R .V. COLLEGE OF ENGINEERING, BENGALURU- 560 059
(Autonomous Institution Affiliated to VTU, Belagavi)

TITLE: ENHANCING THE EFFICIENCY OF OCR FOR KANNADA

Submitted by

Adamyaa D. N.	**(1RV18IS002)**
Ananya G. M.	**(1RV18IS006)**
Nehal N. Shet	**(1RV18IS026)**
Sushrut M.	**(1RV18IS054)**
Varshini P.	**(1RV18IS058)**

Submitted to,

Dr. Rajashekara Murthy S.

Associate Professor,

DTL Course Coordinator,

Department of Information Science,

R V College of Engineering, Bengaluru-59

CERTIFICATE

Certified that the Assignment topic "ENHANCING THE EFFICIENCY OF OPTICAL CHARACTER RECOGNITION FOR KANNADA" is carried out by Adamyaa D. N. (1RV18IS002), Ananya G. M. (1RV18IS006), Nehal N Shet (1RV18IS026), Sushrut M (1RV18IS054) and Varshini P. (1RV18IS058) who are bonafide students of R V College of Engineering, Bengaluru in partial fulfillment of the award of assignment marks for the 4th semester academic year 2019-20 in Design Thinking Lab Course. It is certified that all corrections/ suggestions indicated for the internal assessment have been incorporated in the report, and a soft copy is deposited in the department library. The assignment report has been approved as it satisfies the academic requirement in respect of the work prescribed by the institution for the said course.

Marks awarded:

COs	**CO1**	**CO2**	**CO3**	**CO4**	**Total**
Max. Marks	**03**	**04**	**01**	**02**	**10**
Marks Obtained					

CONTENTS

ABSTRACT

In this report, we have discussed how to improve the efficiency of Tesseract OCR for Kannada. Kannada has roughly 44 million native speakers. Kannada is also spoken as a second and third language by over 12.9 million non-Kannada speakers in Karnataka, which adds up to 56 million speakers and writes Kannada script. Research in Optical Character Recognition (OCR) is popular for its application potential in banks, post offices, defense organizations and library automation etc. In this report, we have proposed a technique for improving the efficiency for Tesseract OCR.

INTRODUCTION

Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast).

Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Why Kannada OCR?

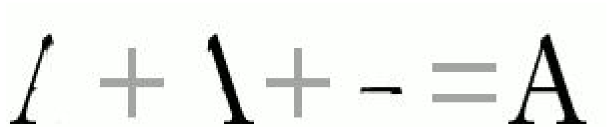
With the main aim of serving our society and creating and implementing something new that would be of some help to people, we chose to work for the literature domain of the society. One of the most common problems in the field of literature in India is transforming printed text into digitized format. We wanted to digitize Kannada script literary scripts as was the objective of Kannada Ganaka Parishat. With the ongoing Digital India Campaign, we realized this could contribute to the campaign. Hence, we decided to work on Optical Character Recognition (OCR). OCR can make your life easier by:

• Making paper-based information searchable in seconds, rather than hours.

• Reduce or eliminate costly data entry by automatically grabbing information you need from paper and putting it where it needs to go.

• Enabling entirely new ways to process documents that can eliminate "human touches", thereby reducing costs and dramatically reducing processing times.

As early as the 1960's, engineers were trying to develop a way for machines to recognize text. Unlike humans with visual capability, computers don't have eyes, nor can they differentiate between font types to be able to form a character or word. There was an early capability to read a singular font type called OCR-A, but there was no way to assure that everyone used this one typeface, so its usefulness was limited. After a spell back at the proverbial drawing board, a new way of performing OCR was developed. This new method relied on pattern recognition,whereby the computer didn't have to recognize the whole letter "R" to know it was an "R." Instead the computer would look for common points, patterns and combinations of lines and shapes to determine which letter it was reading. This image, found on the website "explainthatstuff.com" demonstrates this phenomenon.



Utilizing points, patterns and lines helped speed the OCR process and enabled the flexibility to read hundreds of fonts. This technology has expanded to handwriting as well, though it works best on structured forms rather than free form handwriting. Over the last 15 years, OCR technology has gotten faster and more accurate. Modern OCR software will have multiple language packs and the ability to ready scientific symbols and other less common font types.

LITERATURE SURVEY

1. History of machine recognition of scripts

The overwhelming volume of paper-based data in corporations and offices challenges their ability to manage documents and records. Computers, working faster and more efficiently than human operators, can be used to perform many of the tasks required for efficient document and content management. Computers understand alphanumeric characters as ASCII code typed on a keyboard where each character or letter represents a recognizable code. However, computers cannot distinguish characters and words from scanned images of paper documents. Therefore, where alphanumeric information must be retrieved from scanned images such as commercial or government documents, tax returns, passport applications and credit card applications, characters must first be converted to their ASCII equivalents before they can be recognized as readable text. Optical character recognition system (OCR) allows us to convert a document into electronic text, which we can edit and search etc. It is performed off-line after the writing or printing has been completed, as opposed to on-line recognition where the computer recognizes the characters as they are written. For these systems to effectively recognize hand-printed or machine-printed forms, individual characters must be well separated. This is the reason why most typical administrative forms require people to enter data into neatly spaced boxes and force spaces between letters entered on a form. Without the use of these boxes, conventional technologies reject fields if people do not follow the structure when filling out forms, resulting in a significant overhead in the administration cost.

Optical character recognition for English has become one of the most successful applications of technology in pattern recognition and artificial intelligence. OCR is the machine replication of human reading and has been the subject of intensive research for more than five decades. To understand the evolution of OCR systems from their challenges, and to appreciate the present state of the OCRs, a brief historical survey of OCRs is in order now.

Depending on the versatility, robustness and efficiency, commercial OCR systems may be divided into the following four generations. It is to be noted that this categorization refers specifically to OCRs of English language.

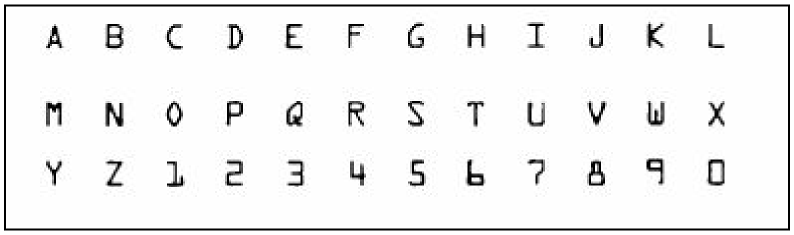
1

First generation OCR systems

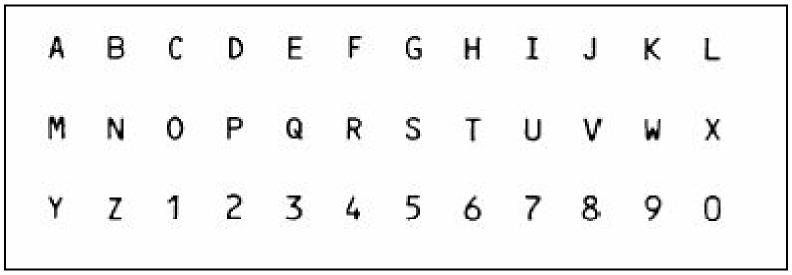
Character recognition originated as early as 1870 when Carey invented the retina scanner, which is an image transmission system using photocells. It is used as an aid to the visually handicapped by the Russian scientist Tyurin in 1900. However, the first generation machines appeared in the beginning of the 1960s with the development of the digital computers. It is the first time OCR was realized as a data processing application to the business world [Mantas, 1986]. The first generation machines are characterized by the "constrained" letter shapes which the OCRs can read. These symbols were specially designed for machine reading, and they did not even look natural. The first commercialized OCR of this generation was IBM 1418, which was designed to read a special IBM font, 407. The recognition method was template matching, which compares the character image with a library of prototype images for each character of each font.

Second generation OCR system

Next generation machines were able to recognize regular machine-printed and hand- printed characters. The character set was limited to numerals and a few letters and symbols. Such machines appeared in the middle of 1960s to early 1970s. The first automatic letter- sorting machine for postal code numbers from Toshiba was developed during this period. The methods were based on the structural analysis approach. Significant efforts for standardization were also made in this period. An American standard OCR character set: OCR-A font was defined, which was designed to facilitate optical recognition, although still readable to humans. A European font OCR-B was also designed.



OCR-A font



A	B	C	D	E	F	G	H	I	J	K	L
M	N	O	P	Q	R	S	T	U	V	W	X
Y	Z	1	2	3	4	5	6	7	8	9	0

OCR-B font

Third generation OCR system

For the third generation of OCR systems, the challenges were documents of poor quality and large printed and hand-written character sets. Low cost and high performance were also important objectives. Commercial OCR systems with such capabilities appeared during the decade 1975 to 1985.

OCR's Today (Fourth generation OCR system)

The fourth generation can be characterized by the OCR of complex documents intermixing with text, graphics, tables and mathematical symbols, unconstrained hand- written characters, color documents, low-quality noisy documents, etc. Among the commercial products, postal address readers, and reading aids for the blind are available in the market.

Nowadays, there is much motivation to provide computerized document analysis systems. OCR contributes to this progress by providing techniques to convert large volumes of data automatically. A large number of papers and patents advertise recognition rates as high as 99.99%; this gives the impression that automation problems seem to have been solved. Although OCR is widely used presently, its accuracy today is still far from that of a seven-year old child, let alone a moderately skilled typist. Failure of some real applications show that performance problems still exist on composite and degraded documents (i.e., noisy characters, tilt, mixing of fonts, etc.) and that there is still room for progress.

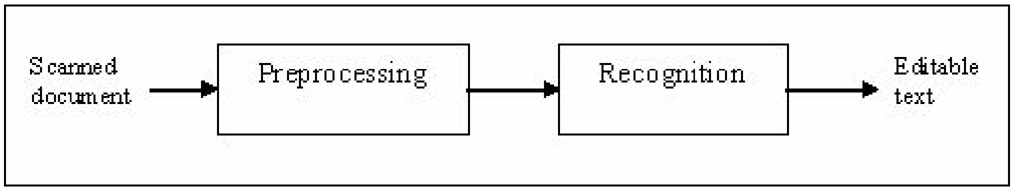
Various methods have been proposed to increase the accuracy of optical character recognizers. In fact, at various research laboratories, the challenge is to develop robust methods that remove as much as possible the typographical and noise restrictions while maintaining rates similar to those provided by limited-font commercial machines.

Thus, current active research areas in OCR include handwriting recognition, and
also the printed typewritten version of non-Roman scripts (especially those with a very large number of characters).

2. Components of a OCR system

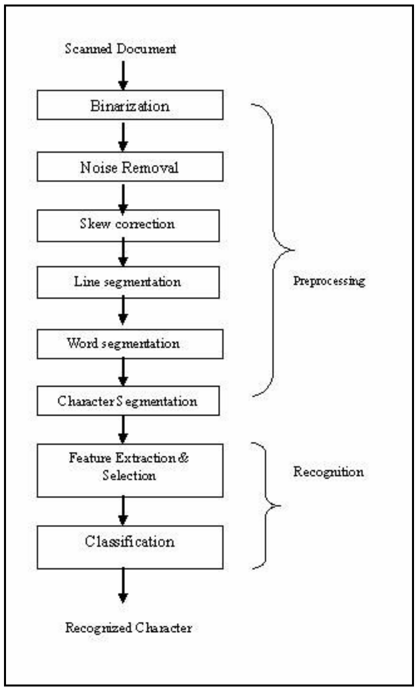
Before we present a survey on various approaches used in the literature for recognizing fonts and characters, a brief introduction to the general OCR techniques is given now.

The objective of OCR software is to recognize the text and then convert it to editable form. Thus, developing computer algorithms to identify the characters in the text is the principal task of OCR. A document is first scanned by an optical scanner, which produces an image form of it that is not editable. Optical character recognition involves translation of this text image into editable character codes such as ASCII. Any OCR implementation consists of a number of preprocessing steps followed by the actual recognition, as shown in Figure 2.3.



OCR process

The number and types of preprocessing algorithms employed on the scanned image depend on many factors such as age of the document, paper quality, resolution of the scanned image, amount of skew in the image, format and layout of the images and text, kind of the script used and also on the type of characters: printed or handwritten. After preprocessing, the recognition stage identifies individual characters, and converts them into editable text. The image below depicts these steps and they are described in the following section.



Steps in an OCR

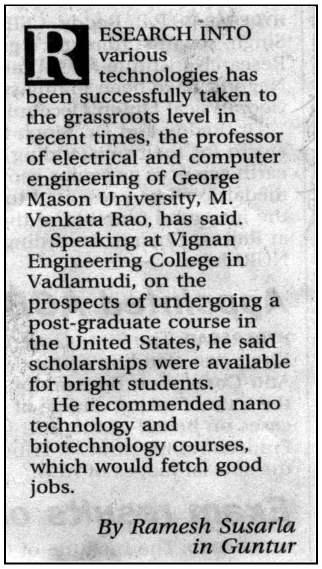
Preprocessing

Typical preprocessing includes the following stages:

1. Binarization
2. Noise removing
3. Skew detection and correction
4. Line segmentation
5. Word segmentation
6. Character segmentation
7. Thinning

I. Binarization

A printed document is first scanned and is converted into a gray scale image. Binarization is a technique by which the gray scale images are converted to binary images. In any image analysis or enhancement problem, it is very essential to identify the objects of interest from the rest. In a document image this usually involves separating the pixels forming the printed text or diagrams (foreground) from the pixels representing the blank paper (background). The goal is to remove only the background, by setting it to white, and leave the foreground image unchanged. Thus, binarization separates the foreground and background information. This separation of text from background is a prerequisite to subsequent operations such as segmentation and labeling [Bunke & Wang, 1997]. Figure 2.5 shows a gray image (a), and binary image (b) of a newspaper document.

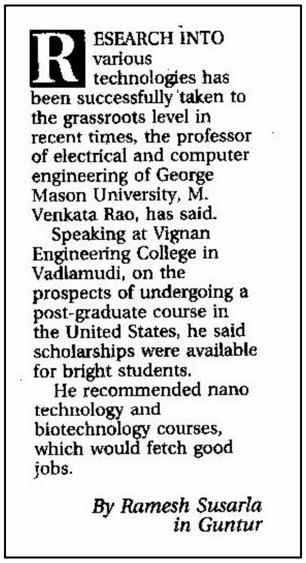


RESEARCH INTO various technologies has been successfully taken to the grassroots level in recent times, the professor of electrical and computer engineering of George Mason University, M. Venkata Rao, has said.

Speaking at Vignan Engineering College in Vadlamudi, on the prospects of undergoing a post-graduate course in the United States, he said scholarships were available for bright students.

He recommended nano technology and biotechnology courses, which would fetch good jobs.

By Ramesh Susarla in Guntur



RESEARCH INTO various technologies has been successfully taken to the grassroots level in recent times, the professor of electrical and computer engineering of George Mason University, M. Venkata Rao, has said.

Speaking at Vignan Engineering College in Vadlamudi, on the prospects of undergoing a post-graduate course in the United States, he said scholarships were available for bright students.

He recommended nano technology and biotechnology courses, which would fetch good jobs.

