Summer 8-3-2018

An IoT System for Converting Handwritten Text to Editable Format via Gesture Recognition

Nidhi patel

An IoT System for Converting Handwritten Text to Editable Format via Gesture Recognition

Master's Thesis

By

Nidhi Patel
MSCS Student
Kennesaw State University
Department of Computer Science

In fulfillment of the Requirements for the Degree of

Master of Science in Computer Science

May 2018

An IoT System for Converting Handwritten Text to Editable Format via Gesture Recognition

This thesis is approved for recommendation to the Graduate Council.

Jing (Selena) He
Thesis Advisor, Committee Member

Chih-Cheng Hung
Professor, Committee Member

Mingon Kang
Professor, Committee Member

DEDICATION

To almighty lord: Shiv , Saibaba

To my mother: Bina Patel

To my brothers: Riken Patel and Yogi Patel

The loving memory of my father
(who dreamt for this): Jayanti Bhai Patel

To all my benevolent and supporters

ACKNOWLEDGEMENTS

With all respect, I would like to thank my thesis advisor Dr. Selena He, for her guidance, motivation and mentorship. Her inspirational quote “Conquer the problems, do not let the problems conquer you” has given me strength to withstand all my challenges in this new environment. Thanks for being patient with me also handling me with being calm and compose. I am also thankful to my committee members for sparing their precious time. Finally, I am so grateful to Kennesaw State University for helping me financially to do my research work.

List of Figures

List of Tables

ABSTRACT

Evaluation of traditional classroom has led to electronic classroom i.e. e-learning. Growth of traditional classroom doesn’t stop at e-learning or distance learning. Next step to electronic classroom is a smart classroom. Most popular features of electronic classroom is capturing video/photos of lecture content and extracting handwriting for note-taking. Numerous techniques have been implemented in order to extract handwriting from video/photo of the lecture but still the deficiency of few techniques can be resolved, and which can turn electronic classroom into smart classroom.

In this thesis, we present a real-time IoT system to convert handwritten text into editable format by implementing hand gesture recognition (HGR) with Raspberry Pi and camera. Hand Gesture Recognition (HGR) is built using edge detection algorithm and HGR is used in this system to reduce computational complexity of previous systems i.e. removal of redundant images and lecture’s body from image, recollecting text from previous images to fill area from where lecture’s body has been removed. Raspberry Pi is used to retrieve, perceive HGR and to build a smart classroom based on IoT. Handwritten images are converted into editable format by using OpenCV and machine learning algorithms. In text conversion, recognition of uppercase and lowercase alphabets, numbers, special characters, mathematical symbols, equations, graphs and figures are included with recognition of word, lines, blocks, and paragraphs. With the help of Raspberry Pi and IoT, the editable format of lecture notes is given to students via desktop application which helps students to edit notes and images according to their necessity.

INDEX

CHAPTER 1

INTRODUCTION

Introduction chapter contains the concise presentation of the proposed system, description of motivation to build this system, problem statement and research methodology of the system.

1.1 Introduction

In this world of technology, traditional class method is rarely being utilized as most of the lectures are being taught through Power Point slides (PPTs). Job of lectures has been soothed aged ago since when PPTs came into pictures and students stressed has been reduced to recollect what was explained in class since world wide web has come into picture. With the help of PPTs, video capturing, internet and websites job of lecturer and students has got relief as electronic classroom came in picture. Electronic classroom provides all the class materials, homework, class video recordings are posted on websites. Form here the growth of classroom technologies stared evolving electronic which led to smart classrooms.

With the advent of the technology, to take notes of lectures needs much less work since the recording and synchronization are all automated. Many useful tools for improving accessibility of lectures have been established. With the help of that tools lectures are now termed as “Multimedia lectures” as students can refer to lecture in the form of video, audio and text.

Advances have been connected to naturally translate educator's lecture and process the interpretation to secure close verbatim lecture transcripts for students [47], [45], [46]. The advantages of creating lecture transcripts have appeared to improve both learning and educating. Students could compensate for missed lecture and additionally to verify the exactness of their own notes amid the lectures they took. Combined with a recorded sound/video lecture track and duplicates of the lecture slides, students could reproduce the lecture material for imitating the lecture at their own learning pace. These lecture transcripts and extra sight and sound chronicles likewise empower educators to audit their own teaching performance and lecture substance to help them to enhance singular instructional method [47].

1.2 Motivation

Doubtlessly, technology has made classroom environment “smart” but these all technology has soothed the job of lecturer. Whether the lecture videos stored and shared via clouds or lectures are taught in class with the help of power point presentation or smart board application, still students need to note down what they have grasped from the lecture. Students got stuck in the middle of taking complete notes and giving careful consideration to the instructor. This implies students invest quite a bit of their time and efforts on taking notes [24], [47], [45], [46]. Most of the time students try to note down entire speech spoken by the teacher but if they fail to write entire speech of instructor then they will definitely try to make brief notes on emphasized or importance sentences spoken by instructor [24]. Hence, by making the process of lecture note automated we can add one more fascinating feature in building of smart classroom. Numerous techniques have been implemented in order to extract handwriting from video/photo of the lecture but still the deficiency of few techniques can be resolved. Existing systems has deficiency like complex computation, high cost to build system, less accuracy of image to text conversion, less efficiency of hardware tools used, fail to convert mathematical formulas, figure, graphs along with entire text blocks, fail to provide editable text notes. Consequently, taking this situation as an inspiration we propose a simplest, cheapest and more accurate framework that helps students to get lecture notes in editable format, which will be useful to students to sort out, condense, and better understand lecture data, recording content for later contemplating, self-directed learning through the dynamic procedure of note taking, and essentially remaining mindful amid class. Moreover, the proposed framework will assist numerous students with inabilities that can't compose lecture notes without human help.

In this thesis, we present an Internet of Things (IoT) system to convert handwritten text into editable format by implementing hand gesture recognition(HGR) with Raspberry Pi and camera. Hand Gesture Recognition (HGR) is built using edge detection algorithm and HGR is used in this system to reduce computational complexity of previous systems i.e. removal of redundant images and lecture's body from image, recollecting text from previous images to fill area from where lecture's body has been removed. Raspberry Pi is used to retrieve, perceive HGR and to build a smart classroom based on IoT. Handwritten images are converted into editable format by using OpenCV and machine learning algorithms. In text conversion, recognition of uppercase and lowercase alphabets, numbers, special characters, mathematical symbols, equations, graphs and figures are included with recognition of word, lines, blocks, and paragraphs. With the help of Raspberry Pi and IoT, the editable format of lecture notes is given to students via desktop application which helps students to edit notes and images according to their necessity.

1.3 Problem Statement

Motivated by converting traditional handwriting notes into editable electronic notes which can be shared among students, we propose to design and implement the following system:

Given the photos of the recorded handwriting to Raspberry Pi as an input, Raspberry Pi can convert them into editable electronic notes, save them on the private clouds, and share among students via desktop application. To be specific, we propose to use static hand gesture as a signal to Raspberry Pi to take an image of blackboard. By using Optical Character Recognition(OCR) the taken images will be converted into editable text files and also will be uploaded on a private cloud. Moreover, a desktop application is built to provide electronic notes to the students, where they can download notes, upload their own edition of electronic notes into group forum.

1.4 Research methodology

The steps mentioned below has been performed to pursue research:

1. Perform comprehensive literature research on gesture reorganization,
2. Perform comprehensive literature research on converting natural hand writing images into text,
3. Identify weaknesses in existing system to get lecture notes and find better approaches for proposed system to work efficiently,
4. Explore different image to text conversion algorithms and choose the most optimal image to text conversion technique using machine learning,
5. Explore the technique to capture static hand gestures through Raspberry Pi and Raspberry Pi Camera Module,
6. Build the desktop application,
7. Assemble the system, solder the components using IoT concept,
8. Evaluate the proposed framework for performance, accuracy and efficiency.

1.5 Contribution

Each of the issues described in the problem statement are thoroughly addressed. The contribution of this thesis is as follows:

1. With scrutiny, the static hand gesture recognition is implemented. A new approach and term has been defined to improve deficiency of existing algorithm of hand gesture

recognition. Our new method helps in recognizing all 31 combinations of hand fingers gestures with clear differentiation among “OK” gesture, 3 fingers gesture, clear background, one finger, fist and thumb. This gesture recognition has been implemented on continuous video streaming.

2. For image to text conversion deep machine learning algorithm have been used. A convolutional neural network has been designed, trained and tested by our own to recognize Letters (A to Z, a to z), numbers (0 to 9), special characters (&, *, ^ etc.), with recognition of blocks, lines, paragraphs and words. For figures (ex: mathematical figures), mathematical formulas and graphs, we have used image processing technique by using Open CV and tesseract.
3. The extracted text and equations are letter appended to editable text format. This editable text format is called as data files in our system. Providing this data file to students in their desktop application is as a totally new approach that we have introduced to distinguish our system form existing system.
4. To make cheapest and efficient IoT based smart classroom, Raspberry Pi 3B has been used. Raspberry Pi, student’s computer and google drive have been connected to one internet network. Hence, “input images” and editable format of “data file” will be stored in google drive via Raspberry Pi so that all the student in class can get input images and data files into their desktop application.
5. A desktop application will be built for students where they get input images and data files within fraction of time in class itself and they can alter and store according to their individual preferences without losing concentration towards lecturer’s explanation.
6. The accuracy of hand gesture recognition, accuracy of handwritten images to text

conversion, accuracy and efficacy of entire system has been improved in comparison with existing systems.

