We can consider using information from color channels, where available, to improve accuracy. The additional information could be useful for creating more discriminating descriptors, and could further mitigate the problem such as incorrect borders and occlusions by seals [4].

E. Seal detection, which has to address a very different set of problems (see [4]), can provide additional information on entities such as government organizations, ports, and processing entities, which would all be useful in this application domain.

F. Given its high level of accuracy, our solution is well- suited for use as a test set generation and annotation scheme for various deep learning-based algorithms for Logo Detection problems.

ACKNOWLEDGMENT

We express sincerest gratitude to our project guide, Head of Department Sushama Khanvilkar, who has been a source of inspiration and has made it possible for us to initiate and carry on with this project.

REFERENCES

[1] ClarifAI Logo Model. https://www.clarifai.com/models/.Accessed on 2018-08-18.
[2] ImageMagick. https://github.com/ImageMagick/ImageMagick. Ac- cessed on 2018-08-18.
[3] Jupyter Notebook. http://jupyter.org/. Accessed on 2018-08-18.
[4] Alaei, A., Roy, P., and Pal, U. Logo and seal based administrative document image retrieval: A survey. Computer Science Review Vol. 22 (09 2016), 47–63.
[5] Alcantarilla, P. F., Bartoli, A., and Davison,A.J. Kaze features. In Proceedings of the 12th European Conference on Computer Vision - Volume Part VI (Berlin, Heidelberg, 2012), ECCV'12, Springer-Verlag, pp. 214–227.
[6] Bay, H., Ess, A., Tuytelaars, T., and Van Gool,L. Speeded-up robust features (SURF). Comput. Vis. Image Underst. 110, 3 (June 2008), 346–359.
[7] Bradski, G. The opencv library. Dr. Dobb's Journal of Software Tools (2000).
[8] Hassaballah, M., Ali, A., and Alshazly, H. Image Features Detection, Description and Matching, vol. 630. Springer, Cham, 02 2016.
[9] Leutenegger, S., Chli, M., and Siegwart, R. Y. Brisk: Binary robust invariant scalable keypoints. In Proceedings of the 2011 International Conference on Computer Vision (Washington, DC, USA, 2011), ICCV '11, IEEE Computer Society, pp. 2548–2555.
[10] Lowe, D. G. Distinctive image features from scale invariant keypoints. Int. J. Compute. Vision 60, 2 (Nov. 2004), 91–110.
[11] Muja, M., and Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. In International Conference on Computer Vision Theory and Application VISSAPP'09) (2009), INSTICC Press, pp. 331–340.

Acknowledgement

It gives immense pleasure in bringing out this synopsis of the project entitled "Logo Detection and Recognition from Scanned Images". Firstly, we would like to thank our guide "Prof. Sushama Khanvilkar" who gave us his valuable suggestions and ideas when we were in need of them. He encouraged us to work on this project.

We are also grateful to our college for giving us the opportunity to work with them and providing us the necessary resources for the project. Working on these project also helped us to do lots of research and also we came to know about so many new things.

We are immensely grateful to all involved in this project as without their inspiration and valuable suggestion it would not have been possible to develop the project within the prescribed time.

VINAY BILLA (08)

SAIKIRANKUMAR
DASARI (15)

SAQIB NAIKWADI (55)

AAKASH TIWARI (89)