By Ramesh Susarla in Guntur

a) Gray image (b) Binary image

Document image binarization

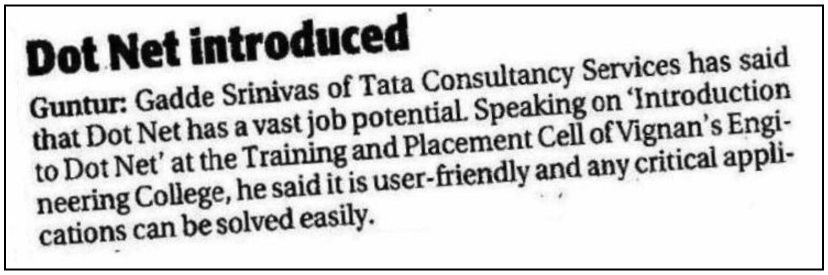
The most common method for binarization is to select a proper intensity threshold for the image and then convert all the intensity values above the threshold to one intensity value, and to convert all intensity values below the threshold to the other chosen intensity. Thresholding (Binarization) methods can be classified into two categories: global and adaptive thresholding. Global methods apply one threshold value to the entire image. Local or adaptive thresholding methods apply different threshold values to different regions of the image. The threshold value is determined by the neighborhood of the pixel to which the thresholding is being applied. Among those global techniques, Otsu's thresholding technique has been cited as an efficient and frequently used technique. Niblack's method and Sauvola's method are among the most well known approaches for adaptive thresholding.

II. Noise removal

Scanned documents often contain noise that arises due to printer, scanner, print quality, age of the document, etc. Therefore, it is necessary to filter this noise before we process the image. Commonly used approach is to process the image through a low-pass filter and use it for later processing. The objective in the design of a noise- filter is that it should remove as much of the noise as possible while retaining the entire signal.

III. Skew detection and correction

When a document is fed to the scanner either mechanically or by a human operator, a few degrees of tilt (skew) is unavoidable. Skew angle is the angle that the lines of text in the digital image make with the horizontal direction. Figure 2.6 shows an image with skew.



Dot Net introduced

Guntur: Gadde Srinivas of Tata Consultancy Services has said that Dot Net has a vast job potential. Speaking on 'Introduction to Dot Net' at the Training and Placement Cell of Vignan's Engineering College, he said it is user-friendly and any critical applications can be solved easily.

An image with skew

A number of methods have previously been proposed in the literature for identifying document image skew angles. Mainly, they can be categorized into the following groups: (i) methods based on projection profile analysis, (ii) methods based on nearest- neighbor clustering, (iii) methods based on Hough transform, (iv) methods based on cross-correlation, and (v) methods based on morphological transforms.

Most existing approaches use the Hough transform or enhanced versions. Hough transform detects straight lines in an image. The algorithm transforms each of the edge pixels in an image space into a curve in a parametric space. The peak in the Hough space represents the dominant line and it's skew. The major drawback of this method is that it is computationally expensive and is difficult to choose a peak in the Hough space when text becomes sparse.

Approaches based on the correlation use cross correlation between the text lines at a fixed distance. It is based on the observation that the correlation between vertical lines in an image is maximized for a skewed document; in general if one line is shifted relatively to the other lines such that the character baseline levels for two lines are coincident. The algorithm can be seen as a special case of the Hough transform. It is accurate for skew angles less than ±10 degrees.

Postl proposed a method in which the horizontal projection profile is calculated at a number of different angles. A projection profile is a one-dimensional array with a number of locations equal to the number of rows in an image. Each location in the projection profile stores a count of the number of black pixels in the corresponding row of the image. This histogram has the maximum amplitude and frequency when the text in the image is skewed at zero degrees since the number of co-linear black pixels is maximized in this condition. To determine the skew angle of a document, the projection profile is computed at a number of angles, and for each angle, the difference between peak and trough heights is measured. The maximum difference corresponds to the best alignment with the text line direction. This, in turn, determines the skew angle. The image is then rotated according to the detected skew angle.

Nearest neighbor methods described by Hoashizume et al. [Hoashizume et al., 1986] finds nearest neighbors of all connected components, the direction vector for all nearest neighbor pairs are accumulated in a histogram and the histogram peak is found to obtain a skew angle. Since only one nearest neighbor connectivity is made for each component, connection with noisy sub parts of characters reduces the accuracy of the method.

Approaches using morphological operations remove ascenders and descenders and merge adjacent characters to produce one connected component for each line of text. A line is then fit to the pixels in each component using a least-squares technique. Histogram of the angles of the lines detected is constructed in a document and a search procedure is applied to the histogram to determine the skew of the document.

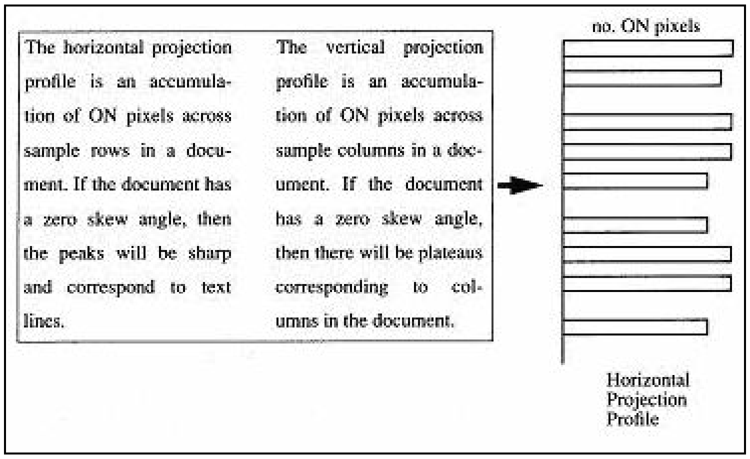
In the methods based on Fourier Transform, the direction for which the density of the Fourier space is the largest, gives the skew angle. Fourier method is computationally expensive for large images.

IV. Line, word and character segmentation

Once the document image is binarized and skew corrected, actual text content must be extracted. Commonly used segmentation algorithms in document image analysis are: Connected component labeling, X-Y tree decomposition, Run-length smearing, and Hough transform.

In connected component labeling, each connected component in the binary document image is assigned a distinct label. A connected component algorithm scans an image and groups its pixels into components based on pixel connectivity, i.e. all pixels in the connected component share similar pixel intensity values and are in some way connected to each other. Once all groups have been determined, each pixel is labeled according to the component (group) it was assigned to. This technique assigns to each connected component of a binary image a distinct label. The labels are natural numbers starting from 1 to the total number of connected components in the image.

The X-Y tree decomposition uses the Horizontal projection profile of the document image to extract the lines from the document. If the lines are well separated, the horizontal projection will have separated peaks and valleys, which serve as the separators of the text lines. These valleys are easily detected and used to determine the location of boundaries between lines. Similarly, gaps in the vertical projection of a line image are used in extracting words in a line, as well as extracting individual characters from the word. Overlapping, adjacent characters in a word (called kerned characters) cannot be segmented using zero-valued valleys of the vertical projection profile. Special techniques/heuristics have to be employed to solve this problem.


The horizontal projection profile is an accumulation of ON pixels across sample rows in a document. If the document has a zero skew angle, then the peaks will be sharp and correspond to text lines.
The vertical projection profile is an accumulation of ON pixels across sample columns in a document. If the document has a zero skew angle, then there will be plateaus corresponding to columns in the document.
no. ON pixels
Horizontal Projection Profile

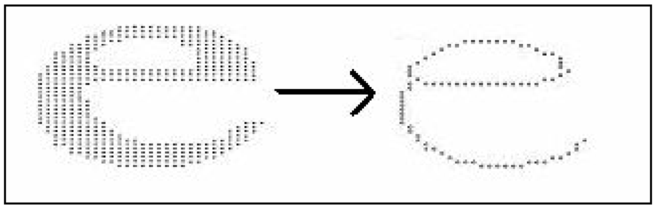
An image and its horizontal projection profile

Run-length smearing algorithm (RLSA)first detects all white runs (sequence of 1’s) of the line. It then converts those runs whose length is shorter than a predefined threshold T, to black runs. To obtain segmentation, RLSA is applied line-by-line first; and then column-by-column, yielding two distinct bitmaps; these are then combined by a logical AND operation.

Hough-transform based methods, described in earlier sections, doesn't require connected or even nearby edge points. They work successfully even when different objects are connected to each other.

V. Thinning

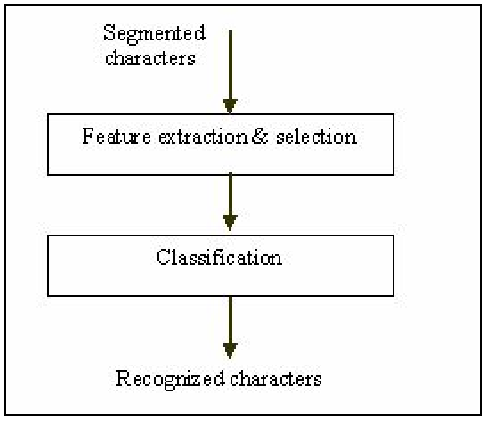
Thinning, or, skeletonization is a process by which a one-pixel-width representation (or the skeleton) of an object is obtained, by preserving the connectedness of the object and its end points. The purpose of thinning is to reduce the image components to their essential information so that further analysis and recognition are facilitated. For instance, an alphabet can be handwritten with different pens giving different stroke thicknesses, but the information presented is the same. This enables easier subsequent detection of pertinent features. Letter 'e' is shown in Figure 2.8 before and after thinning. A number of thinning algorithms have been proposed and are being used. The most common algorithm used is the classical Hilditch algorithm and its variants. For recognizing large graphical objects with filled regions which are often found in logos, region boundary detection is useful, but for small regions such as those which correspond to individual characters, neither thinning nor boundary detection is performed, and the entire pixel array representing the region is forwarded to the subsequent stage of analysis.



An image before and after thinning

VI. Recognition

By character recognition, the character symbols of a language are transformed into symbolic representations such as ASCII, or Unicode. The basic problem is to assign the digitized character into its symbolic class. This is done in two steps: (i) Feature extraction, and selection, and (ii) classification; and is shown below.



Recognition process

VII. Feature extraction and selection

The heart of any optical character recognition system is the formation of feature vectors to be used in the recognition stage. Feature extraction can be considered as finding a set of parameters (features) that define the shape of the underlying character as precisely and uniquely as possible. The term feature selection refers to algorithms that select the best subset of the input feature set. Methods that create new features based on transformations, or combinations of original features are called feature extraction algorithms. However, the terms feature selection and feature extractions are used interchangeably in literature. The features are to be selected in such a way that they help in discriminating between characters. Selection of feature extraction methods is probably the single most important factor in achieving high performance in recognition. A large number of feature extraction methods are reported in literature; but the methods selected depend on the given application. There is no universally accepted set of feature vectors in document image understanding. Features that capture topological and geometrical shape information are the most desired ones. Features that capture the spatial distribution of the black (text) pixels are also very important. Hence, two types of approaches are defined as follows:

1. Structural approach

In structural approaches features that describe the geometric and topological structures of a symbol are extracted. Structural features may be defined in terms of character strokes, character holes, end points, intersections between lines, loops and other character attributes such as concavities. Compared to other techniques, structural analysis gives features with high tolerance to noise and style variations. However, these features are only moderately tolerant to rotation and translation. Structural approaches utilize structural features and decision rules to classify characters. The classifier is expected to recognize the natural variants of a character but discriminate between similar looking characters such as ‘O’ and ‘Q’, ‘c’ and ‘e’, etc.

2. Statistical approach

In statistical approach, a pattern is represented as a vector: an ordered, fixed length list of numeric features. Many samples of a pattern are used for collecting statistics. This phase is known as the training phase. The objective is to expose the system to natural variants of a character. Recognition process uses these statistics for identifying an unknown character. Features derived from the statistical distribution of points include number of holes, geometrical moments, black-to-white crossing counts, width, height, and aspect ratio. Representation of a character image by statistical distribution of points takes care of style variations to a large extent.

Template matching is also one of the most common and oldest classification methods. In template matching, individual image pixels are used as features. Classification is performed by comparing an input character image with a set of templates (or prototypes) from each character class. The template, which matches most closely with the unknown, provides recognition. The technique is simple and easy to implement and has been used in many commercial OCR machines. However, it is sensitive to noise and style variations and has no way of handling rotated characters.

Statistical approach and structural approach both have their advantages and disadvantages. Statistical features are more tolerant to noise than structural descriptions provided the sample space over which training has been performed is representative and realistic. The variation due to font or writing style can be more easily abstracted in structural descriptions. In hybrid approach, these two approaches are combined at appropriate stages for representation of characters and utilizing them for classification of unknown characters. Segmented character images are first analyzed to detect structural features such as straight lines, curves, and significant points along the curves.

For regions corresponding to individual characters or graphical symbols, local features such as aspect ratio, compactness (ratio of area to square of perimeter), asymmetry, black pixel density, contour smoothness, number of loops, number of line crossings and line end points etc. are used.

Feature extraction techniques are evaluated based on: (a) robustness against noise, distortion, size, and style/font variation, (b) Speed of recognition, c) complexity of implementation, and d) how independent the feature set is without requiring any supplementary techniques.

3. Classification

Classification stage in an OCR process assigns labels to character images based on the features extracted and the relationships among the features. In simple terms, it is this part of the OCR which finally recognizes individual characters and outputs them in machine editable form. A number of approaches are possible for the design of a classifier; and the choice often depends on which classifier is available, or best known to the designer. Decision-theoretic methods are used when the description of the character can be numerically represented in a feature vector. The principal approaches to decision-theoretic recognition are: minimum-distance classifiers, statistical classifiers and neural networks. Jain, Duin & Mao identified the following two main categories.

The simplest approach is based on matching / identification of a similar neighbor pixel with the nearest distance. ‘Matching’ covers the groups of techniques based on similarity measures where the distance between the feature vector describing the extracted character and the description of each class is calculated. Different measures may be used, but the common is the Euclidean distance. This minimum distance classifier works well when the classes are well separated, that is when the distance between the mean values is sufficiently large compared to the spread of each class.

When the entire character is used as input to the classification, and no features are extracted (template-matching), a correlation approach is used. Here the distance between the character image and prototype images representing each character class is computed.

A special type of classifier is Decision tree which is trained by an iterative selection of individual features that are most salient at each individual node of the tree.

Another category is to construct decision boundaries directly by optimizing certain error criteria. Examples are: Fisher's linear discriminant that minimizes the Mean Squared Error (MSE) between classifier output and the desired labels.

Neural networks are another category. Considering a back-propagation neural network, the network is composed of several layers of interconnected elements. A feature vector enters the network at the input layer. Each element of the layer computes a weighted sum of its input and transforms it into an output by a nonlinear function. During training, the weights at each connection are adjusted until a desired output is obtained. A problem of neural networks in OCR may be their limited predictability and generality, while an advantage is their adaptive nature. Other approaches include Single layer and Multilayer Perceptrons, and Support vectors.