1.6 Organization of Thesis

In this thesis, chapter 2 presents the literature review. Chapter 3 presents the architecture of system and technical challenges encountered during designing the system. Chapter 4 presents the implementation of the system. Chapter 5 provides evaluation and results of the system followed by conclusion and future work.

CHAPTER 2

LITRATURE REVIEW

This chapter provides literature review. The sub section of this chapter gives literature review on hand gesture recognition and handwritten image to text conversion.

2.1 Hand Gesture Recognition

For 2D and 3D hand gesture recognition there are numerous methods and techniques available. In these numerous techniques there few frequently used techniques are: vision based approaches, depth based approaches, instrumented(data) glove approaches, colored markers approaches and convexity depth approaches. To see in more detail about all the 2D in 3D hand gesture recognition techniques, method and tools refer our own survey paper [1]. The subsection 2.1.1 shows the literature review of hand gesture recognition techniques used by different researchers and the subsection 2.1.2 shows the deficiency existing method of hand gesture recognition and what we are doing to remove that deficiency of existing methods.

2.1.1 Literature review on Hand gesture recognition

In paper [57] they used Fusing frame images approach. Hierarchical identification model is used for recognition. But this approach is only able to find rough hand gesture recognition. And the accuracy of this approach is 90%. In [57] SVM classifier is used and the accuracy of this approach is 99.2%. In [59] they used Hidden Markov model (HMM) technique and they also include machine learning in that. By applying some modification in normal system, they got 93.7% correct recognition.

In [35] images are captured to communicate between human and computers. In vision based approach only camera(s) are required to capture images no requirement for other devices. Digital cameras are used to capture hand images. Captured images are further processed and analyzed by using the vision based techniques. Different kind of cameras can be used like monocular, fish eye, time of flight and infrared cameras etc. [35].

In paper [34], researchers have used vision baes technique, through vision based techniques, recognition of the represented alphabets and numbers are being so easy. This approach is very easy, natural and well-situated to use so this approach is very popular [34]. Here human and computer interact directly. In contrast, many gesture challenges [28] are raised. Variations of lights, other object which are of skin color, illumination changes, complex background are such challenges in this approach. Besides that, recognition time, robustness, velocity and computational efficiency [27] are more challenging task here.

In papers [28] [29] [30], for capturing hand position and motion sensor devices are used in instrumented data glove approach. Numerous kinds of sensors are existing like mount bases, multi-touch screen and vision based sensors. Coordinates which are exactly provide palm and finger's location; orientation and configuration are perfectly calculated in this approach [28] [29] [30]. Reaction speed is fast and it has high accuracy rate. Main requirement of this approach [30] is physically the user must be connected with computer. This limitation is barrier to interaction between users and computers. These devices are very costly [30] and not much efficient for virtual reality.

In paper [31], to locate palm and fingers and to direct tracking process, human hand is worn with some colors in colored marker approach [31]. For forming hand shape necessary geometric features are extracted using this functionality [31]. The color glove shape has small regions with different colors. Three different colors are used to represent the finger and palms as applied in [32] where a wool grove is used. It's very easy to use and quite cheap compare to instrumented data glove [32]. Still human computer interaction is not that much natural in this technology [31]. That's the main limitation of this approach.

In paper [62] Statistic learning theory is used with nonlinear SVM and Bag features. This approach is much effective and got accuracy up to 97%. But it has some limitation that it not need segmentation and hand tracking. In [52] distance measurement is done between two hand gestures based on shape and texture. For that they are using Novel distance metric to overcome the partial matching problem. This approach is 98% appropriate for all kind of input data.

In paper [22], the convexity defect approach has been used to identify hand gesture recognition. To recognize hand palm first contour function has been applied then after extracting hand region, convex hull algorithm has been applied to find hand area, 6 convexity defects points are used for hand figure recognition, tightness of hand gesture (δ) has been counted for feature extraction.

In paper [33] also the convexity defect approach has been used to identify hand gesture recognition. In this paper to recognize hand gestures after image acquisition of hand they have applied skin color detection method to acquired images. After that they have used erode and dilation noise removal methods. After noise removal, they have found contours and convex hull foe hand region. After finding convex hull, they have found 7 convexity defects to recognize fingers of the hands.

In paper [44] the convexity defect approach is used to identify hand gesture recognition, same procedure as paper [22], [33] has been used but in paper [44] 5 convexity defect point has used to recognize fingers but the position of convexity defect point is totally different than paper [22].

2.1.2 Our approach for hand gesture recognition

We are using convexity defect points method to recognize fingers as paper [22], [33], [44] but our convexity defect point is totally different form theirs , we are using area ratio and hand area to clearly distinguish among “OK” gesture , 3 finger gesture , thumb , one finger, fist , clear background with 100% accuracy of recognition.

2.2 Image to text conversion

2.2.1 Literature review of Image to text conversion

In paper [81], Singh, Gurpreet, and Manoj Sachan have presented a technique for the recognition of isolated offline handwritten characters of Gurmukhi script. They have presented a technique based on Multi-Layer Perceptron (MLP) Neural Network model as it uses generalized delta learning rules and easily gets trained in less number of iterations. The proposed method in this paper detect graphical symbols by identifying lines and characters from the image. The performance rate is upto 98.96 %.

In paper [83], Meng, Hann, and Daniel Morariu has presented Khmer Character Recognition (KCR) system. The system is integration of two artificial neural network techniques to work together namely self-organization map (SOM) and multilayer perceptron. They have used backpropagation algorithm to make the machine learn. The system is being implemented in matlab environment. The average recognition of trained dataset resulted 65% of correct predictions and untrained dataset is approximately only 30% correct prediction with noise rate. The KCR system is expected to be able to work with offline handwritten Khmer character.

In paper [84], Fanany, Mohamad Ivan proposed a workflow and a machine learning model for recognizing handwritten characters on form document. The learning model is based on Convolutional Neural Network (CNN) as a powerful feature extraction and Linear Support Vector Machines (SVM) as a high-end classifier. The proposed method is more efficient than modifying the CNN with complex architecture. The recognition rate achieved by the proposed method are 98.85% on numeral characters, 93.05% on uppercase characters, 86.21% on lowercase characters, and 91.37% on the merger of numeral and uppercase characters. The pre-processing, segmentation and character recognition are integrated into one system. The output of the system is converted into an editable text.

In paper [85], Impact of grid based approach in offline handwritten word recognition is experimented. The study is experimented on handwritten word comprising of 28 district names of Karnataka state. The proposed method first divides the input word into four grids and for each grid the popular subspace learning approach i.e., Principal Component Analysis is applied for better representation. For classification purpose, distance measure technique i.e., Euclidean distance is computed. The experiment result revealed that the proposed grid based approach with subspace learning approach outperforms standard PCA approach.

In paper [86], Nair, Pranav P., Ajay James and C. Saravanan have implemented a handwritten Malayalam character recognition system. The proposed method uses CNN to extract and classify Malayalam characters. This method is different from the conventional method that requires handcrafted features that needs to be used for finding features in the text. The classification of 6 malayalam characters is being done. They have stated that both Sample generation and CNN modelling are time consuming tasks and it also requires a CUDA enabled GPU for parallel processing. CNN provides higher accuracy for malayalam characters.

In paper [82], Kajale, Renuka, Soubhik Das, and Paritosh Medhekar has proposed a method for intelligent character recognition using classifiers and transferring the data to excel sheet. For hand written character recognization, they have used supervised machine learning. It gives accurate results even when the dataset becomes extensively large due to multiple language scripts.They have used a dataset of 200 samples for 36 alpha numerals and further 100 for special characters. The result shows upto 95 % accuracy for alpha numerals and special characters.

Vani, M. Shyni Beaulah and R. Deepalakshmi [87] has created an application interface for OCR using artificial neural network as a back end to achieve high accuracy in recognition of characters. They have used isolated character database consisting of English characters, digits and keyboard special characters for training and testing. The system can classify 30 characters per second. The network of connections and weights obtained by back propagation learning is readily implementable on commercial digital signal processing hardware. Preliminary results on alphanumeric characters show that the method can be extended to larger tasks.

In paper [88], Roy, Partha Pratim, Youssouf Chherawala, and Mohamed Cheriet have presented a verification based re-ranking approach using Deep Belief Networks (DBNs) for handwritten word recognition. A recurrent neural network based sequential text recognition system is used at first to provide the N-best recognition hypotheses of word images. Word hypotheses are aligned with the word image to obtain the character boundaries. Then a verification approach using a DBN classifier is performed for each character segments. DBNs are recently proved to be very effective for a variety of machine learning problems. Finally, the N-best recognition hypotheses list is reranked according to the new score.

The numeric representation of a 2D image is known as a Digital image. Advanced pictures go about as a fundamental and key medium for passing on, extricating data and utilizing the data as an essential contribution for machine learning and machine vision applications. Optical Character Recognition (OCR) is the sub-space of Pattern Recognition domain. OCR in early days mainly use for two problems, first was utilized to extend the use of telecommunication and also it was utilized as a guide for the visually impaired. The examination and research of character recognition began in 1870 when C.R Carrey of Massachusetts created the retina scanner which was image transmission framework. Current and progress OCR version did not show up until 1940.

In business and industries, the data processing played most important role during year 1950. At that time punch cards were used to feed data in to computers. Machines were so large and require so many installations. Moreover, their maintenance cost is so high and at the same time they were so slow. To overcome some limitations of that from1960 to 1965 the first-generation OCR system was developed which contain some limited character shapes.
Then second generation came with the system which able to recognize machine and hand printed characters. But it had very limited number of character sets. Systems in middle 1970s represent the third generation OCR system. These was able to overcome the problems of previous era machine but still was very luxurious and limited to marketable use. Today the OCR is used commercially and also available for the people. OCR system been feasible for economical purpose but still the OCR system are not able to match the human reading ability and development is still going on.

Ravina Mithe [11] developed new system where the functionality of Optical Character Recognition and Speech Synthesis are combined. They conclude that environment variation in real photo is affecting the reliability of OCR system. They also elaborate process and usage of this system. Réjean Plamondon [12] done survey on both the modes of OCR called online mode and offline mode. The nature of handwritten language is briefly discussed in this paper. Fred W. M. Stentiford [4] provided deep analysis on feature extraction stage of an OCR application. If we add some tilted as well as unscaled character data at the time of training, the efficiency and flexibility of the system is increased.

Amarjot Singh [5] performed the survey of application of OCR system in different fields. This paper concentrated majorly on three applications of OCR defeating CAPTCHA, storage of data for institutional or business purposes, musical note recognition. Youjie Qiu [7] proposed an algorithm for license plate extraction based on vertical edge detection and morphological operations. License plate is extracted by finding Region of Interest (ROI) in this paper. After that outer shape feature analysis is performed and finding texture feature of the license plate.

Patrik Berggren [8] studied and researched various Sudoku solving algorithms in their thesis. This thesis considered and evaluated three algorithms: Backtracking, Advance Backtracking with rule based solving technique and Boltzmann machine. The thesis also analyzes the difficulty rating of Sudoku and ways to generate a Sudoku Puzzle.

Researcher always wants to make the efficient and robust system for recognition just because of that OCR system matters a lot for them. They always concentrate on one area like normal OCR or segmentation, though the task of focusing on Intelligent Character Recognition (ICR) [21] is too tough. Dynamic template creation and matching is requiring for that. They have to convert character to text very accurately. A texture character of microscopic image which is related to roughness of surface is discussed and various approaches for that is also mentioned in [25] there. According to surface roughness various parameters have been change regularly. They also perform so many efforts to capture characters from paper forms.

It is very difficult to develop a system in which recognize character automatically transfer to excel sheet. Most important phase in character recognition is Segmentation. So many efforts are made to improve accuracy of this phase. Segmentation is discussed in [28] in which they used trained classifier to segment the connected characters in text-based CAPTCHAs. Modified form of SVM [30] in which freedom chain is used is discussed and features are extracted using this SVM in [30].

Krishnamurthy [29] and the team members are discussing the hybrid approach with both the classifier (supervised and unsupervised). They used fuzzy k-Nearest Neighbor and fuzzy c-means as base classifiers for individual feature sets in this hybrid approach. Final feature vector for the k-NN classifier is formed using this both classifiers.

One of the best deep learning architecture is Convolutional Neural Network (CNN). Currently, CNN is become a state of the art of handwriting characters recognition. Modified CNN is used in [26] with two training feedbacks, one is reconstruction feedback and second is classification feedback. They got 99.59% accuracy on MNIST dataset.

Elleuch [23] combines a CNN with another end classifier. For recognizing hand-written Arabic character, the kernel SVM is used as an end classifier. Hey slightly modify the architecture of CNN and also the SVM approach. SVM is better just because it has good generalization ability than neural network on standard CNN. Neural network is working with the Empirical Risk Minimization (ERM) principle while SVM working with Structural Risk Minimization (SRM) principle.

Some learning changes the CNN with a variety of advancement to increasing the accuracy rate and the practice time. Some of them change the input data to get better the accuracy rate of CNN method. Mega et al. [7] have joint GLCM (Gray- Level Co-occurrence Matrix) and CNN for cattle classification. Feature Extraction uses a GLCM method to extract contrast, energy, and homogeneity feature of the image and it is used as input of CNN. Kwolek [8] has changed the architecture of CNN by merging CNN and Gabor filter to detect the facial region. The proposed method gives classification performance better than original CNN.

2.2.2 Our approach for image to text conversion.

We have built, trained and tested our Convolutional Neural Network for classification of handwritten images.

2.3 Existing Systems for transcribing lecture notes

There have been numerous researches performed to provide lecture content to students in many different formats such as audio, video and audio to text, lecture images to text. Section 2.3.1 describes various lecture transcribing lecture notes.

2.3.1 Existing system

F. H. Yeh and his team [8] introduce robust handwriting extraction and lecture video summarization technique. Researchers have implemented a system to extract the handwriting from blackboard. This system efficiently extracts handwritings from blackboard even if there is a variation of light in classroom or instructor stands in front of blackboard. The system summarizes and indexes video so that videos can be used for e-learning application. To locate the boundary of the blackboard, K-mean segmentation has been used in each frame of video. The image in lecture videos are convert from RGB color space to CIELAB color space (L* a* b*) color space. For blackboard, two-color channels (a* and b*) have been used. Connected component technique has been applied to extract lecture's body.

The adaptive thresholding has been used to extract handwriting from the blackboard. Time sequence information from video has been used to extract accurate handwriting and to remove noise from the images. Some disadvantage of this system is, they have to remove lectures body in order to get proper result of handwriting extraction, they have to check when the lecturer erased entire board and new writing started. Moreover, there will be many handwriting images for same theory until and unless that theory is not erased. So, many redundant images will be generated for same theory. Only the last image which is captured before erasing the board will contain entire theory. Hence, except last picture all the captured pictures will be in vain. The images of extracted handwriting are not in editable format.

Markus Wienecke, Gernot A. Fink and Gerhard Sagerer [9] proposed new approach towards automatic video-based whiteboard reading. This paper has mentioned all the technical challenges of handwriting extraction. They have also introduced that how to recognize lines, blocks, paragraph from the image. This paper has procedure steps for handwriting conversion as: Extracting Text Regions, Preprocessing, Feature Extraction, and Statistical Modeling & Recognition. Character error rate is of 29.5%. The accuracy of text conversions is low. Conversation of mathematical formula is not mentioned. Appropriate results of image to text conversion have not been shown so that one cannot judge how accurately the system works.

Zhengyou Zhang and Li-wei He [7] proposed new system for note taking with a camera: with whiteboard scanning and image enhancement. In this paper they use digital camera and scan the entire white board and then enhances the images. They have made a user interface where they give images to students. After that students are free to take photo of that. Boundary of whiteboard is automatically located and region belongs to whiteboard are crops out. That region is rectifying into a rectangle, and corrects the color to make the whiteboard completely white. They used robust technique to stitch multiple overlapping image if single image is not enough to get information.

Ali Shariq Imran, Faouzi Alaya Cheikh [6] introduce lecture content classification tool. In that paper researchers have detected figures, formulas and text from the handwritten text of the blackboard. In this paper researchers only can recognize which content is text, which is figure and which is formula. They cannot convert this recognized text, figures, formulas into electronic editable text format. They have also made user interface tool which can only extract figures, text and formulas form the handwritten content of blackboard.

Seiji Okuni and his team member [5] proposed robust approach for video Scene Segmentation Using the State Recognition of Blackboard for Blended Learning. In this paper, unnecessary video content are removed by researcher from the entire lecture video. Lecturer behavior is recognized such as up down movement of hands. That is considering as the sign of content segmentation for students. Event time is noted down and using that segmentation is done from lecture movie to some shots. They done experiment on various real lectures and got 97% accurate segmentation.

Marcus LIWICKI and Horst BUNKE [4] introduced new concept for handwriting Recognition of Whiteboard Notes. They used eBeam interface based on infrared sensing. Rather than using video camera, this system is much easier to use. Along with that this system is less vulnerable to problems which occurring due to poor lighting conditions, self-occlusion and low image resolution.

Szil´ard Vajda [2] introduces a prototype model for recognizing and visualizing mind maps which are written on white board. This approach acquire image by a camera. Next, they performed binarization and extraction of connected component is done. Without conceited about any prior information about the document, its style, layout, etc., the examination starts with connected components. Then labelling is applying as text, lines, circles or arrows according to trained classifier of neural network based on features. Detection of word is done after identified the patches of text. According to gravity centre of text it was modelled. Density based clustering is used to group them into possible words. Hidden Markov Model (HMM) recognizer is applying to recognize grouped connected components. Average recognition rate of neural network for different connected components are 95.7%.

Tiecheng Liu and Chekuri Choudary [3] proposed a new technique to extract and summarize the textual content of teaching videos for recognizing hand writing, provide indexing and some other applications related with e-learning. For indexing and retrieval most, suitable elements are the characters and figures which are written on boards. Video summarization is used to extract small set of key frames. One frame is only process per every 150 frames because instructional videos are highly redundant in content. To partition instructional video frames into over-segmented regions, mean shift segmentation is used. Largest region is finding from that frame. Distribution parameters are estimated for largest frame regions. Total numbers of pixels are calculated and using some proper threshold value two regions is merged. Luminance (L) and color components (a and b) are treats separately in this model.

Video key frames are extracted after analyzing the content fluctuation curve. Local maxima of that curve are located through shifting window algorithm. Using that candidate key frame is extracted. Further elimination is done based on redundant frames by applying matching the text in the candidate frames based on the Kth Hausdorff distance between content pixels. Experimentation is performed on three instructional videos which are recorded in real classrooms. According to that results they conclude that, proposed algorithm is highly effective in extracting and summarizing the content of instructional videos.