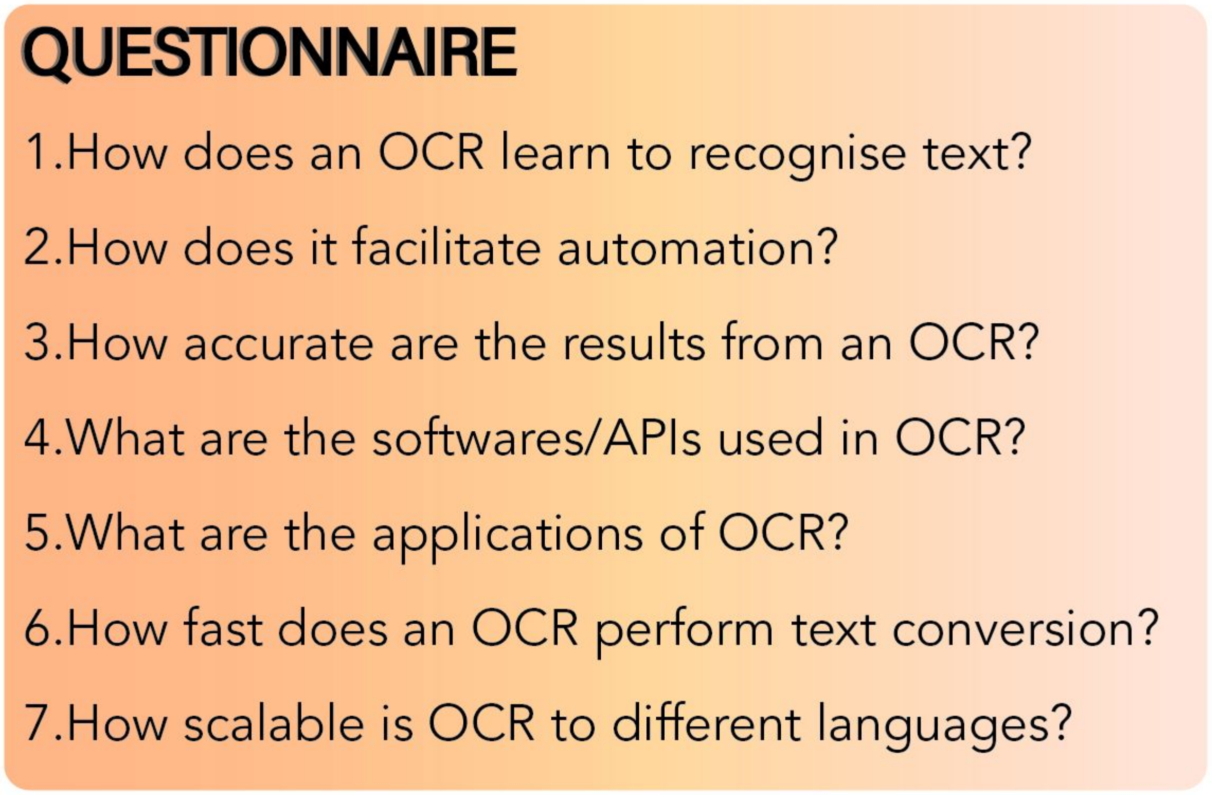
The second main concept used in the classifier design is based on the probabilistic approaches such as Bayes decision rule and maximum likelihood rule. The principle is to use a classification scheme that is optimal in the sense that, on average, it gives the lowest probability of making classification errors. A classifier that minimizes the total average loss is called the Bayes' classifier. Given an unknown symbol described by its feature vector, the probability that the symbol belongs to class c is computed for all classes c =1...N. The symbol is then assigned the class which gives the maximum probability. For this scheme to be optimal, the probability density functions of the symbols of each class must be known, along with the probability of occurrence of each class.

DESIGN THINKING PROCESS

EMPATHY PHASE

The first stage of the Design Thinking process involves developing a sense of empathy towards the people you are designing for, to gain insights into what they need, what they want, how they behave, feel, and think, and why they demonstrate such behaviours, feelings, and thoughts when interacting with products in a real-world setting. Since we were working on improving the efficiency of OCR for everybody, including us, we approached the problem by considering ourselves as the people that we need to empathize with.

As we were exploring various OCRs available, we realized that most of them are not very efficient. We tried different OCRs with various inputs and found that it gives the correct output in certain cases and incorrect output in certain cases. We did not understand why that was happening and we could not figure out the common character in all the different inputs due to which we were getting incorrect output.



QUESTIONNAIRE

1.How does an OCR learn to recognise text?

2.How does it facilitate automation?

3.How accurate are the results from an OCR?

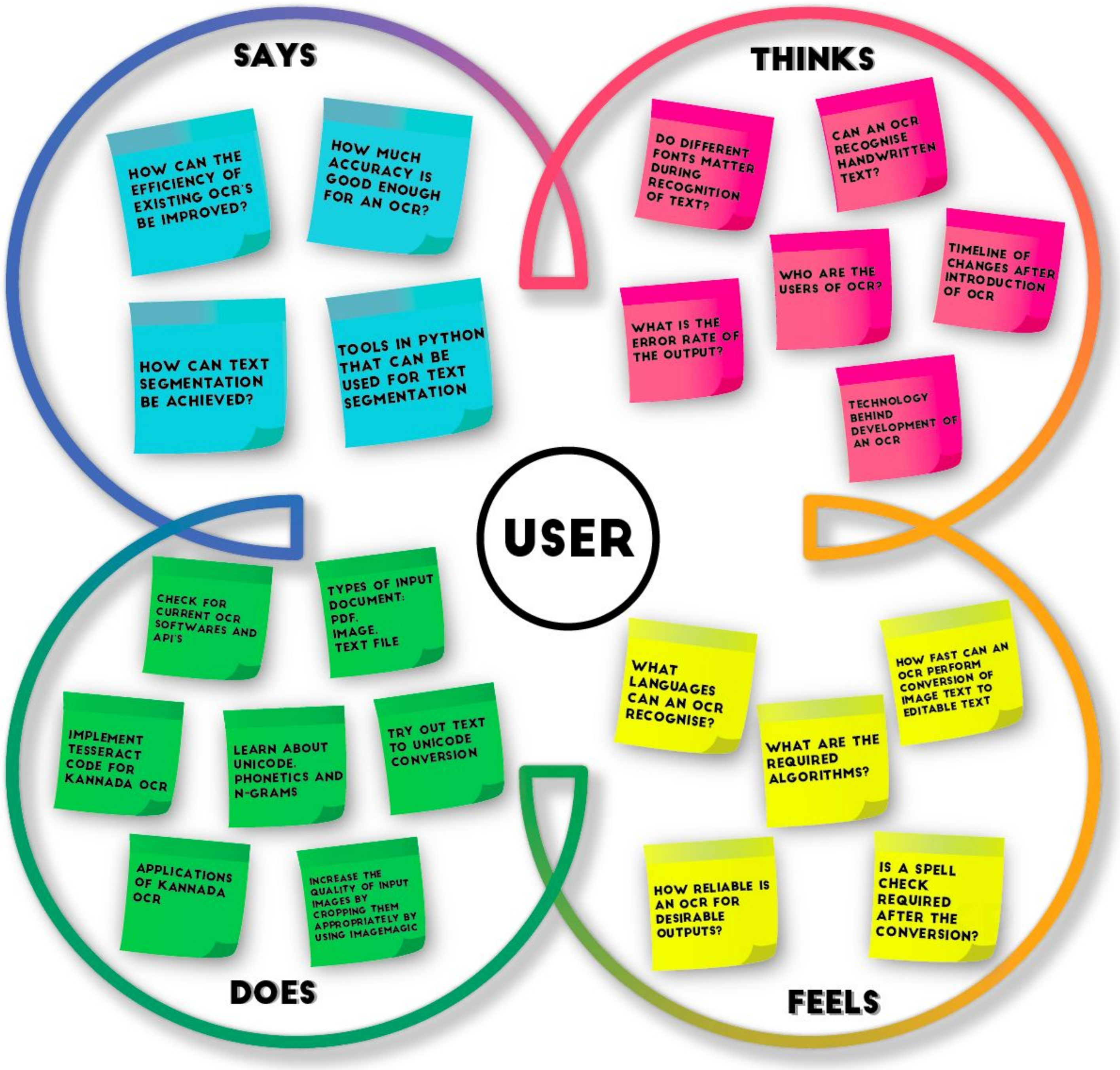
4.What are the softwares/APIs used in OCR?

5.What are the applications of OCR?

6.How fast does an OCR perform text conversion?

7.How scalable is OCR to different languages?

EMPATHY CHART


SAYS
HOW CAN THE EFFICIENCY OF EXISTING OCR'S BE IMPROVED?
HOW MUCH ACCURACY IS GOOD ENOUGH FOR AN OCR?
HOW CAN TEXT SEGMENTATION BE ACHIEVED?
TOOLS IN PYTHON THAT CAN BE USED FOR TEXT SEGMENTATION
THINKS
DO DIFFERENT FONTS MATTER DURING RECOGNITION OF TEXT?
CAN AN OCR RECOGNISE HANDWRITTEN TEXT?
WHO ARE THE USERS OF OCR?
TIMELINE OF CHANGES AFTER INTRODUCTION OF OCR
WHAT IS THE ERROR RATE OF THE OUTPUT?
TECHNOLOGY BEHIND DEVELOPMENT OF AN OCR
USER
DOES
CHECK FOR CURRENT OCR SOFTWARES AND API'S
TYPES OF INPUT DOCUMENT: PDF, IMAGE, TEXT FILE
IMPLEMENT TESSERACT CODE FOR KANNADA OCR
LEARN ABOUT UNICODE, PHONETICS AND N-GRAMS
TRY OUT TEXT TO UNICODE CONVERSION
APPLICATIONS OF KANNADA OCR
INCREASE THE QUALITY OF INPUT IMAGES BY CROPPING THEM APPROPRIATELY BY USING IMAGEMAGIC
FEELS
WHAT LANGUAGES CAN AN OCR RECOGNISE?
HOW FAST CAN AN OCR PERFORM CONVERSION OF IMAGE TEXT TO EDITABLE TEXT
WHAT ARE THE REQUIRED ALGORITHMS?
HOW RELIABLE IS AN OCR FOR DESIRABLE OUTPUTS?
IS A SPELL CHECK REQUIRED AFTER THE CONVERSION?

DEFINE PHASE

The main concern in the “define” phase of Design Thinking is around clearly articulating the problem you are trying to solve. Without clearly defining the problem, you will stumble in the dark and come up with solutions that don’t work. Our goal should be to frame the problem correctly. By doing so, we generate a variety of questions, which in turn give us different options and ways of thinking about the problem. In order to answer these questions, we use the data collected by members of the team through engagement with users during the “empathize” stage. This data is interpreted and assigned meaning by all members of the cross-functional team. As a result, more solutions will open up.

Initially we could not figure out why there were errors in the output only with certain inputs. So we discussed among ourselves and collectively observed all inputs that gave incorrect results as well as all the inputs that gave correct results. After looking at all the different inputs we realized that all the inputs containing small texts with just one or two paragraphs worked without any hassle but the inputs which were large in size were the only ones which gave errors and hence we were able to define our problem – All inputs of large sizes give incorrect results. Hence the “define” phase came to an end.

IDEATE PHASE

Ideate is the space in design thinking where individuals and teams elevate and celebrate the power of possibility. It is the transition from identifying a particular question or problem to generating a wide variety of potential answers and solutions. The whole process of ideation discourages linear thinking that supports one idea, often the “pet idea” of an authority figure or aggressive team member, arrived upon through a narrow group think brainstorming approach.

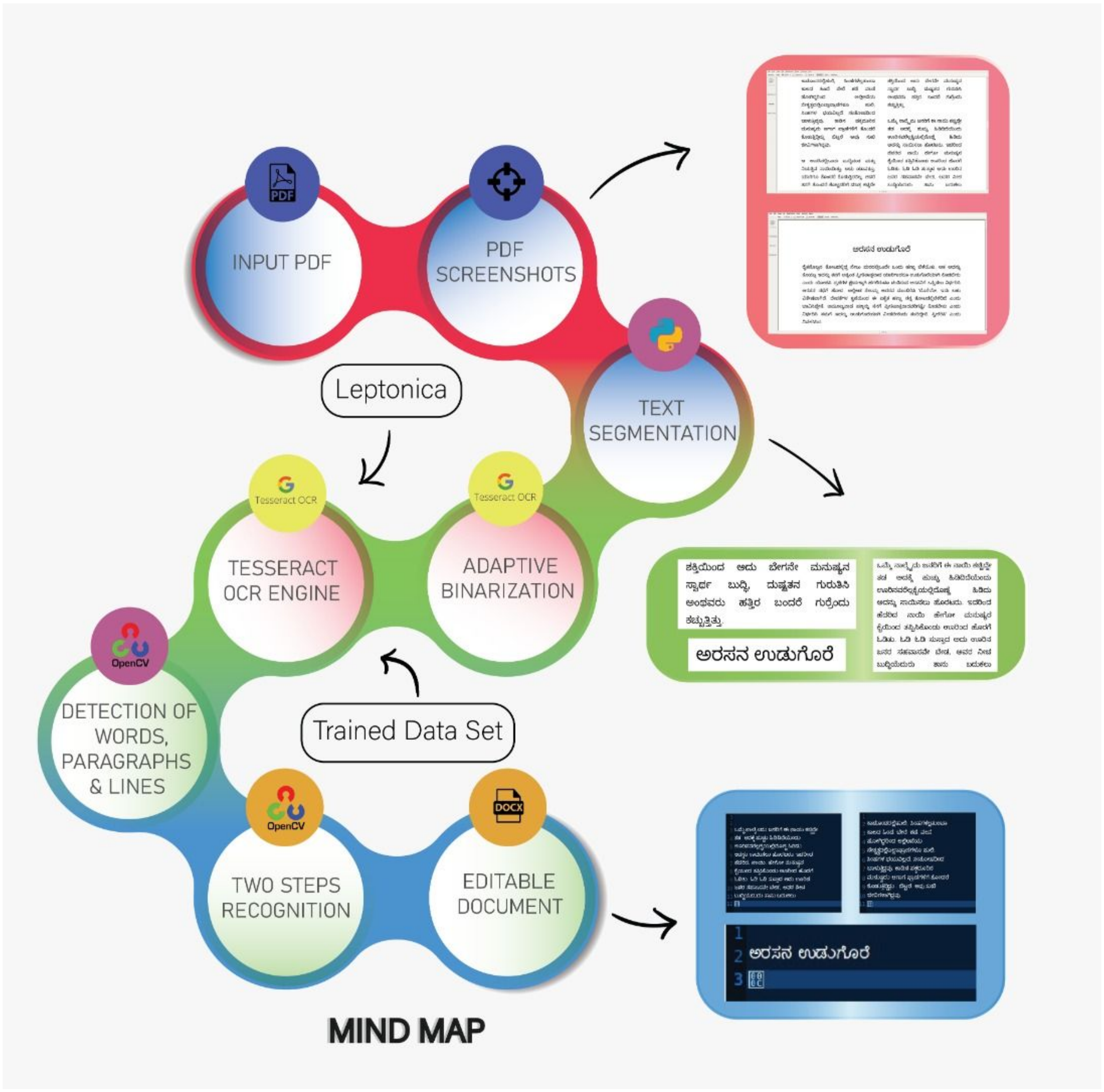
Initially in order to get some ideas and thoughts we referred some of the research papers and blogs. The previous works on the OCR system has been explained in detail in the literature survey section and hence we came to know about new concepts like text segmentation, image pre-processing, some algorithms used for pre-processing an image and the main stuff that is the algorithms based on neural networks for text recognition and its conversion. At the same time we found out that there already exists an OCR system for kannada called Tesseract-OCR developed by google. So our next step was to get our hands on it, test it, use our own text script images and get the output. By doing this we came to know that Tesseract-OCR can produce output for printed scripts but those outputs were not perfect and efficient, those output misspelt some words hence we thought to do something about it. Finally we came up with the idea of increasing the efficiency of the Tesseract-OCR using some of the tools available in the linux system. Looking at the outputs of the Tesseract-OCR we came to know that Tesseract-OCR gives efficient output for smaller inputs. So we thought to build a wrapper around Tesseract-OCR which will use some linux system tools and hence increase the efficiency of already existing Tesseract-OCR.