Nael Hirzallah [11] introduced new alternative solution to e-boards. Extraction of lecture notes which are added or removed by lecturer is implemented using web camera and video processing. Results of this algorithm are demonstrated under two different scenarios. One is adding soft notes presented on projector through lecturer and the second one is adding text which are displayed on white board which are located behind that projector. Brightness variation is the normal effect occurs during the transition of slides. This was taken as consideration in first scenario of this algorithm.

In second scenario movement of lecturer in front of white board is taken in consideration. Snapshot is taken when lecturer is added or removed notes. However, the lecturer is considered as an absent in the snapshot. First image is saved as layer zero. In next step one can only take the difference between successive snapshots. Successive information is represented by the addition of the layer two, three and so on. To diminish unhelpful action of this algorithm, an effort was made to eliminate the brightness section and work on the IQ part of the YIQ colour space. Main advantages of using this algorithm is that, it was applicable for both with and without slide show as well as white and black boards. Solution given in this paper was much cheaper as $15 worth of a PC camera and simpler to install if match up with that of the e-boards.

2.3.2 Our proposed system

Existing systems has deficiency like complex computation, high cost to build system, less accuracy of image to text conversion, less efficiency of hardware tools used, fail to convert mathematical formulas, figure, graphs along with entire text blocks, fail to provide editable text notes to students. Hence, we designed a system to improve all these deficiency and provide high accuracy and efficiency. The architecture of proposed system is described in next chapter.

CHAPTER 3

ARCHITECTURE OF SYSTEM & CHALLANGES

This chapter will give thorough description of the architecture of system, technical challenges encountered while constructing the system.

The first step of the proposed system is capturing hand gesture. The hand gesture will be captured and perceived by Raspberry Pi (see figure 3.1 & 3.2). After the gesture is perceived by Raspberry Pi, the picture of blackboard will be taken by camera attached with Raspberry Pi to avoid capturing unnecessary images of board. The conversion of handwritten images to text is done via machine learning programs installed in Raspberry Pi. The hand-written images captured by camera is called as input images and the text file generated after extracting text from images is called as data files. The input images and data files are sent to google drive via Raspberry Pi. Students will have a desktop application installed in their laptops in order to receive the input images and data file in to their laptops or PC.

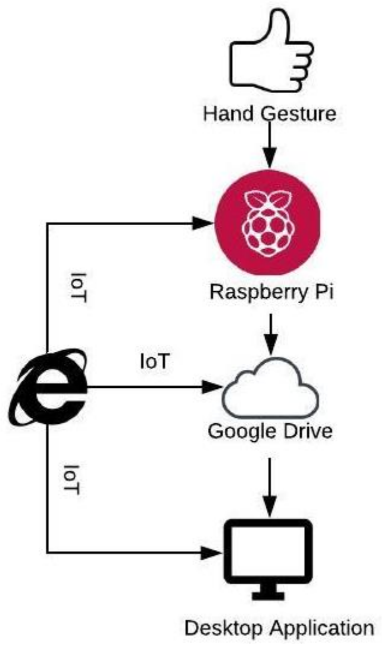
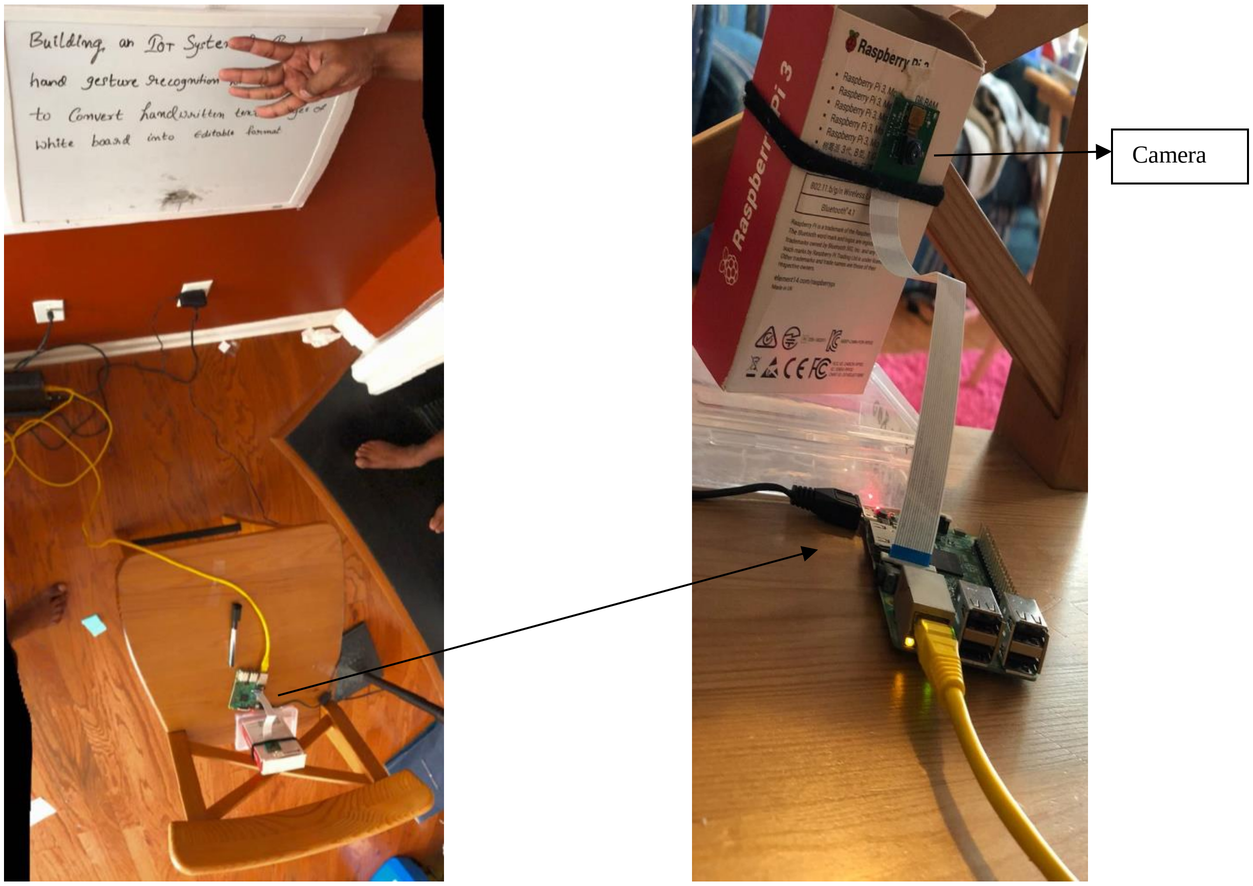


Fig 3.1: Architecture of Proposed System

Figure 3.1 depicts the system architecture. The detail description of all the system component is described in sub section of this chapter.



(a) (b)

Fig 3.2(a) : Hand gesture

Fig 3.2(b) Raspberry Pi

Figure 3.2: Visual representation of system architecture

Figure 3.2 shows visual representation of system architecture. Figure 3.2(a) shows that how a lecture should give Hand gestures to a Raspberry Pi camera. Also, Figure 3.2(a) shows the arrangement of Raspberry Pi to capture a gesture. Figure 3.2(b) shows the visual representation of Raspberry Pi and attachment of camera in Raspberry Pi.

All the subsection mentioned below gives thorough description of all components of system architecture.

3.1 Gesture Recognition:

3.1.1 Motivation to use Gesture Recognition

The major deficiencies of previous systems are computational complexity and delivering editable format of lecture notes. For efficient handwriting extraction, systems [3], [8] had performed few complex techniques in order to remove redundant and unwanted images/video content, remove lecture's body form images, recollecting text from previous images which has been missed from the area where lecture's body has been removed. Figures 3 and 4 clearly describes the complexity issue of systems [3], [8] .

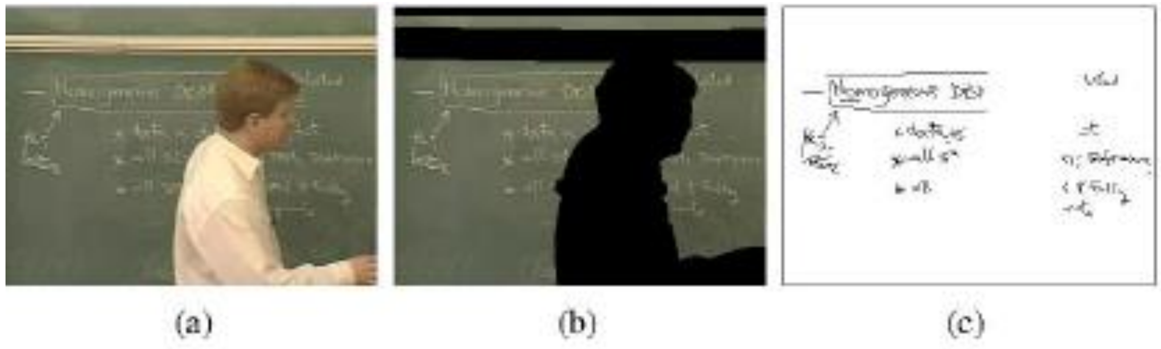


Fig 3.3(a): Lecture's body hides few text of board [8].

Fig 3.3(b): Locating Lecture's body and removing it [8].

Fig 3.3(c): Few text is missed because of lecture's body removal [8].