PROTOTYPE PHASE

The prototype stage is when you create a model designed to solve consumers' problems or validate ideas you can test in the next stage of the process. The importance of this stage is in building something that can be tested in the next and final stage of the design thinking process. The feedback received is critical for refining a final product that will be a good fit for the market. The prototype stage allows the design team to explore and perfect their solutions to the problems identified early on in a quick and cost-effective manner.

In order to enhance the already existing software that's the Tesseract OCR, we decided to use Imagemagik package to improve the quality of the input and modify it as required. Imagemagik is a package that's available in linux ubuntu which can be utilised to modify the properties of the image as per the requirements of the user . It utilizes multiple computational threads to increase performance and can read, process, or write mega-, giga-, or tera-pixel image sizes. This can be achieved by making use of linux commands such as identify, crop, resize, gravity etc. A detailed explanation of these commands is provided in the weekly analysis of the same. Adding these scripts served the purpose of automating the process of increasing the quality of input images that are fed to Tesseract OCR. This was accomplished by dividing an input image into multiple images of defined size and increased resolution, thereby reducing the number of words per input image which increased the accuracy in the output. Since the inputs were divided into several components, the output it gave was also divided corresponding to the inputs. So we merged the outputs of all the components to obtain an output text file that was complete and in order. The output efficiency of this process was much better than most of the other techniques that are used to achieve OCR.

MIND MAP



ಶಕ್ತಿಯಿಂದ ಅದು ಬೇಗನೇ ಮನುಷ್ಯನ ಸ್ವಾರ್ಥ ಬುದ್ಧಿ, ದುಷ್ಟತನ ಗುರುತಿಸಿ ಅಂಥವರು ಹತ್ತಿರ ಬಂದರೆ ಗುರ್ರೆಂದು ಕಚ್ಚುತ್ತಿತ್ತು.

ಅರಸನ ಉಡುಗೊರೆ

TEST PHASE

Testing can be undertaken throughout the progress of a Design Thinking project, although it is most commonly undertaken concurrently with the Prototyping stage. Testing, in Design Thinking, involves generating user feedback as related to the prototypes you have developed, as well as gaining a deeper understanding of your users. When undertaken correctly, the Testing stage of the project can often feed into most stages of the Design Thinking process: it allows you to Empathise and gain a better understanding of your users; it may lead to insights that change the way you Define your problem statement; it may generate new ideas in the Ideation stage; and finally, it might lead to an iteration of your Prototype.

We took into consideration various types of inputs which can be given for processing. Keeping that in mind, we experimented by giving inputs in different formats. This includes scanning handwritten texts, feeding digital images, scanning printed texts, altering the dimensions of the image, changing resolution of the image and so on. We also inculcated divide and conquer strategy to increase the efficiency while we were experimenting. Using shell commands, we cut the image into many parts and then scanned them exclusively. Later, we combined all the outputs and framed them into a single file. Although this proved to be more efficient, we had to build a wrapper to the code such that the entire process was completely automated. It was necessary to make it as less user-dependent as possible. So we used a code that automatically takes the screenshot of the text in a pdf and gives it as the input to the OCR. The main process is to generate a border/bounding box around the text recognised and process the image thus obtained. This is based on ssliding window technique wich can be easily achieved by making use of tenserflow or opencv pakages offered in linux. With these new additions made, the input was much more segmented and hence the code produced more accurate and less error-prone results.

WEEKLY ANALYSIS

WEEK 1

With the main aim of serving our society and creating and implementing something new that would be of some help to people, we chose to work for the literature domain of the society. One of the most common problems in the field of literature in India is transforming printed text into digitized format. We wanted to digitize Kannada script literary scripts as was the objective of Kannada Ganaka Parishat. With the ongoing Digital India Campaign, we realized this could contribute to the campaign. Hence, we decided to work on Optical Character Recognition (OCR). OCR can make your life easier by:

• Making paper-based information searchable in seconds, rather than hours.

• Reduce or eliminate costly data entry by automatically grabbing information you need from paper and putting it where it needs to go.

• Enabling entirely new ways to process documents that can eliminate “human touches”, thereby reducing costs and dramatically reducing processing times.

WEEK 2

In week 2, we explored further into the concept and took up various IEEE research papers and learnt that they use AI and Machine Learning as well as trained custom models. In practice, OCR involves:

Printout: We need to get the best possible printout of the existing document. The quality of the original printout makes a huge difference to the accuracy of the OCR process.

Scanning: We need to run the printout through the optical scanner.

Two-Color: The first stage in OCR involves generating a black-and-white (two-color/one-bit) version of the color or grayscale scanned page. OCR is essentially a binary process: it recognizes things that are either there or not. If the original scanned image is perfect, any black it contains will be part of a character that needs to be recognized while any white will be part of the background.

OCR: It should process the image of each page by recognizing the text character by character, word by word, and line by line.

Basic error correction: The program instantly processes the entire page and then uses a built-in spell checker to highlight any apparently misspelled words that may indicate a misrecognition, so you can automatically correct the mistake. This is optional.

Layout analysis: Good OCR programs automatically detect complex page layouts, such as multiple columns of text, tables, images, and so on. Images are automatically turned into graphics, tables are turned into tables, and columns are split up correctly automatically joined to the text from the first line of the second column).

Proofreading: Even the best OCR programs aren't perfect, especially when they're working from very old documents or poor quality printed text. That's why the final stage in OCR should always be a good, old-fashioned human proofread!

Text Recognition depends on a variety of factors to produce a good quality output. OCR output highly depends on the quality of input image. Here Image pre-processing comes into play to improve the quality of input image so that the OCR engine gives you an accurate output. Pre-processing steps include scaling of image, skew correction, binarization and noise removal. We studied about how the OCR is actually implemented.

WEEK 3

In the third week, we performed some ground work on the subject of interest. We tried to understand how Kannada characters are decoded and represented in different platforms. During this phase we came across the concepts of Unicode and N-grams.

Unicode is a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems. The Unicode Standard consists of a set of code charts for visual reference, an encoding method and set of standard character encodings, a set of reference data files, and a number of related items, such as character properties, rules for normalization, decomposition, collation, rendering, and bidirectional display order. Unicode can be implemented by different character encodings. The Unicode standard defines UTF-8, UTF-16, and UTF-32, and several other encodings are in use. Unicode covers almost all scripts (writing systems) in current use today. A total of 150 scripts are included in the latest version of Unicode including Kannada, Telugu and many other Indian languages.

An n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application.

An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram", size 3 is a "trigram". English cardinal numbers are sometimes used, e.g., "four-gram",

"five-gram", and so on. An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n − 1)–order. N-gram models are now widely used in probability, communication theory, computational linguistics, computational biology and data compression. Two benefits of n-gram models are simplicity and scalability. N-gram models are widely used in statistical natural language processing.
For example, the word KANNADA would have the following N-grams:
Bigrams (N = 2): _K, KA, AN, NN, NA, AD, DA, A_
Trigrams (N = 3): _KA, KAN, ANN, NNA, NAD,ADA,DA_, A__
Quadrigrams (N = 4): _KAN, KANN, ANNA, NNAD, NADA, ADA_, DA__, A___
Pentagram (N = 5): _KANN, KANNA, ANNAD, NNADA, NADA_, ADA__, DA___, A____
Hexagram (N = 6): _KANNA, KANNAD, ANNADA, NNADA_, NADA__, ADA___, DA____,
A_____ and similar constructs are followed for N = 7.

Because they decompose strings into smaller parts, N-gram-based matching is particularly effective in dealing with textual errors and character recognition issues in OCR generated documents. As these errors often affect only part of a word, words can still be identified.

WEEK 4

With pre-requisite knowledge of corpus concepts and phonetics of Kannada language, we then started exploring various OCRs which can be implemented for Kannada language. Our experimentation included testing and analyzing the outputs of them, with the objective of enhancing the efficiency of the OCRs.

Initially, we uploaded various Google images consisting of Kannada fonts in different formats. Further, we started to scan some handwritten fonts and analyzed the output and compared it with the latter. With this, we thought of enhancing the recognition of characters in Kannada handwritten scripts by various methods. In this regard, we learnt about types of Kannada keyboards along with mapping of the language with English alphabet keys.

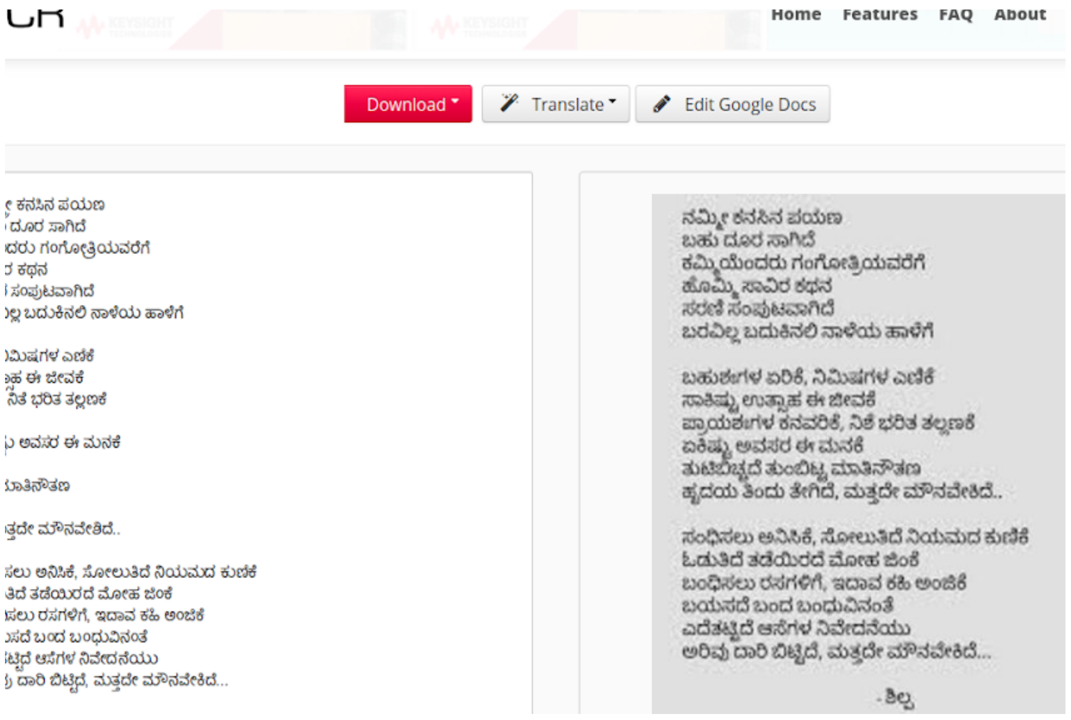
With marginal inefficiencies of the output, we also increased the resolution of the .jpg file manually and again scanned the image for the OCR. Thus, we obtained slight favorable changes with respect to the latter. .

The insightful knowledge of Unicode has majorly contributed towards ideation and further implementation of our concept. All the teammates discussed and shared knowledge about Unicode. We decided to transform the image into text format with the virtue of Unicode, which provides us the desired language i.e Kannada. Running the Unicode in python platform seemed comfortable to implement character recognition.

We concluded the session by discussing many methods to improvise our design and also with a load of thoughts regarding our mother language, Kannada and its scientific aspects.

WEEK 5

In week 5, we used an online OCR by i2OCR to test its efficiency with images of different kinds. i2OCR is software developed by Sciweavers which can convert text images of over 33 languages to editable text. It previously used to give corresponding unicode as output to any image that was fed to the software. Later, it was able to provide the text itself. It failed to recognize the handwritten Kannada text and the image from the camera. But, for an image with printed text, the following image was the output.



ಕನಸಿನ ಪಯಣ
ದೂರ ಸಾಗಿದೆ
ದರು ಗಂಗೋತ್ರಿಯವರೆಗೆ
ರ ಕಥನ
ಸಂಪುಟವಾಗಿದೆ
ಲ್ಲ ಬದುಕಿನಲಿ ನಾಳೆಯ ಹಾಳೆಗೆ

ಮಿಷಗಳ ಎಣಿಕೆ
ಹ ಈ ಜೀವಕೆ
ನಿತೆ ಭರಿತ ತಲ್ಲಣಕೆ

ಅವಸರ ಈ ಮನಕೆ

ತಿನೌತಣ

ತ್ತದೇ ಮೌನವೇತಿದೆ..

ಸಲು ಅನಿಸಿಕೆ, ಸೋಲುತಿದೆ ನಿಯಮದ ಕುಣಿಕೆ
ತಿದೆ ತಡೆಯಿರದೆ ಮೋಹ ಜಿಂಕೆ
ಸಲು ರಸಗಳಿಗೆ, ಇದಾವ ಕಹಿ ಅಂಜಿಕೆ
ಸದೆ ಬಂದ ಬಂಧುವಿನಂತೆ
ಟ್ಟಿದೆ ಆಸೆಗಳ ನಿವೇದನೆಯು
ದಾರಿ ಬಿಟ್ಟಿದೆ, ಮತ್ತದೇ ಮೌನವೇಕಿದೆ...

ನಮ್ಮೀ ಕನಸಿನ ಪಯಣ
ಬಹು ದೂರ ಸಾಗಿದೆ
ಕಮ್ಮಿಯೆಂದರು ಗಂಗೋತ್ರಿಯವರೆಗೆ
ಹೊಮ್ಮಿ ಸಾವಿರ ಕಥನ
ಸರಣಿ ಸಂಪುಟವಾಗಿದೆ
ಬರವಿಲ್ಲ ಬದುಕಿನಲಿ ನಾಳೆಯ ಹಾಳೆಗೆ

ಬಹುಶಃಗಳ ಏರಿಕೆ, ನಿಮಿಷಗಳ ಎಣಿಕೆ
ಸಾಕಿಷ್ಟು ಉತ್ಸಾಹ ಈ ಜೀವಕೆ
ಪ್ರಾಯಶಃಗಳ ಕನವರಿಕೆ, ನಿತೆ ಭರಿತ ತಲ್ಲಣಕೆ
ಏಕಿಷ್ಟು ಅವಸರ ಈ ಮನಕೆ
ತುಟಿಬಿಚ್ಚದೆ ತುಂಬಿಟ್ಟ ಮಾತಿನೌತಣ
ಹೃದಯ ತಿಂದು ತೇಗಿದೆ, ಮತ್ತದೇ ಮೌನವೇಕಿದೆ..

ಸಂಧಿಸಲು ಅನಿಸಿಕೆ, ಸೋಲುತಿದೆ ನಿಯಮದ ಕುಣಿಕೆ
ಓಡುತಿದೆ ತಡೆಯಿರದೆ ಮೋಹ ಜಿಂಕೆ
ಬಂಧಿಸಲು ರಸಗಳಿಗೆ, ಇದಾವ ಕಹಿ ಅಂಜಿಕೆ
ಬಯಸದೆ ಬಂದ ಬಂಧುವಿನಂತೆ
ಎದೆತಟ್ಟಿದೆ ಆಸೆಗಳ ನಿವೇದನೆಯು
ಅರಿವು ದಾರಿ ಬಿಟ್ಟಿದೆ, ಮತ್ತದೇ ಮೌನವೇಕಿದೆ...

- ಶಿಲ್ಪ

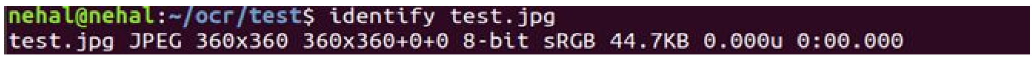
We observed that when we feed a page as input, there was scope for more morphological mistakes in the output text, as compared to the image which contained a paragraph or line.Results obtained were much closer to the expected output. Then there was a thought of how to increase the aptness of the resulting text without compromising with the input text size. This is where the idea to enhance the efficiency of the existing OCR for Kannada kicked in.

WEEK 6

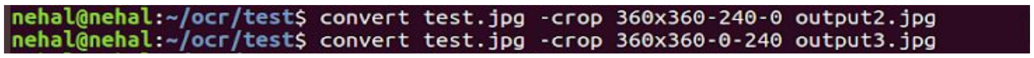
Finally we decided to use the already available open source Tesseract OCR and enhance the output produced by it. For this we decided to build a wrapper which will have Tesseract OCR has its core component, then the ImageMagick package and write some scripts which will automate our work in refining the input and feeding it to Tesseract OCR. The work of this wrapper will be cropping the images given as input last week in the form of small components. Then use the ImageMagick package to divide the source input image into finite small components by cropping and resizing it. This can be done with the help of commands. This will also increase the resolution of the cropped images and then give it as an input to Tesseract OCR.

ImagicMagick is a package in Linux Ubuntu which can be used to modify images. This package can be used using the commands in the terminal. Here are some of the commands which we used

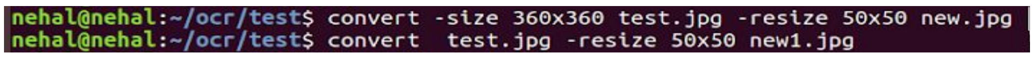
Properties :-identify <img_name>
Resize :-convert <img_name> -resize <resolution> <new_img_name>
Crop :-convert <img_name> -crop <resolution_and_dimensions> <cropped_img_name>
Crop from top :–convert <img_name> -gravity north <dimensions>
<cropped_img_name>
Crop from bottom :-convert <img_name> -gravity south <dimensions>
<cropped_img_name>



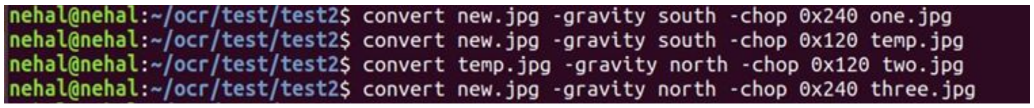
```
nehal@nehal:~/ocr/test$ identify test.jpg
test.jpg JPEG 360x360 360x360+0+0 8-bit sRGB 44.7KB 0.000u 0:00.000
```



```
nehal@nehal:~/ocr/test$ convert test.jpg -crop 360x360-240-0 output2.jpg
nehal@nehal:~/ocr/test$ convert test.jpg -crop 360x360-0-240 output3.jpg
```

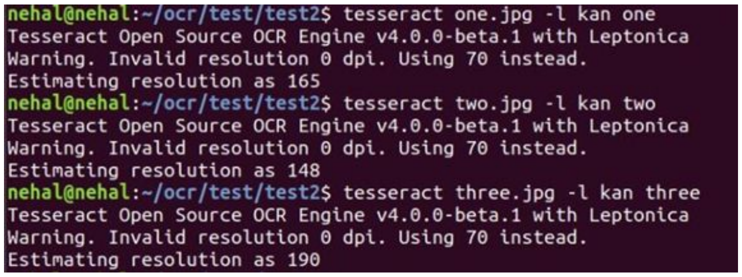


```
nehal@nehal:~/ocr/test$ convert -size 360x360 test.jpg -resize 50x50 new.jpg
nehal@nehal:~/ocr/test$ convert  test.jpg -resize 50x50 new1.jpg
```



```
nehal@nehal:~/ocr/test/test2$ convert new.jpg -gravity south -chop 0x240 one.jpg
nehal@nehal:~/ocr/test/test2$ convert new.jpg -gravity south -chop 0x120 temp.jpg
nehal@nehal:~/ocr/test/test2$ convert temp.jpg -gravity north -chop 0x120 two.jpg
nehal@nehal:~/ocr/test/test2$ convert new.jpg -gravity north -chop 0x240 three.jpg
```

By using all these commands we were able to get new input components. These components were given as input to the Tesseract OCR.
tesseract <input_component_name> -l kan <output_component_file name>



```
nehal@nehal:~/ocr/test/test2$ tesseract one.jpg -l kan one
Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
Warning. Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 165
nehal@nehal:~/ocr/test/test2$ tesseract two.jpg -l kan two
Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
Warning. Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 148
nehal@nehal:~/ocr/test/test2$ tesseract three.jpg -l kan three
Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
Warning. Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 190
```

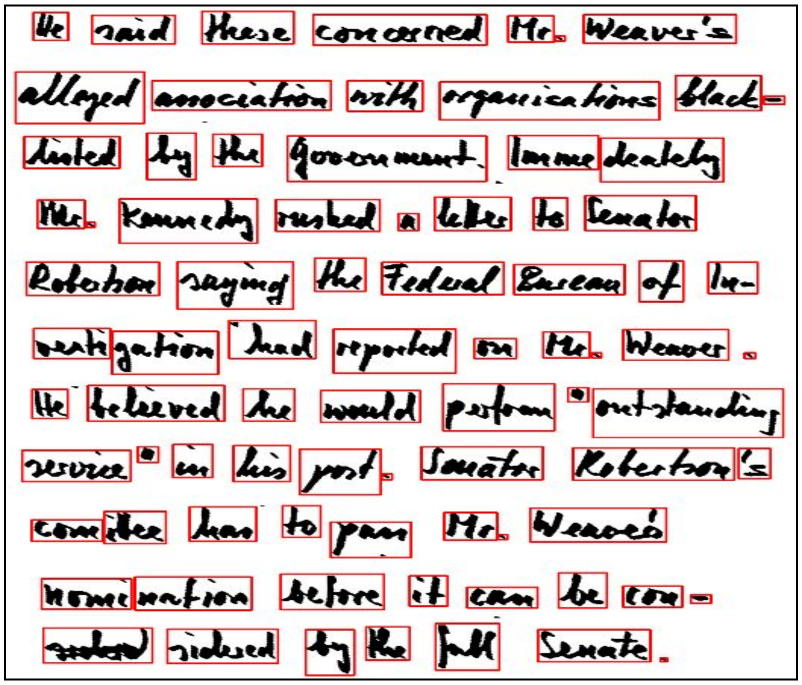
This was the command written in the terminal to get output components. The output which we got by implementing this method was more accurate when compared to the output which we got by providing the entire image as a single input. Some of the complicated words in kannada language were read correctly by the Tesseract OCR which resulted in more efficient output. Then we made some suitable changes like removing some invalid character obtained due to EOF (End of File) and merged the output components to obtain the major text file output of the input source image. The final output obtained in this process was more accurate and error-free when compared to the output obtained by following traditional methods.

So finally we decided to build a wrapper which will automate everything, starting from processing the input image and performing operations on it like resizing, cropping and then the Tesseract OCR processes the input components and gives respective outputs which will be further merged by the wrapper after making necessary changes.

WEEK 7

Finally for our wrapper we have a code which takes screenshots automatically from a pdf which is given as input and processes those images. The texts of the images are basically segmented and divided into subimages that contain only one paragraph. These subimages can be given as input to Tesseract-OCR and obtain outputs that are more efficient and errorless compared to the output obtained on giving the whole image as an input to the

Tesseract-OCR. The main process that goes on in the back-end is basically text segmentation. Text segmentation is a precursor to text retrieval, automatic summarization, information retrieval (IR); language modeling (LM) and natural language processing (NLP). In written texts, text segmentation is the process of identifying the boundaries between words, phrases, or some other linguistic meaningful units, such as sentences or topics. The term separated from such processing is useful to help humans reading texts, and are mainly used to assist computers to do some artificial processes as fundamental units, such as NLP, and IR.

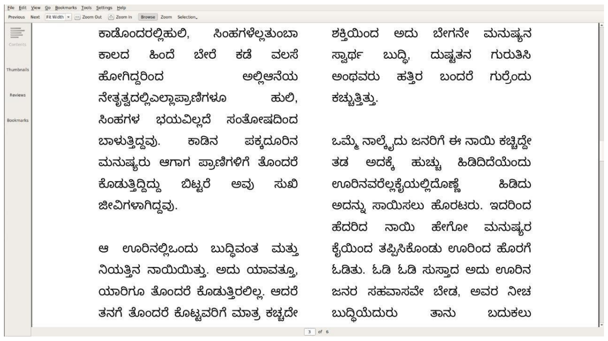


He said these concerned Mr. Weaver's
alleged association with organizations black-
listed by the Government. Immediately
Mr. Kennedy rushed a letter to Senator
Robertson saying the Federal Bureau of In-
vestigation had reported on Mr. Weaver.
He believed he would perform "outstanding
service" in his post. Senator Robertson's
committee has to pass Mr. Weaver's
nomination before it can be con-
~~sidered~~ sidered by the full Senate.

In the above picture we can see that the image containing some script has undergone image processing and the words are segmented. The words contain a bounding box which is the impact of text recognition. The bounding box technique is based on sliding window technique which can be implemented easily with the help of tensorflow as well as opencv. In our case the paragraphs are segmented from the image and hence these segmented text are given as input to Tesseract-OCR to get better and errorless output in the form of text.

WEEK 8

The result of the text segmentation are shown below:

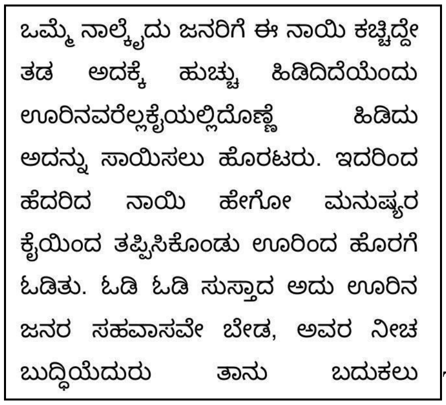

File Edit View Go Bookmarks Tools Settings Help
Previous Next Fit Width Zoom Out Zoom In Browse Zoom Selection
Contents
Thumbnails
Reviews
Bookmarks
3 of 6

ಕಾಡೊಂದರಲ್ಲಿಹುಲಿ, ಸಿಂಹಗಳೆಲ್ಲತುಂಬಾ ಕಾಲದ ಹಿಂದೆ ಬೇರೆ ಕಡೆ ವಲಸೆ ಹೋಗಿದ್ದರಿಂದ ಅಲ್ಲಿಆನೆಯ ನೇತೃತ್ವದಲ್ಲಿಎಲ್ಲಾಪ್ರಾಣಿಗಳೂ ಹುಲಿ, ಸಿಂಹಗಳ ಭಯವಿಲ್ಲದೆ ಸಂತೋಷದಿಂದ ಬಾಳುತ್ತಿದ್ದವು. ಕಾಡಿನ ಪಕ್ಕದೂರಿನ ಮನುಷ್ಯರು ಆಗಾಗ ಪ್ರಾಣಿಗಳಿಗೆ ತೊಂದರೆ ಕೊಡುತ್ತಿದ್ದಿದ್ದು ಬಿಟ್ಟರೆ ಅವು ಸುಖಿ ಜೀವಿಗಳಾಗಿದ್ದವು.

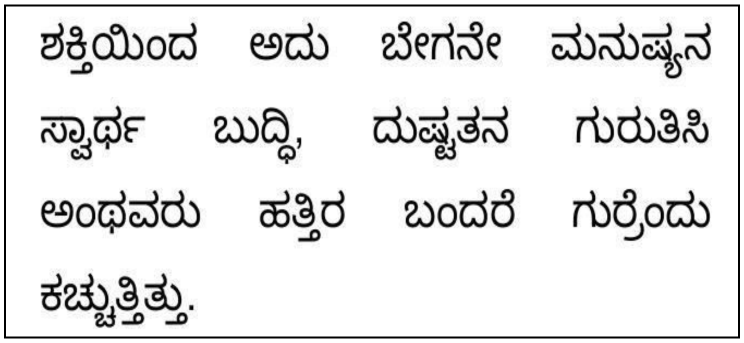
ಆ ಊರಿನಲ್ಲಿಒಂದು ಬುದ್ಧಿವಂತ ಮತ್ತು ನಿಯತ್ತಿನ ನಾಯಿಯಿತ್ತು. ಅದು ಯಾವತ್ತೂ, ಯಾರಿಗೂ ತೊಂದರೆ ಕೊಡುತ್ತಿರಲಿಲ್ಲ. ಆದರೆ ತನಗೆ ತೊಂದರೆ ಕೊಟ್ಟವರಿಗೆ ಮಾತ್ರ ಕಚ್ಚದೇ ಶಕ್ತಿಯಿಂದ ಅದು ಬೇಗನೇ ಮನುಷ್ಯನ ಸ್ವಾರ್ಥ ಬುದ್ಧಿ, ದುಷ್ಟತನ ಗುರುತಿಸಿ ಅಂಥವರು ಹತ್ತಿರ ಬಂದರೆ ಗುರ್ರೆಂದು ಕಚ್ಚುತ್ತಿತ್ತು.