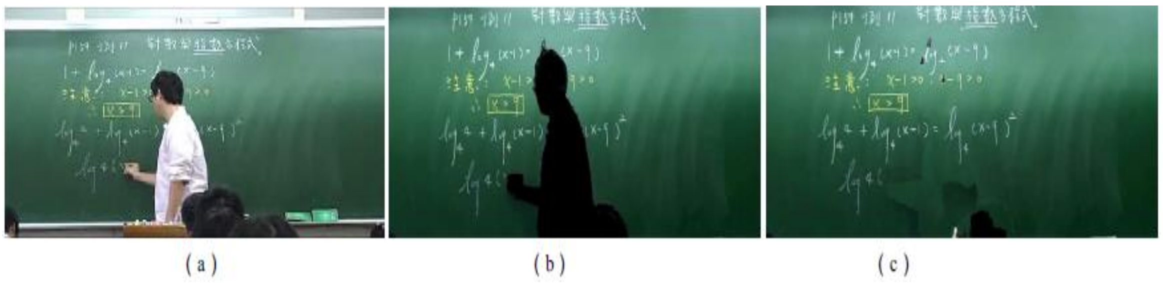


Fig 3.4(a): Lecture's body hides few text of board [3].

Fig 3.4(b): Locating lecture's body in removing it [3].

Fig 3.4(c): Text is recollected which is missed after lecture's body removal [3].

Figure 3 shows that lecture’s body is hiding text of the blackboard. When the lectures’ body is removed few text is also missing. Figure 4 depicts the same situation. The only difference between figure 3.3 and 3.4 is “text re-collection”. In figure 3.3(c) few text is missing from where lecture’s body is missing but in Figure 3.4(c) the missing text because of lecturer body is regained.

In system [2], [8], [5] the time frame has been set in camera to take picture or videos i.e. 1 minute. As per time frame, camera will take pictures at every 1 minute. Hence, there is a possibility of having text and lecture’s body or only text or only lecture’s body in image. Also, if professor is explaining some content and he has not written anything on board till 5 minutes then there is possibly of having 5 same pictures since camera takes picture at every 1 minute. This scenario clearly shows that researches have to do more complex computation in order to remove lecture’s body and redundant images. Alternative and very simple solution to reduce this complexity is to use hand gesture recognition. Whenever lecturer is done writing and explaining entire theorem then he can show a hand gesture. So that, camera will take picture after recognizing valid gesture. Hence, gesture recognition will remove most of the computation like: locating lecture’s body in image, removing lecture’s body from image, capturing redundant images, removing redundancy of images, sorting out important content of lecture and removing unwanted content, recollecting text from previous images in order to fill area from where lecture’s body has been removed.

3.1.2 Methodology used for gesture recognition

The main motive to use hand gesture in our system is to remove computation that existing system do in order to remove lecture’s body from image. Hence, we have used hand gesture recognition to reduce this computational complexity. This subsection clearly mentions what methodology we have used in order to remove the computational complexity.

1. The Raspberry Pi and its camera is kept exactly in front of the board (see figure 3.5). So that lecturer can show the gesture to Raspberry Pi.

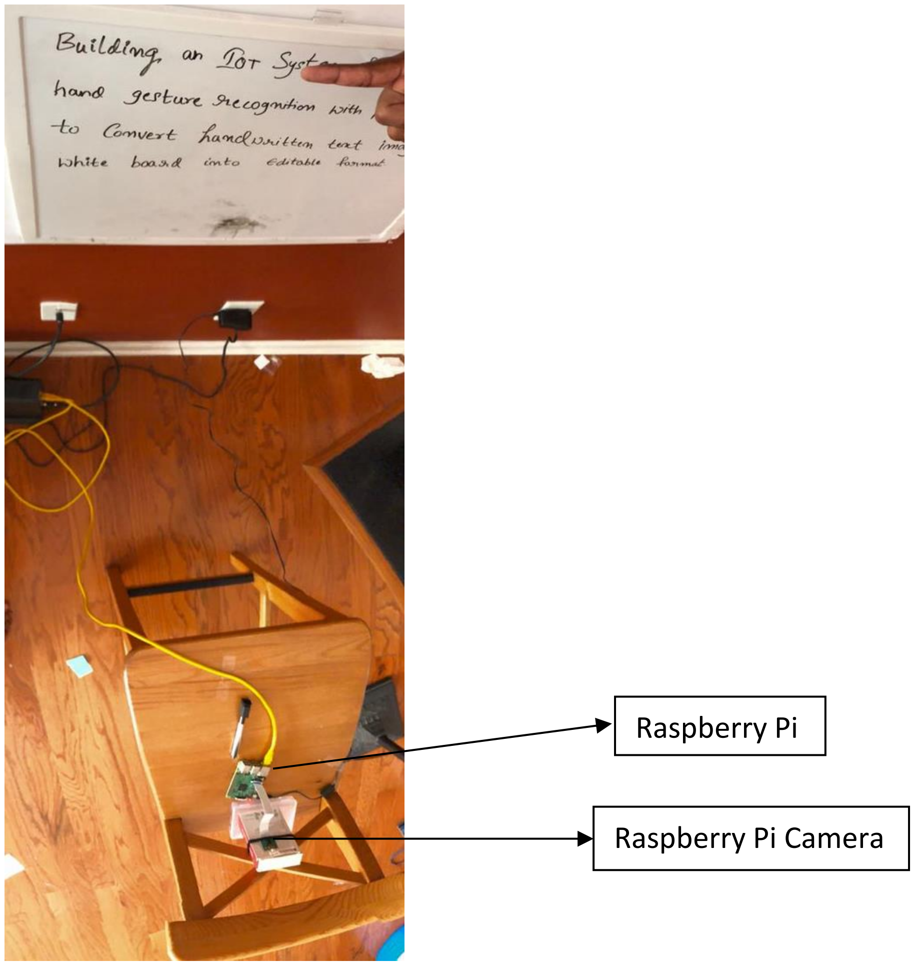
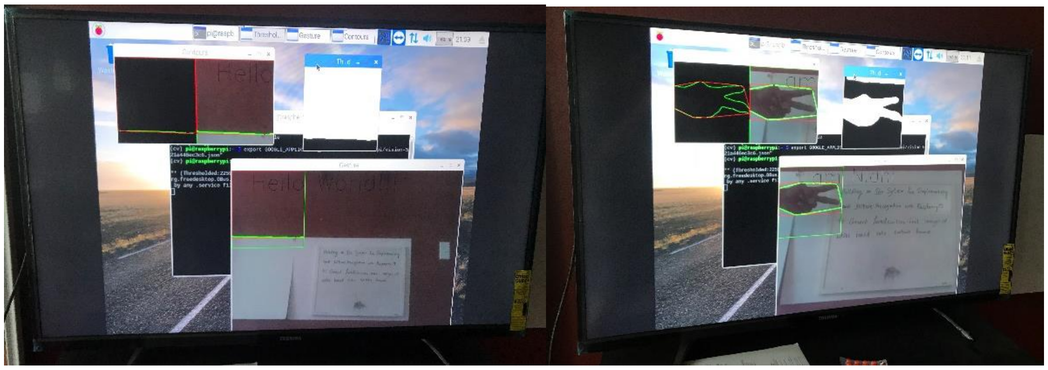


Fig 3.5: Arrangement of board and Raspberry Pi

Figure 3.5 shows the arrangement of board and Raspberry Pi. The Distance between Raspberry and board has been reduced in order to take proper picture. Picture has been taken from top view so that we can clearly see Raspberry Pi and camera.

2. To capture gesture, the video streaming of Raspberry Pi camera is utilized. The contours are drawn on video streaming (see figure 3.6). The contours drawn on video streaming will look for a gesture to arrive in a contour box. The moment gestures are captured, the video

streaming will close and camera will turn on to capture photo of board. The detailed description of drawing contours and capturing gesture is described in chapter 4.



(a) (b)

Fig 3.6(a) : Contours on video streaming

Fig 3.6(b): Hand gesture on countour box

Figure 3.6 shows the contours drwan on video streaming and hand gesture captured in countour box.

3.1.3 Challenges of Gesture Recognition

Hand gesture recognition has several technical challenges while implementing. The challenges we mainly faced in order to implement hand gesture recognition are mentioned below. The recovery of challenges has been thoroughly described in chapter 4.

1. **Shadow of Hand**

When there is a poor lightning or if lecturer is standing opposite to light than we can have shadow of hand (see figure 3.7). While perceiving gestures if we have not programmed the hand recognition properly then there is a possibility of perceiving shadow as a second hand. Hence, according to attached figure below, poor hand gesture recognition program can perceive 8 fingers including shadow. Shadow may lead to inaccurate hand gesture recognition.

To remove this hand shadow, we are using blurring technique. Detail description of hand gesture recognition implementation is described in chapter 4.

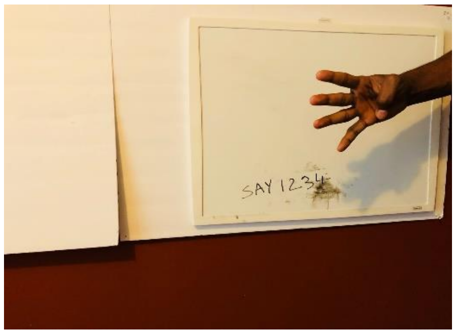


Fig 3.7: Hand Shadow

Figure 3.7 shows the hand and its shadow when there is poor lighting.