ಒಮ್ಮೆ ನಾಲ್ಕೈದು ಜನರಿಗೆ ಈ ನಾಯಿ ಕಚ್ಚಿದ್ದೇ ತಡ ಅದಕ್ಕೆ ಹುಚ್ಚು ಹಿಡಿದಿದೆಯೆಂದು ಊರಿನವರೆಲ್ಲಕೈಯಲ್ಲಿದೊಣ್ಣೆ ಹಿಡಿದು ಅದನ್ನು ಸಾಯಿಸಲು ಹೊರಟರು. ಇದರಿಂದ ಹೆದರಿದ ನಾಯಿ ಹೇಗೋ ಮನುಷ್ಯರ ಕೈಯಿಂದ ತಪ್ಪಿಸಿಕೊಂಡು ಊರಿಂದ ಹೊರಗೆ ಓಡಿತು. ಓಡಿ ಓಡಿ ಸುಸ್ತಾದ ಅದು ಊರಿನ ಜನರ ಸಹವಾಸವೇ ಬೇಡ, ಅವರ ನೀಚ ಬುದ್ಧಿಯೆದುರು ತಾನು ಬದುಕಲು

The screenshots taken from pdf by the code:

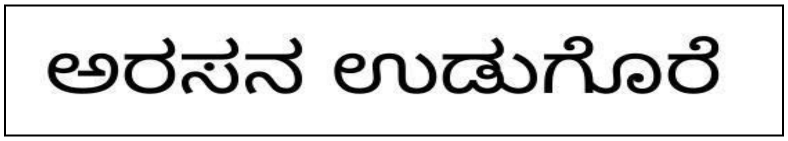


ಒಮ್ಮೆ ನಾಲ್ಕೈದು ಜನರಿಗೆ ಈ ನಾಯಿ ಕಚ್ಚಿದ್ದೇ ತಡ ಅದಕ್ಕೆ ಹುಚ್ಚು ಹಿಡಿದಿದೆಯೆಂದು ಊರಿನವರೆಲ್ಲಕೈಯಲ್ಲಿದೊಣ್ಣೆ ಹಿಡಿದು ಅದನ್ನು ಸಾಯಿಸಲು ಹೊರಟರು. ಇದರಿಂದ ಹೆದರಿದ ನಾಯಿ ಹೇಗೋ ಮನುಷ್ಯರ ಕೈಯಿಂದ ತಪ್ಪಿಸಿಕೊಂಡು ಊರಿಂದ ಹೊರಗೆ ಓಡಿತು. ಓಡಿ ಓಡಿ ಸುಸ್ತಾದ ಅದು ಊರಿನ ಜನರ ಸಹವಾಸವೇ ಬೇಡ, ಅವರ ನೀಚ ಬುದ್ಧಿಯೆದುರು ತಾನು ಬದುಕಲು



ಶಕ್ತಿಯಿಂದ ಅದು ಬೇಗನೇ ಮನುಷ್ಯನ ಸ್ವಾರ್ಥ ಬುದ್ಧಿ, ದುಷ್ಟತನ ಗುರುತಿಸಿ ಅಂಥವರು ಹತ್ತಿರ ಬಂದರೆ ಗುರ್ರೆಂದು ಕಚ್ಚುತ್ತಿತ್ತು.

The output after text segmentation

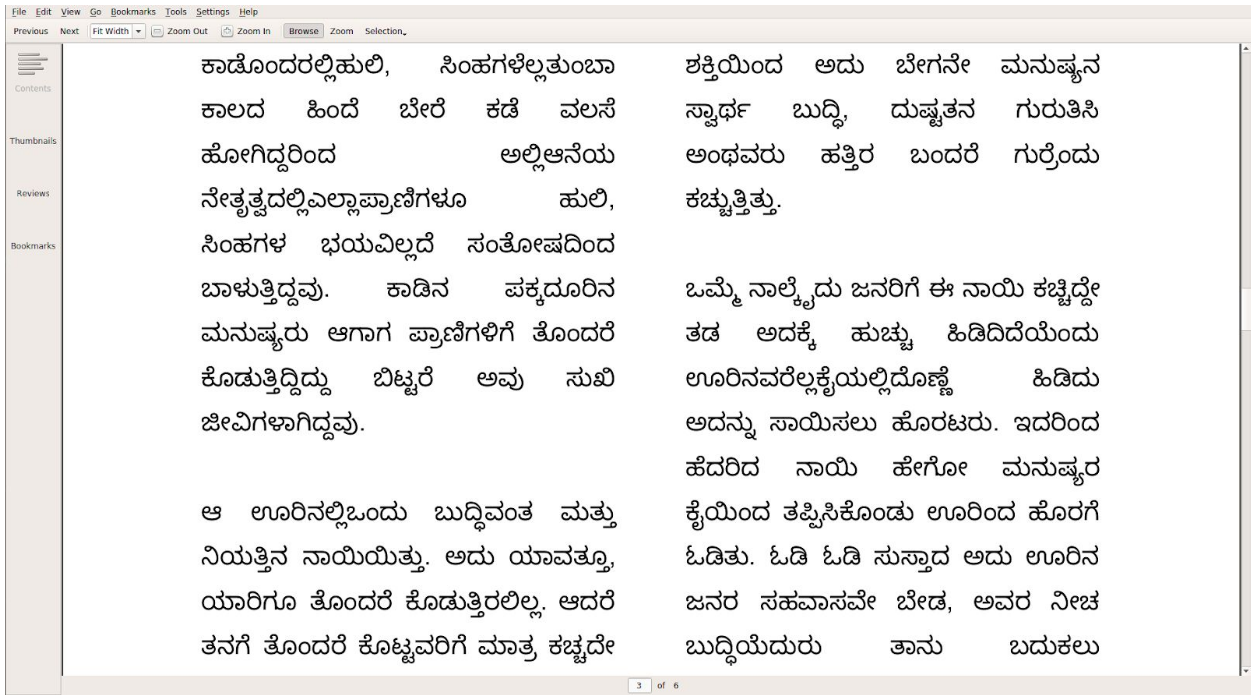


ಅರಸನ ಉಡುಗೊರೆ

These outputs are the images of the segmented paragraphs of the input image. These segmented paragraphs will be given as input to Tesseract-OCR and hence we obtain an efficient text output for the corresponding sub images which can be then brought together to obtain the output for the whole image in the form of text. So this process indeed improves the efficiency of Tesseract-OCR.

OUTPUTS

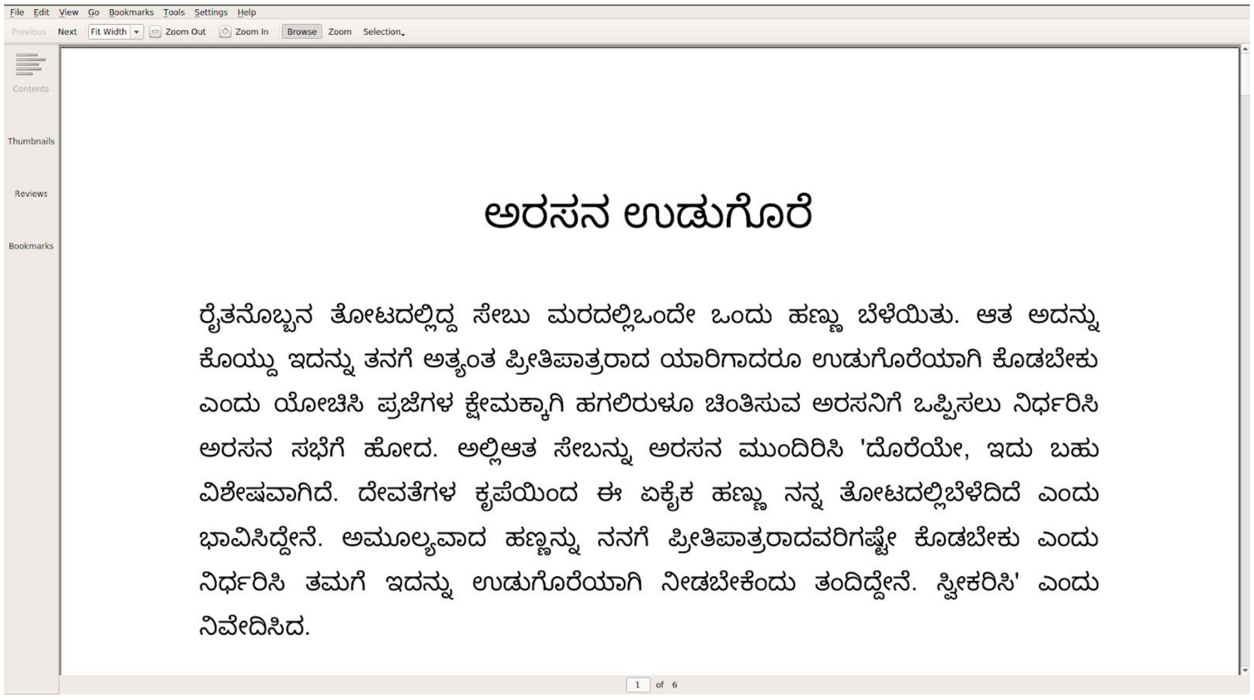
A few screenshots of the PDF taken by scrolling it down as defined in the code:



ಕಾಡೊಂದರಲ್ಲಿಹುಲಿ, ಸಿಂಹಗಳೆಲ್ಲತುಂಬಾ ಕಾಲದ ಹಿಂದೆ ಬೇರೆ ಕಡೆ ವಲಸೆ ಹೋಗಿದ್ದರಿಂದ ಅಲ್ಲಿಆನೆಯ ನೇತೃತ್ವದಲ್ಲಿಎಲ್ಲಾಪ್ರಾಣಿಗಳೂ ಹುಲಿ, ಸಿಂಹಗಳ ಭಯವಿಲ್ಲದೆ ಸಂತೋಷದಿಂದ ಬಾಳುತ್ತಿದ್ದವು. ಕಾಡಿನ ಪಕ್ಕದೂರಿನ ಮನುಷ್ಯರು ಆಗಾಗ ಪ್ರಾಣಿಗಳಿಗೆ ತೊಂದರೆ ಕೊಡುತ್ತಿದ್ದಿದ್ದು ಬಿಟ್ಟರೆ ಅವು ಸುಖಿ ಜೀವಿಗಳಾಗಿದ್ದವು.

ಆ ಊರಿನಲ್ಲಿಒಂದು ಬುದ್ಧಿವಂತ ಮತ್ತು ನಿಯತ್ತಿನ ನಾಯಿಯಿತ್ತು. ಅದು ಯಾವತ್ತೂ, ಯಾರಿಗೂ ತೊಂದರೆ ಕೊಡುತ್ತಿರಲಿಲ್ಲ. ಆದರೆ ತನಗೆ ತೊಂದರೆ ಕೊಟ್ಟವರಿಗೆ ಮಾತ್ರ ಕಚ್ಚದೇ ಶಕ್ತಿಯಿಂದ ಅದು ಬೇಗನೇ ಮನುಷ್ಯನ ಸ್ವಾರ್ಥ ಬುದ್ಧಿ, ದುಷ್ಟತನ ಗುರುತಿಸಿ ಅಂಥವರು ಹತ್ತಿರ ಬಂದರೆ ಗುರ್ರೆಂದು ಕಚ್ಚುತ್ತಿತ್ತು.

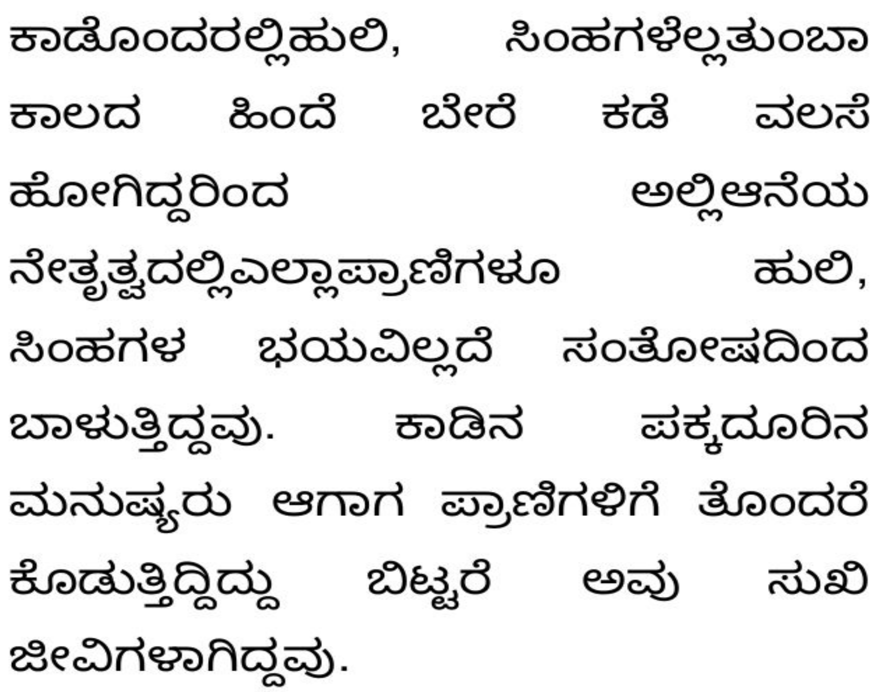
ಒಮ್ಮೆ ನಾಲ್ಕೈದು ಜನರಿಗೆ ಈ ನಾಯಿ ಕಚ್ಚಿದ್ದೇ ತಡ ಅದಕ್ಕೆ ಹುಚ್ಚು ಹಿಡಿದಿದೆಯೆಂದು ಊರಿನವರೆಲ್ಲಕೈಯಲ್ಲಿದೊಣ್ಣೆ ಹಿಡಿದು ಅದನ್ನು ಸಾಯಿಸಲು ಹೊರಟರು. ಇದರಿಂದ ಹೆದರಿದ ನಾಯಿ ಹೇಗೋ ಮನುಷ್ಯರ ಕೈಯಿಂದ ತಪ್ಪಿಸಿಕೊಂಡು ಊರಿಂದ ಹೊರಗೆ ಓಡಿತು. ಓಡಿ ಓಡಿ ಸುಸ್ತಾದ ಅದು ಊರಿನ ಜನರ ಸಹವಾಸವೇ ಬೇಡ, ಅವರ ನೀಚ ಬುದ್ಧಿಯೆದುರು ತಾನು ಬದುಕಲು

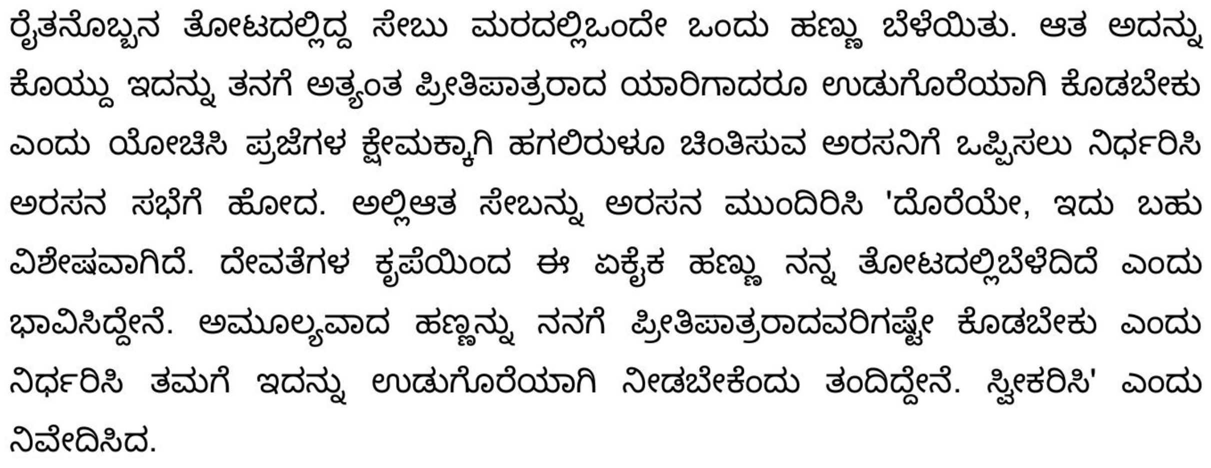


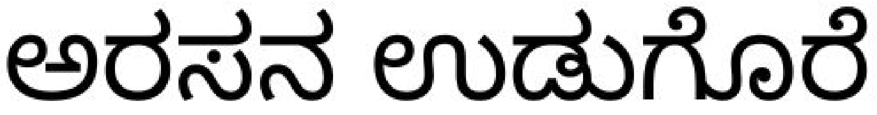
ಅರಸನ ಉಡುಗೊರೆ

ರೈತನೊಬ್ಬನ ತೋಟದಲ್ಲಿದ್ದ ಸೇಬು ಮರದಲ್ಲಿಒಂದೇ ಒಂದು ಹಣ್ಣು ಬೆಳೆಯಿತು. ಆತ ಅದನ್ನು ಕೊಯ್ದು ಇದನ್ನು ತನಗೆ ಅತ್ಯಂತ ಪ್ರೀತಿಪಾತ್ರರಾದ ಯಾರಿಗಾದರೂ ಉಡುಗೊರೆಯಾಗಿ ಕೊಡಬೇಕು ಎಂದು ಯೋಚಿಸಿ ಪ್ರಜೆಗಳ ಕ್ಷೇಮಕ್ಕಾಗಿ ಹಗಲಿರುಳೂ ಚಿಂತಿಸುವ ಅರಸನಿಗೆ ಒಪ್ಪಿಸಲು ನಿರ್ಧರಿಸಿ ಅರಸನ ಸಭೆಗೆ ಹೋದ. ಅಲ್ಲಿಆತ ಸೇಬನ್ನು ಅರಸನ ಮುಂದಿರಿಸಿ 'ದೊರೆಯೇ, ಇದು ಬಹು ವಿಶೇಷವಾಗಿದೆ. ದೇವತೆಗಳ ಕೃಪೆಯಿಂದ ಈ ಏಕೈಕ ಹಣ್ಣು ನನ್ನ ತೋಟದಲ್ಲಿಬೆಳೆದಿದೆ ಎಂದು ಭಾವಿಸಿದ್ದೇನೆ. ಅಮೂಲ್ಯವಾದ ಹಣ್ಣನ್ನು ನನಗೆ ಪ್ರೀತಿಪಾತ್ರರಾದವರಿಗಷ್ಟೇ ಕೊಡಬೇಕು ಎಂದು ನಿರ್ಧರಿಸಿ ತಮಗೆ ಇದನ್ನು ಉಡುಗೊರೆಯಾಗಿ ನೀಡಬೇಕೆಂದು ತಂದಿದ್ದೇನೆ. ಸ್ವೀಕರಿಸಿ' ಎಂದು ನಿವೇದಿಸಿದ.

Images obtained after text segmentation:


ಕಾಡೊಂದರಲ್ಲಿಹುಲಿ, ಸಿಂಹಗಳೆಲ್ಲತುಂಬಾ
ಕಾಲದ ಹಿಂದೆ ಬೇರೆ ಕಡೆ ವಲಸೆ
ಹೋಗಿದ್ದರಿಂದ ಅಲ್ಲಿಆನೆಯ
ನೇತೃತ್ವದಲ್ಲಿಎಲ್ಲಾಪ್ರಾಣಿಗಳೂ ಹುಲಿ,
ಸಿಂಹಗಳ ಭಯವಿಲ್ಲದೆ ಸಂತೋಷದಿಂದ
ಬಾಳುತ್ತಿದ್ದವು. ಕಾಡಿನ ಪಕ್ಕದೂರಿನ
ಮನುಷ್ಯರು ಆಗಾಗ ಪ್ರಾಣಿಗಳಿಗೆ ತೊಂದರೆ
ಕೊಡುತ್ತಿದ್ದಿದ್ದು ಬಿಟ್ಟರೆ ಅವು ಸುಖಿ
ಜೀವಿಗಳಾಗಿದ್ದವು.


ರೈತನೊಬ್ಬನ ತೋಟದಲ್ಲಿದ್ದ ಸೇಬು ಮರದಲ್ಲಿಒಂದೇ ಒಂದು ಹಣ್ಣು ಬೆಳೆಯಿತು. ಆತ ಅದನ್ನು
ಕೊಯ್ದು ಇದನ್ನು ತನಗೆ ಅತ್ಯಂತ ಪ್ರೀತಿಪಾತ್ರರಾದ ಯಾರಿಗಾದರೂ ಉಡುಗೊರೆಯಾಗಿ ಕೊಡಬೇಕು
ಎಂದು ಯೋಚಿಸಿ ಪ್ರಜೆಗಳ ಕ್ಷೇಮಕ್ಕಾಗಿ ಹಗಲಿರುಳೂ ಚಿಂತಿಸುವ ಅರಸನಿಗೆ ಒಪ್ಪಿಸಲು ನಿರ್ಧರಿಸಿ
ಅರಸನ ಸಭೆಗೆ ಹೋದ. ಅಲ್ಲಿಆತ ಸೇಬನ್ನು ಅರಸನ ಮುಂದಿರಿಸಿ 'ದೊರೆಯೇ, ಇದು ಬಹು
ವಿಶೇಷವಾಗಿದೆ. ದೇವತೆಗಳ ಕೃಪೆಯಿಂದ ಈ ಏಕೈಕ ಹಣ್ಣು ನನ್ನ ತೋಟದಲ್ಲಿಬೆಳೆದಿದೆ ಎಂದು
ಭಾವಿಸಿದ್ದೇನೆ. ಅಮೂಲ್ಯವಾದ ಹಣ್ಣನ್ನು ನನಗೆ ಪ್ರೀತಿಪಾತ್ರರಾದವರಿಗಷ್ಟೇ ಕೊಡಬೇಕು ಎಂದು
ನಿರ್ಧರಿಸಿ ತಮಗೆ ಇದನ್ನು ಉಡುಗೊರೆಯಾಗಿ ನೀಡಬೇಕೆಂದು ತಂದಿದ್ದೇನೆ. ಸ್ವೀಕರಿಸಿ' ಎಂದು
ನಿವೇದಿಸಿದ.

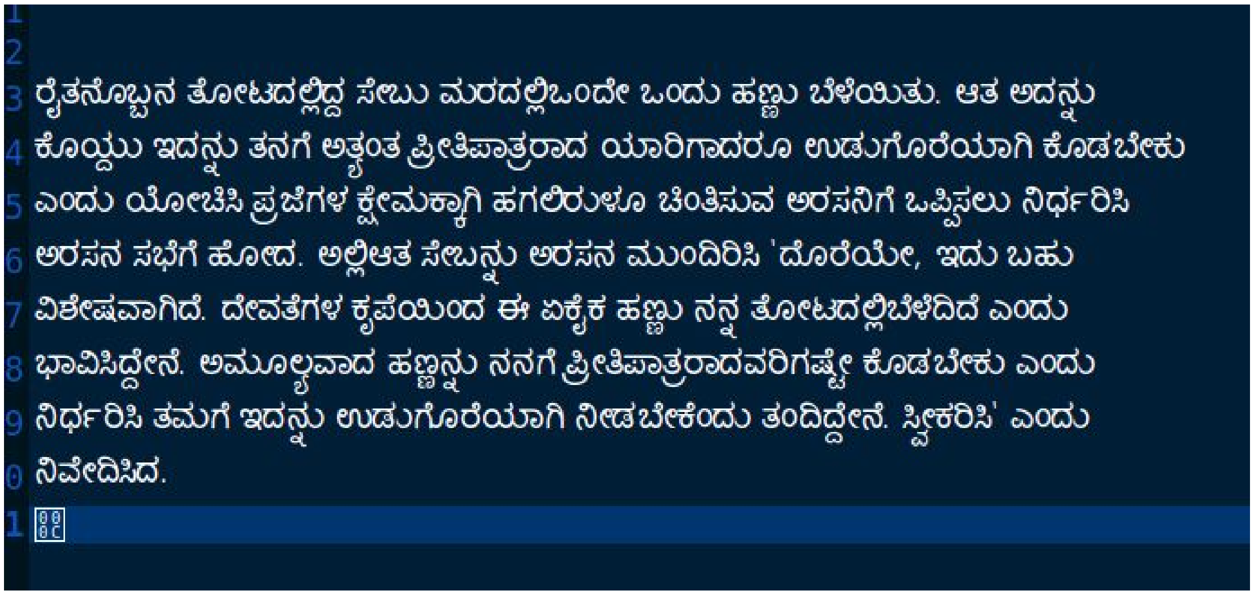

ಅರಸನ ಉಡುಗೊರೆ

ಶಕ್ತಿಯಿಂದ ಅದು ಬೇಗನೇ ಮನುಷ್ಯನ ಸ್ವಾರ್ಥ ಬುದ್ಧಿ, ದುಷ್ಟತನ ಗುರುತಿಸಿ ಅಂಥವರು ಹತ್ತಿರ ಬಂದರೆ ಗುರ್ರೆಂದು ಕಚ್ಚುತ್ತಿತ್ತು.

ಒಮ್ಮೆ ನಾಲ್ಕೈದು ಜನರಿಗೆ ಈ ನಾಯಿ ಕಚ್ಚಿದ್ದೇ ತಡ ಅದಕ್ಕೆ ಹುಚ್ಚು ಹಿಡಿದಿದೆಯೆಂದು ಊರಿನವರೆಲ್ಲಕೈಯಲ್ಲಿದೊಣ್ಣೆ ಹಿಡಿದು ಅದನ್ನು ಸಾಯಿಸಲು ಹೊರಟರು. ಇದರಿಂದ ಹೆದರಿದ ನಾಯಿ ಹೇಗೋ ಮನುಷ್ಯರ ಕೈಯಿಂದ ತಪ್ಪಿಸಿಕೊಂಡು ಊರಿಂದ ಹೊರಗೆ ಓಡಿತು. ಓಡಿ ಓಡಿ ಸುಸ್ತಾದ ಅದು ಊರಿನ ಜನರ ಸಹವಾಸವೇ ಬೇಡ, ಅವರ ನೀಚ ಬುದ್ಧಿಯೆದುರು ತಾನು ಬದುಕಲು

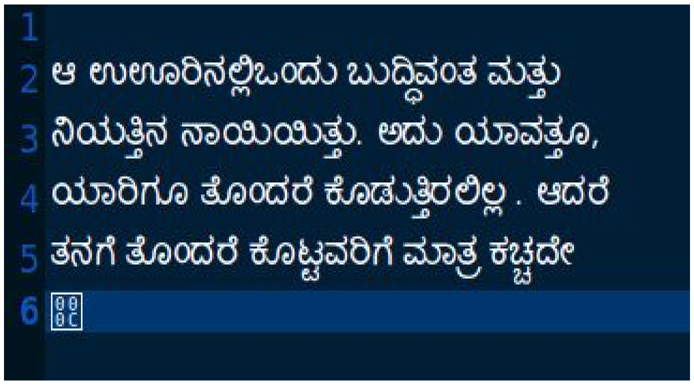
ಆ ಊರಿನಲ್ಲಿಒಂದು ಬುದ್ಧಿವಂತ ಮತ್ತು ನಿಯತ್ತಿನ ನಾಯಿಯಿತ್ತು. ಅದು ಯಾವತ್ತೂ, ಯಾರಿಗೂ ತೊಂದರೆ ಕೊಡುತ್ತಿರಲಿಲ್ಲ. ಆದರೆ ತನಗೆ ತೊಂದರೆ ಕೊಟ್ಟವರಿಗೆ ಮಾತ್ರ ಕಚ್ಚದೇ

Outputs obtained after Tesseract OCR converts the text to editable form from the text segmented images Editable text obtained as output from Tesseract OCR:



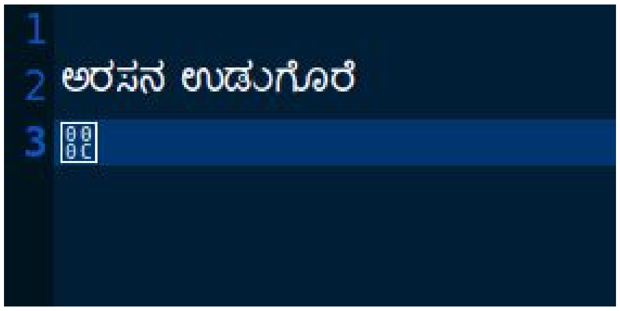
```


ರೈತನೊಬ್ಬನ ತೋಟದಲ್ಲಿದ್ದ ಸೇಬು ಮರದಲ್ಲಿಒಂದೇ ಒಂದು ಹಣ್ಣು ಬೆಳೆಯಿತು. ಆತ ಅದನ್ನು
ಕೊಯ್ದು ಇದನ್ನು ತನಗೆ ಅತ್ಯಂತ ಪ್ರೀತಿಪಾತ್ರರಾದ ಯಾರಿಗಾದರೂ ಉಡುಗೊರೆಯಾಗಿ ಕೊಡಬೇಕು
ಎಂದು ಯೋಚಿಸಿ ಪ್ರಜೆಗಳ ಕ್ಷೇಮಕ್ಕಾಗಿ ಹಗಲಿರುಳೂ ಚಿಂತಿಸುವ ಅರಸನಿಗೆ ಒಪ್ಪಿಸಲು ನಿರ್ಧರಿಸಿ
ಅರಸನ ಸಭೆಗೆ ಹೋದ. ಅಲ್ಲಿಆತ ಸೇಬನ್ನು ಅರಸನ ಮುಂದಿರಿಸಿ 'ದೊರೆಯೇ, ಇದು ಬಹು
ವಿಶೇಷವಾಗಿದೆ. ದೇವತೆಗಳ ಕೃಪೆಯಿಂದ ಈ ಏಕೈಕ ಹಣ್ಣು ನನ್ನ ತೋಟದಲ್ಲಿಬೆಳೆದಿದೆ ಎಂದು
ಭಾವಿಸಿದ್ದೇನೆ. ಅಮೂಲ್ಯವಾದ ಹಣ್ಣನ್ನು ನನಗೆ ಪ್ರೀತಿಪಾತ್ರರಾದವರಿಗಷ್ಟೇ ಕೊಡಬೇಕು ಎಂದು
ನಿರ್ಧರಿಸಿ ತಮಗೆ ಇದನ್ನು ಉಡುಗೊರೆಯಾಗಿ ನೀಡಬೇಕೆಂದು ತಂದಿದ್ದೇನೆ. ಸ್ವೀಕರಿಸಿ' ಎಂದು
ನಿವೇದಿಸಿದ.
```