2. Improper Lighting

Lightning is a very important factor while in computer vision. If we do not have proper light on object that we want to detect then we may end up detecting object with lowest accuracy (see figure 3.8). Also, pre-processing steps to reduce noise will be increased in order to detect object with proper features. Hence, lightning plays very crucial role in this system in order to detect object properly. Chapter 4 describes all pre-processing steps used in our system to detect hand very accurately even though there is poor lightning.

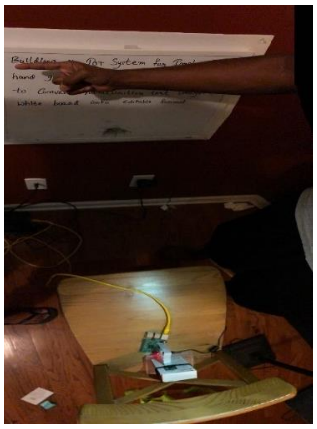
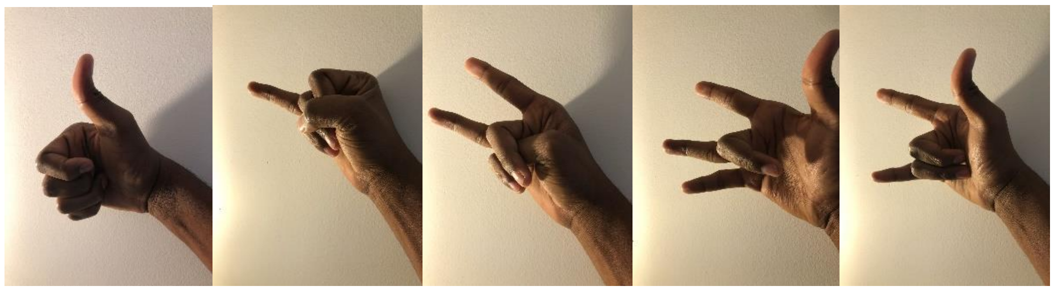


Fig 3.8: Improper Lightning

Figure 3.8 shows poor lightning while capturing gesture.

3. Showing gestures

While showing hand gestures, few people may have long, short, fat, skinny fingers. If person is handicap or have some natural disabilities then he might can show only few fingers. Hence, if we show hand gesture for number one then accurate hand gesture recognition system should recognize all the combination of fingers. We can show 1 finger, 2 fingers, 3 fingers in many different ways (see figure 3.9). An accurate hand gesture recognition should recognize all five fingers clearly.



(a) (b) (c) (d) (e)

Fig 3.9(a), 3.9(b): Alternative way to show number 1 with hand gesture

Fig 3.9(c): Alternative way to show number 2 with hand gesture

Fig 3.9(d): Alternative way to show number 4 with hand gesture

Fig 3.9(e): Alternative way to show number 3 with hand gesture

Figure 3.9 shows different alternative ways to show hand gestures for number 1, 2, 3, 4

Number of combinations to show hand gestures are calculated as below:

- *Number one*

By using permutation combination, with five fingers if we want to show two fingers then we will have 5 combinations ($5C_1 = \frac{5}{1} = 5$) . We aim to recognize all 5 combinations.

- *Number two*

By using permutation combination, with five fingers if we want to show two fingers then we will have 10 combinations($5C_2 = \frac{5\times4}{1\times2}$ = 10). We aim to recognize all 10 combinations.

- *Number three*

By using permutation combination, with five fingers if we want to show three fingers then we will have 10 combinations ($5C_3 = \frac{5\times4\times3}{1\times2\times3} = 10$). We aim to recognize all 10 combinations.

- *Number four*

By using permutation combination, with five fingers if we want to show four fingers then we will have 5 combinations ($5C_4 = \frac{5*4*3*2}{1*2*3*4} = 5$). We aim to recognize all 5 combinations.

- *Number five*

By using permutation combination, with five fingers if we want to show five fingers then we will have 1 combination ($5C_5 = \frac{5*4*3*2*1}{1*2*3*4*5} = 1$). We aim to recognize all 5 combinations.

4. Movement of lecture’s hand

It is very natural to have some hand movements while lecturer is explaining some stuff in class. It is also normal to have finger pointing to some pictures or object while explaining. Lecturer cannot control his body movements while explaining. If there a class with having all handicap and disable students then it is very obvious that lecturer may can use sign language to explain.

In our system, we use hand gesture recognition to click picture of board. Hence, we need to identify that which gesture lecturer made to capture picture and which gesture lecturer made to explain some stuff. We first thought to use motion sensors to capture gestures. If we use motion sensor for hand gesture recognition then it will sense all the hand gestures made by lecturer. If senor senses all the movements then camera will take thousands of unnecessary white-board images including lecture's body in images. Hence, we discarded the idea of using motion sensor as it is not fulfilling our motivation to reduce computation complexity of existing system. Instead of reducing computation complexity, motion sensors are increasing our computational complexity by taking many pictures of whiteboard and including lecture's body. Hence, we decided to recognize hand gesture with camera that can provide continuous video streaming. Also, by implementing Region of Hand (ROI), we are allowing lecturer to make as many movement of hand they want to make while explaining a theorem to student. The detail description of ROI is given in chapter 4.

3.2 Raspberry Pi

3.2.1 Motivation to use Raspberry Pi

This section will give the main motivation to use Raspberry Pi in this system.

1. **Price**

   With the help of Raspberry Pi we can build project in any field of computer science. For example, IoT based projects, computer vision projects and many more. We are using Raspberry Pi in our system to build highly inexpensive IoT based smart classroom. The price of Raspberry Pi model 3B is 35$ and price of Raspberry Pi 5 mega pixel camera is 15$. Hence, the total cost to build this system is 50$. Now let say we want to install this system into 10 class then total price will be 500$ which much cheaper than the price of smart interactive white board. The price range of smart interactive white board is 1000-7000* per class. Hence, interactive smart white board will be very costly to install in every class.

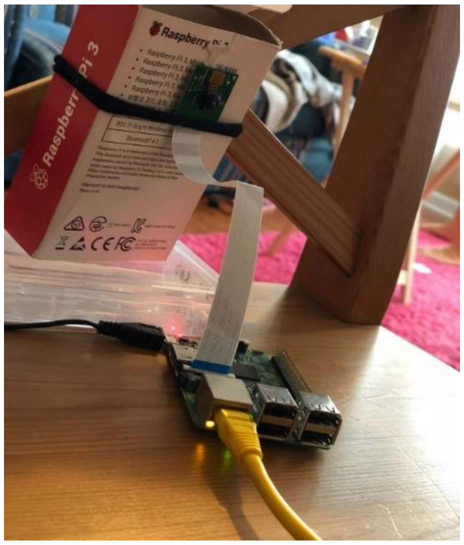


Figure 3.10: Raspberry Pi with Camera

Figure 3.10 Raspberry Pi model 3B & 5 mega pixel camera

*Price range information has been taken from: https://homeschoolbase.com/smart-board-cost/

2. **Size**

   With the help of internet and Raspberry Pi we can connect all the devices available in entire classroom to one network and all the devices can interact with each other. There is no need of using computers with high computational quality and long wires to connect sensors, cameras and other devices of classroom like smart chairs, e-boards to a computer. Raspberry Pi is as strong as computer with smaller size like a credit card.

3. **Raspberry Pi specification**

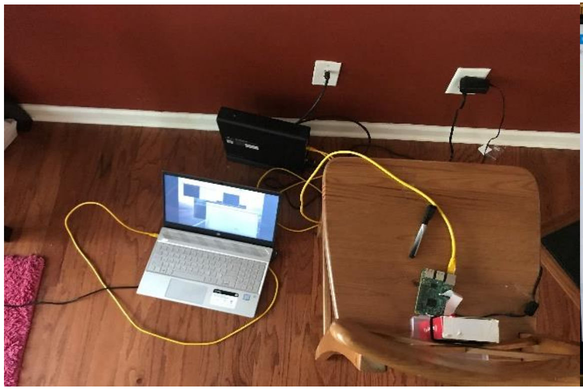
   Raspberry Pi has below mentioned technical specification:

   - CPU: Broadcom BCM2837 SOC 64-bit quad-core
   - Memory: 1 GB of RAM
   - Wi-Fi Support: 802.11n Wireless LAN
   - Bluetooth: 4.1 Bluetooth Low Energy (BLE)
   - USB Ports: 4-USB ports to connect mouse, keyboards etc.

- Ethernet Port: Ethernet port to use internet and set up Raspberry Pi for the first time with monitor.
- GPIO Pins: Raspberry Pi 3 supports 40 GPIO Pins General Purpose Input Output. These digital input/output pins can be used to drive LED, Switches, and Sensors etc.
- Full HDMI Port: Support HDMI port (High-Definition Multimedia Interface) which can be used to quickly connect raspberry pi to HDMI Monitor. With HDMI Cable and Monitor we can add Screen to Raspberry Pi.
- Micro SD card slot: The Micro SD Card will hold the operating system which will boot while we power on Raspberry Pi 3. In next tutorial, we will learn how to setup and prepare SD card with Raspbian OS.
- Audio/Video: Combined 3.5mm audio jack and composite video
- Display interface (DSI): enable us to interface Display Module
- Camera interface (CSI): enable us to interface Camera Module
- Graphics Support: Video Core IV 3D graphics core for advance graphics capabilities.

3.2.2 Initial Installation of programs into Raspberry Pi

In this system Raspberry Pi is a very crucial device. All programs of this system are installed in Raspberry Pi such as gesture recognition program, image to text conversion program, connection to google drive program (see figure 3.11). To initially install all the programs into Raspberry Pi, we are connecting Raspberry Pi to the laptop via Ethernet. After that, to monitor the functionality of installed programs, we are connecting laptop to LED (Liquid Crystal Display) TV.