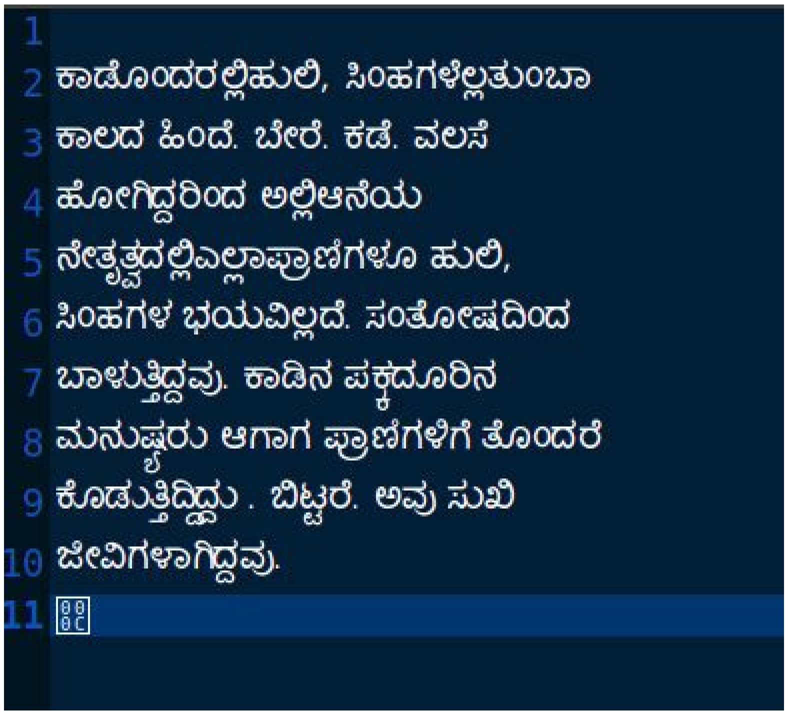
```

ಆ ಉಊರಿನಲ್ಲಿಒಂದು ಬುದ್ಧಿವಂತ ಮತ್ತು
ನಿಯತ್ತಿನ ನಾಯಿಯಿತ್ತು. ಅದು ಯಾವತ್ತೂ,
ಯಾರಿಗೂ ತೊಂದರೆ ಕೊಡುತ್ತಿರಲಿಲ್ಲ . ಆದರೆ
ತನಗೆ ತೊಂದರೆ ಕೊಟ್ಟವರಿಗೆ ಮಾತ್ರ ಕಚ್ಚದೇ
```



```

ಅರಸನ ಉಡುಗೊರೆ
```



```

ಕಾಡೊಂದರಲ್ಲಿಹುಲಿ, ಸಿಂಹಗಳೆಲ್ಲತುಂಬಾ
ಕಾಲದ ಹಿಂದೆ. ಬೇರೆ. ಕಡೆ. ವಲಸೆ
ಹೋಗಿದ್ದರಿಂದ ಅಲ್ಲಿಆನೆಯ
ನೇತೃತ್ವದಲ್ಲಿಎಲ್ಲಾಪ್ರಾಣಿಗಳೂ ಹುಲಿ,
ಸಿಂಹಗಳ ಭಯವಿಲ್ಲದೆ. ಸಂತೋಷದಿಂದ
ಬಾಳುತ್ತಿದ್ದವು. ಕಾಡಿನ ಪಕ್ಕದೂರಿನ
ಮನುಷ್ಯರು ಆಗಾಗ ಪ್ರಾಣಿಗಳಿಗೆ ತೊಂದರೆ
ಕೊಡುತ್ತಿದ್ದು . ಬಿಟ್ಟರೆ. ಅವು ಸುಖಿ
ಜೀವಿಗಳಾಗಿದ್ದವು.
```

SIGNIFICANCE OF OCRs

Indian society has evolved with its extrinsic predominance of civilization and ever remembered literature. The glory of the past has been enshrined in the manuscripts and palm leaves and many other forms of publications before print culture was evolved. In the age of the digitized era, it is of prime importance and responsibility to preserve and pass on these valuable contents to a vibrant upcoming generation.

The treasure equivalent civilization of the medieval age and modern versatile folklore have constituted a major standard of Indian literature. But these are still existing in originally hand written format, which is very tedious to reproduce and which may eventually destroy if proper maintenance isn't made.

E-Literature is easily accessible, widely approachable, reproducible and can be effectively preserved for a gigantic amount of time. This appealingly calls for a demand for an effective OCR with impressive efficiency to successively document the paper works to the cloud. Although advanced OCRs are available in English, which is the widely used language in the digital world, it is still not considerably effective for our Indic languages.

Several researches have been done to digitize Indic languages which are in their primitive form. However, OCRs are error prone. This may be caused because of poor image framing, inadequate pre-processing, improper documentation or complex fonts. As these constraints are irreparable, we must look forward in coming over the limitations by exploring technical advancements and modern approaches.

Functioning of OCRs occurs in three stages. Namely; Documentation, Text-Recognition and Post-processing. Our main focus of exploration is for Kannada language. Since Kannada is entangled with a systematic grammar for combination and bifurcation of words, we can frame a set of rules and operations to contemplate the efficiency of the character recogniser. Once we achieve a great intact of the output delivered, we can get along to digitize the scriptures with complete automation and processing.

OCRs must be tailored with utmost proximity in scanning and recognition of Dravidian languages. The circular strokes of these scripts constitute various letters, results are more error prone and require additional rectification.

A basic OCR system operating procedure includes three major stages.

Pre-processing:

This step involves preparing the image to ensure that the best quality image is input to the next stage. The scanned pages or images usually contain noises which hinder the recognition process. Methods used for pre-processing include skew correction, binarisation, layout analysis, script recognition etc. The segmentation of characters, words, lines etc. are also done in this stage.

Recognition:

In this stage, the text in the image is recognized and the image is converted into editable text format. Image features like HOG,SIFT, profile based features etc. are used for recognition. A classifier then assigns the glyph in the image to the closest matching glyph in the language. Popular classifiers used for this purpose include nearest neighbour, SVM and neural networks etc.

Post-processing:

This is the stage where the errors made in the recognition phase are identified and corrected. The two stages in post-processing are error detection and error correction. The knowledge of grammar of the language and co-occurrence frequencies can help in this process. For example, if the word ”Hello” is recognized as ”He110”, the occurrence of the former word sequence has more chance of existing together than the latter. Statistical language models are widely used to obtain this co-occurrence probabilities of words or characters in words. This information along with the information about the characters which can be confused with each other can help us identify the mistakes and correct them.

Extended Post Processing

After scanning the script, the input may not be in the desired format so as to its dimensions, size, font or paraphrase. This shall cause an error which must be aided with further alterations. To manifest the required output, we may inculcate various error detection and correction techniques. On the prototype segment, it is yet to be decided on what type of mechanism to adopt for error rectification.

It is possible through morphological analysis, which roots back to the structure of formation of a word in relevance to the grammar. Hence, we can strip the suffix and prefix parts to obtain a root word to check whether the correct suffix is present in the word through its morphology. In case of an incorrect suffix, we can provide some suggestions to the detected error.

To enable the machine to provide accurate suggestions, it is necessary to have a dictionary which consists of commonly used words. By partitioning the n-grams of the word, we can track the error causing word and replace it with a word from the dictionary, which has the nearest hamming distance. For this purpose, Tries are the most suitable and adapt to the purpose of the rectification.

CONCLUSION

OCR engines have been developed into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, check OCR, legal billing document OCR.

They can be used for many purposes like data entry for business documents, e.g.check, passport, invoice, bank statement and receipt, Automatic number plate recognition, In airports, for passport recognition and information extraction, Automatic insurance documents key information extraction[, Traffic sign Recognition, Extracting business card information into a contact list, More quickly make textual versions of printed documents, e.g. book scanning etcMake electronic images of printed documents searchable, e.g.Google Books, Converting handwriting in real time to control a computer (pen computing). Hence it's extremely vital that the entire process is efficiently automated and enhanced for better outputs.

Keeping this objective in mind, an attempt for a fast, efficient and robust method is made in this paper for achieving better recognition rates for handwritten Kannada script which is based on the concepts of Unicode and N-Grams. It is based on an image compression algorithm which generates a unique vector which helps to identify each symbol in the script. The result was considerably high in terms of recognition rate. Our future work aims to improve the tesseract code to achieve still better recognition The proposed method can also be attempted to extend for recognition of symbols of other Indic scripts.

FUTURE ENHANCEMENTS

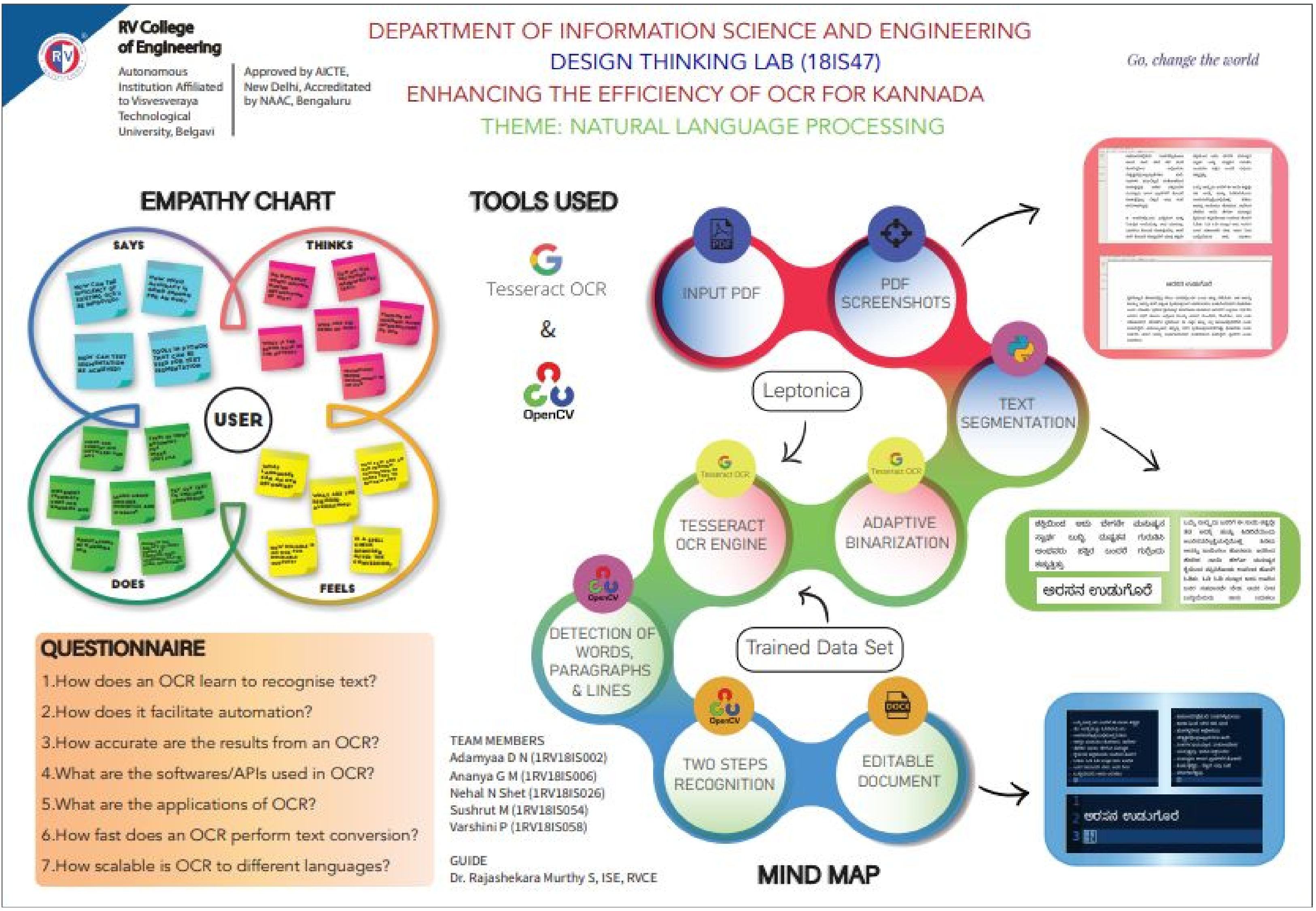
What does the future hold for OCR? Given enough entrepreneurial designers and sufficient research and development dollars, OCR can become a powerful tool for future data entry applications. However, the limited availability of funds in a capital-short environment could restrict the growth of this technology. But, given the proper impetus and encouragement, a lot of benefits can be provided by the OCR system. They are:-

1. The automated entry of data by OCR is one of the most attractive, labor reducing technology .
2. The recognition of new font characters by the system is very easy and quick.
3. We can edit the information of the documents more conveniently and we can reuse the edited information as and when required.
4. The extension to software other than editing and searching is a topic for future works. The Grid infrastructure used in the implementation of Optical Character Recognition system can be efficiently used to speed up the translation of image based documents into structured documents that are currently easy to discover, search and process.

The Optical Character Recognition software can be enhanced in the future in different kinds of ways such as:

1. Training and recognition speeds can be increased greater and greater by making it more user-friendly.
2. Many applications exist where it would be desirable to read handwritten entries. Reading handwriting is a very difficult task considering the diversities that exist in ordinary penmanship. However, there’s constant effort being made to make sure that progress is being made in the field.

DIGITAL POSTER



RV College of Engineering

Autonomous Institution Affiliated to Visvesveraya Technological University, Belgavi

Approved by AICTE, New Delhi, Accreditated by NAAC, Bengaluru

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DESIGN THINKING LAB (18IS47)

ENHANCING THE EFFICIENCY OF OCR FOR KANNADA

THEME: NATURAL LANGUAGE PROCESSING

Go, change the world

QUESTIONNAIRE

1.How does an OCR learn to recognise text?

2.How does it facilitate automation?

3.How accurate are the results from an OCR?

4.What are the softwares/APIs used in OCR?

5.What are the applications of OCR?

6.How fast does an OCR perform text conversion?

7.How scalable is OCR to different languages?

TEAM MEMBERS
Adamyaa D N (1RV18IS002)
Ananya G M (1RV18IS006)
Nehal N Shet (1RV18IS026)
Sushrut M (1RV18IS054)
Varshini P (1RV18IS058)

GUIDE
Dr. Rajashekara Murthy S, ISE, RVCE

REFERENCES

1. https://ieeexplore.ieee.org/document/4787744
2. A comprehensive survey on OCR techniques for Kannada script by Dr.Chandrakala H T and Dr Thippeswamy G
   https://www.google.com/url?sa=t&source=web&rct=j&url=https://pdfs.semanticscholar.org/75d2/a12faa64a69430db05b82b2c000832aeb167.pdf&ved=2ahUKEwje5pPx9aPpAhUTyjgGHeOZDHwQFjABegQIAxAB&usg=AOvVaw0Bp0EckZRsszRKpl8sUmz_&cshid=1588929091103
3. https://en.wikipedia.org/wiki/Optical_character_recognition
4. https://medium.com/cashify-engineering/improve-accuracy-of-ocr-using-image-preprocessing-8df29ec3a033
5. https://cvit.iiit.ac.in/research/thesis/thesis-students/error-detection-and-correction-in-indic-ocrs
6. http://www.lisindia.net/Kannada/Kan_hist.html
7. https://medium.com/@arthurflor23/text-segmentation-b32503ef2613