(a)

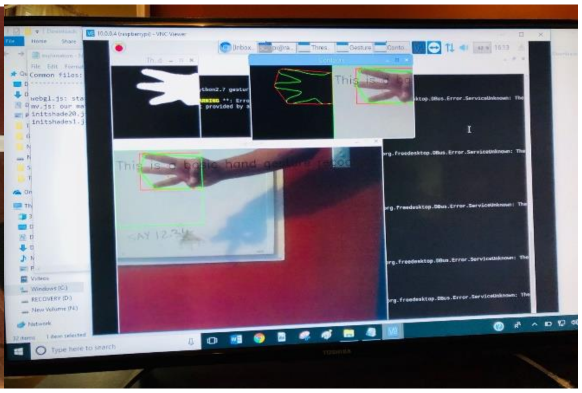

(b)

Fig 3.11(a): Raspberry Pi, Laptop connection to ethernet

Fig 3.11(b): Laptop connection to LED TV

3.2.3 Utilization of Raspberry Pi

Raspberry Pi is used in this system to capture gestures and images of white-board. As mentioned in section 3.1.2, Raspberry Pi is kept in front of white-board. The lecturer will show gestures in contour box of video streaming. As soon as the gesture is captured and perceived, the Raspberry Pi will capture the image of board. The captured images through Raspberry Pi is called as “input images” in our system. Machine learning programs installed in Raspberry Pi will extract text from the input images. This extracted text will later be stored in editable format of text file which we call as “data file” in our system.

This data files and input images are now provided to student’s desktop application. A small Internet of Things system (IoT) is created by connecting Raspberry Pi, Google Drive and Student’s desktop to one network (see figure 3.12). Raspberry Pi transfers data file to google drive, and students retrieve data file from google drive. From capturing gestures till sending data file to google drive all the process are done with the help of Raspberry Pi. Hence, Raspberry Pi is very important device in this system.

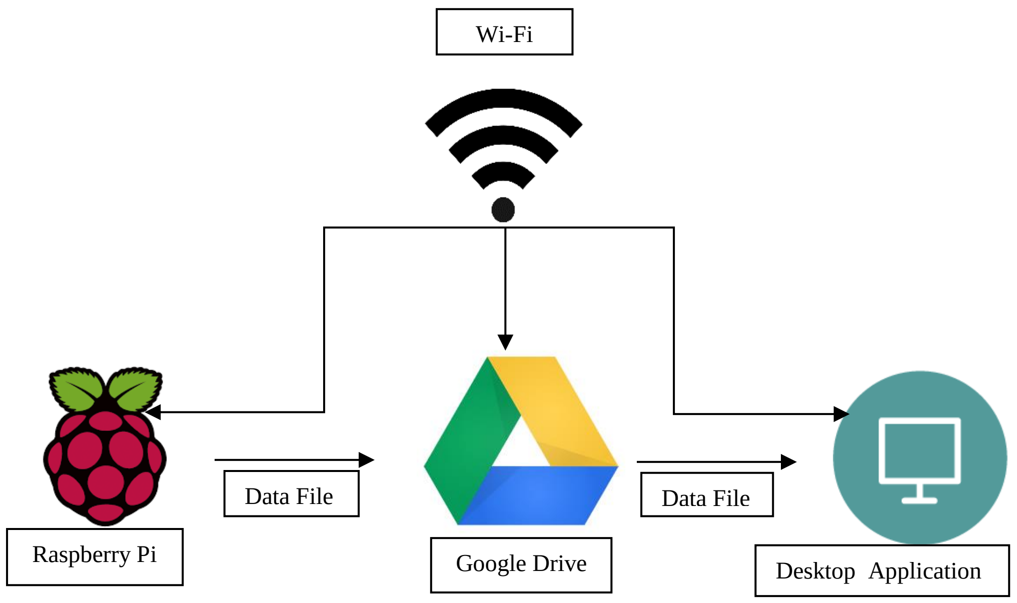


Fig 3.12: IoT system built to transfer data file

Figure 3.12 shows the architecture of IoT system. The Raspberry Pi, Google Drive and Desktop Application is connected through one network (Wi-Fi). Raspberry Pi will send data files and input images to google drive. Students will retrieve the input images and data files form google drive via desktop application.

3.2.4 Challenges of Raspberry Pi

1. Constant Electricity Supply

Raspberry Pi needs to be constantly connected to power supply till it is working. Hence, if we want to install this system in every classroom then we need to make sure that there should be a power supply to a Raspberry Pi. In this system, Raspberry Pi should be kept in front of board hence if power outlet is not nearby then we can use spike busters to give power supply to Raspberry Pi (see figure 3.13).

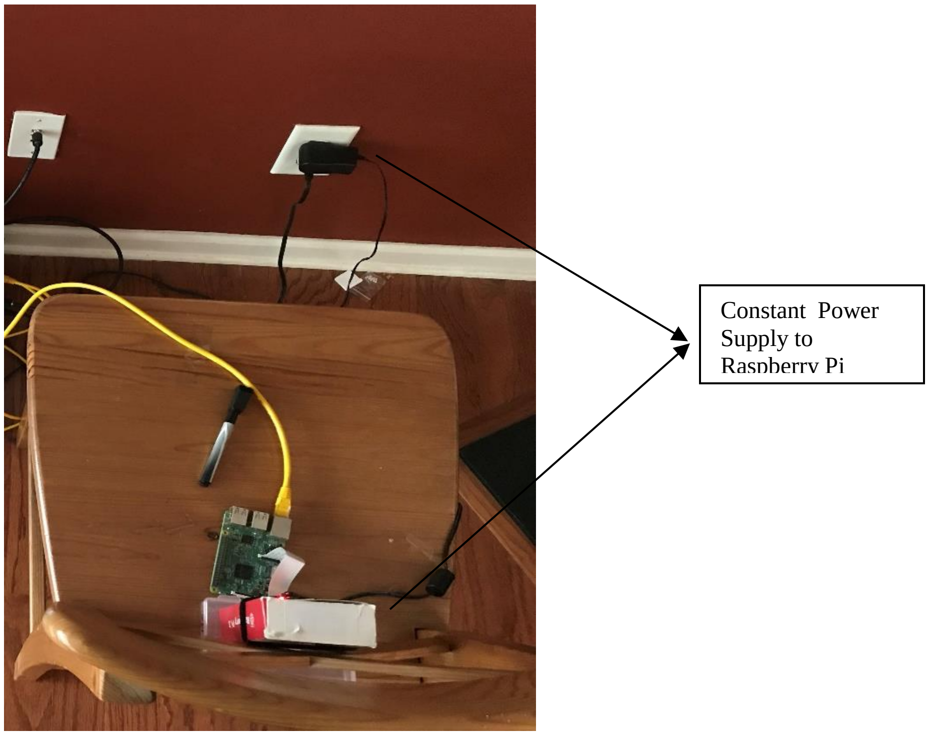
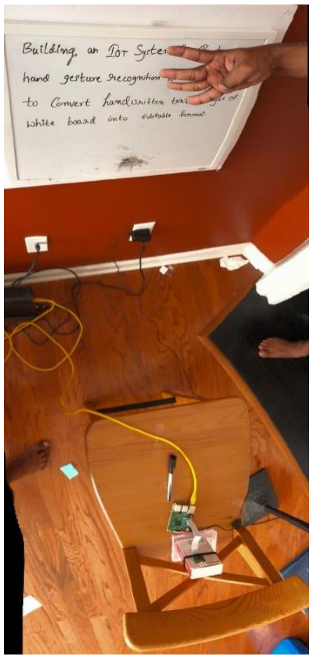


Fig 3.13: Constant Power Supply to Raspberry Pi

Figure 3.13 shows that Raspberry Pi needs constant power supply till it is working.

2. Mounting of Raspberry Pi

In this system, Raspberry Pi needs to be in front of board with constant power supply. Hence, Raspberry Pi should be mounted in such way that it can capture entire board image. Also, while doing such arrangement we need to keep in mind that Raspberry Pi is constantly connected with power supply. Right now, our system is in experimental phase, hence we are using chair to keep Raspberry Pi (see figure 3.14). While utilizing this system professionally in every class we may need some mounting devices that can keep Raspberry Pi in front of board covering entire board with constant power supply.



(a) (b)

Fig 3.14(a): Mounting of Raspberry Pi through chair.

Fig 3.14(b): Example of mounting device

Figure 3.14(a) shows that Raspberry Pi is mounted in front of board covering entire board with constant power supply. Figure 3.14(b) shows an example of mounting device that can be used in future to mount Raspberry Pi in every class.

3.3 Handwritten image to text

To extract text from Handwritten images we are using machine learning algorithms:

3.3.1 Motivation to use Deep machine learning

In our system we are extracting text from handwritten images hence we need a proper recognition of character with higher accuracy. Also, it is a well-known fact that Machine learning gives proper accuracy and perfect result. Hence, we are using machine learning algorithms in our system. Below subsection mentions the strength of machine learning.

3.3.3.1 Basics of neural network

Artificial neural network has been use in many fields of study such as image processing, document clustering, segmentation, biology system, geography, chemistry, decision-making, data mining, optical character recognition (OCR), pattern recognition and many other interesting fields of research [31].

Warren McCulloch and Walter Pitts introduced a binary threshold as a computational model for neural network in 1943 [32]. Neural network is the most popular choice for developing character recognition system, which this network could learn well and provide high accuracy and speed for character identification. Neural network has been using to solve many character recognition problem; such as Chinese character recognition.

The area of Artificial Neural Network derives its basis from the way neurons interacts and function in human brain. The human brain is known to operate in a parallel manner for recognition, reasoning and damage recovery. Because of the ability of ANN to deal with above kind of processes, it can be used from simple applications to the complex applications like pattern recognition algorithms. One neuron in the network at one time is able to link with more than 10,000 other neurons to generate and share new knowledge. Neurons are linked with other neurons in the network through links known as synapse. Neurons in the network receive many inputs either from other neurons or directly as original data. Each neuron has a single threshold value also.

Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the “output layer” and in classification settings it represents the class scores. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

3.3.1.2 Convolutional Neural Network

Pattern or object recognition is usually done with feature extraction and classification. The feature extraction typically uses a variety of methods to get a representation of the data and then use the classifier to classify the data. The process is conducted manually and separately. Lately, the feature extraction and classification integrated automatically in one process or method. The method used to model high-level abstractions in data [4]. It is often called as deep learning techniques.

Convolutional Neural Network (CNN) is one of the deep learning architecture. It can extract multiple features from low-features to high-features automatically. Convolutional Neural Network is also called as ConvNet. Currently, CNN is a state of the art of handwriting characters recognition.

ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the number of parameters in the network.

Some problems in handwriting recognition are due to the high uncertainty of the input data, as the written characters of each person are different, some characters have a very similar shape, disconnected or distortion characters, the written characters have a different thickness and use of various scanners.

Regular Neural Nets don’t scale well to full images. The fully-connected structure of regular Neural Network does not scale to larger images. For example, an image of more respectable size, e.g. 200x200x3, would lead to neurons that have 200*200*3 = 120,000 weights. Moreover, we would almost certainly want to have several such neurons, so the parameters would add up quickly. The fully connectivity of Regular NN leads to wastage of memory and the huge number of parameters would quickly lead to overfitting. While, in CNN images are passed smaller in size compare to Regular NN.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. Unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note: The word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.)

For example, the input images have an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). The neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer has dimensions 1x1x10, because by the end of the ConvNet architecture, the full image would be reduced to single vector of class scores, arranged along the depth dimension.

Here is a visualization:

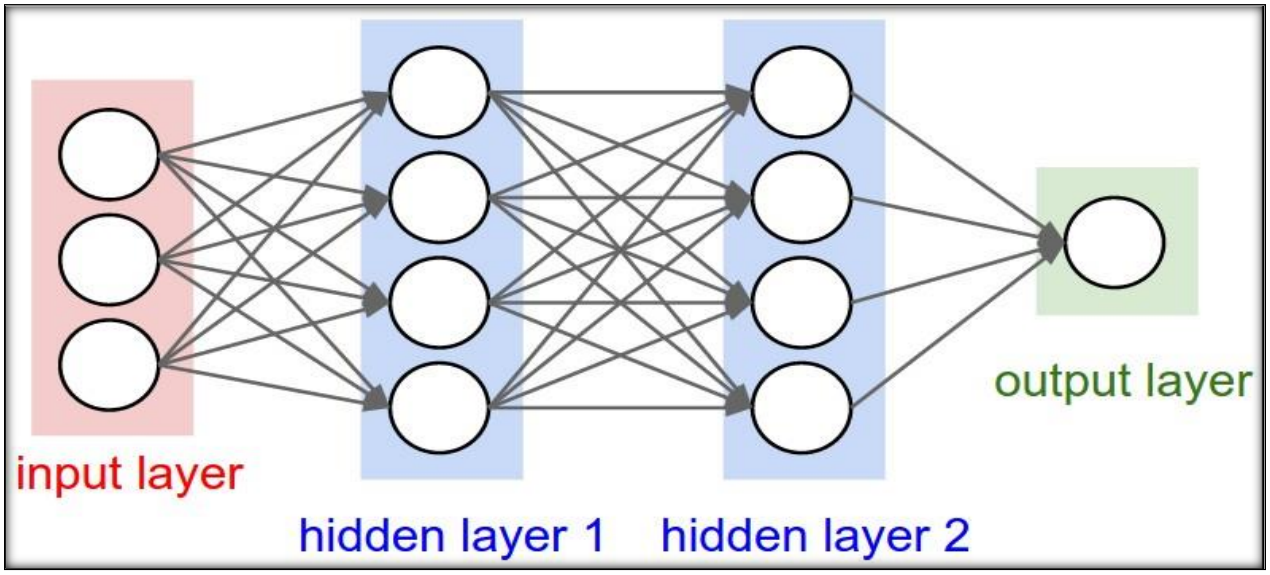


Fig 3.15: Regular 3- layer Neural Network

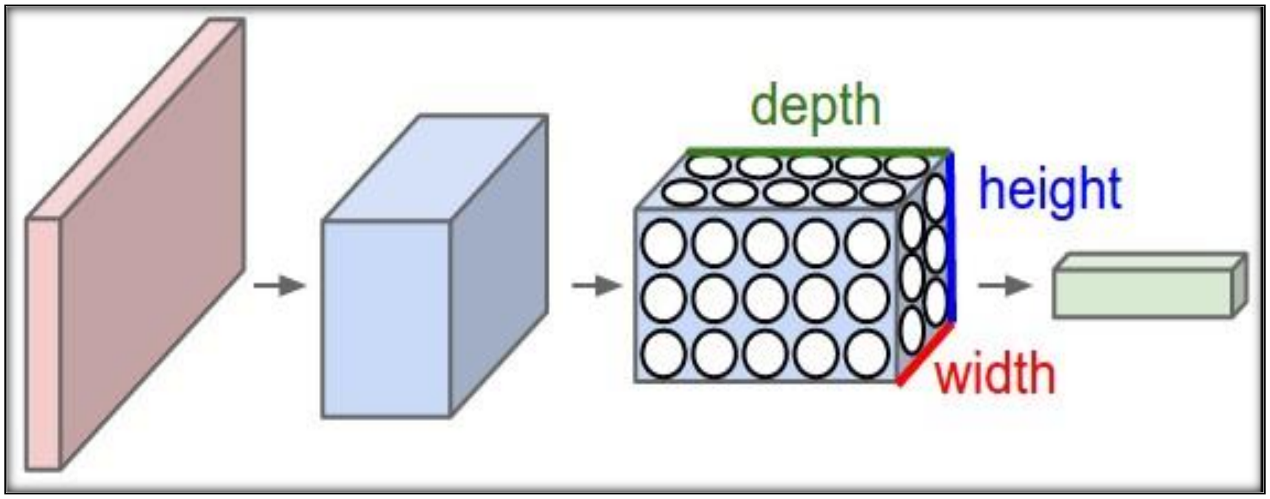


Fig 3.16: Convolutional Neural Network

In figure 3.15, A regular 3-layer Neural Network is shown. In figure 3.16, A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers.

Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels)

3.3.2 Challenges of Text Conversion

Image to text conversion has below mentioned difficulties:

- A Neural Network needs to be trained with different types of handwriting to make it robust and to get higher accuracy in character recognition.
- The classic difficulty of being able to correctly recognize language symbols is the complexity and the irregularity among the pictorial representation of characters due to variation in writing styles, size of symbols etc.
- Text conversion needs some pre-processing, segmentation, feature extraction etc.
- Variation in handwriting among different writers occurs since each writer possesses own speed of writing, different styles, sizes or positions for characters or text. Variation in handwriting styles also exists within individual person's handwriting. This variation may take place due to: writing in various situations that may or may not be comfortable to writer; different moods of writer; style of writing same characters with different shapes in different situations or as a part of different words; using different kinds of hardware for handwriting.

To reduce the difficulty level of text conversion all the possible ways are considered while training dataset and a CNN model. The further detailed description of CNN model training has been provided in chapter 4.

CHAPTER 4

IMPLEMENTATION OF SYSTEM

This chapter gives thorough explanation of how all components of the system have been designed and implemented.

4.1 Gesture Recognition

Our very first step for the entire system is to recognize the pre-set gesture and capture a photo of white board. Hence, capturing gesture is very first stage for implementation. This chapter gives thorough knowledge of how we have implemented gesture recognition. In paper [22], [33] , [44] convexity defect method has been used to recognize finger. We have also used convexity defect method to recognize finger but our approach is quite different then approach of papers [22], [33], [44].

<u>Comparison of our approach with existing approach</u>

In paper [22] , [33] , [44] and in our system to recognize hand gesture , convexity defect approach has been used. In this approach , the convexity defect is found to calculate numbers of fingers shown in gesture. The gap between two finger is called as a finger angle and one point is drawn in this gap is called as the convexity defect (see figure 4.1). Hence, the convexity defect between fingers is used to calculate numbers of fingers shown in gestures. So , for five fingers we will have four gaps among fingers hence we will have 4 convexity defect points, for four fingers we will have 3 gaps hence our convexity defect is 3. So , for the number of fingers we show in gesture , we will have one less number of convexity defect. For one finger convexity defect = 0 , for two fingers convexity defect =1 , for three fingers convexity defect = 2 , for four fingers convexity defect = 3 , for five fingers convexity defect = 4. Problem raise for convexity defect approach when we don't show any gesture , when we show fist and when we show one finger. For all these gestures we will have convexity defect = 0 since there is no gap between fingers. So , how to differentiate between thumb , first finger and clear back ground. We are using area ratio to come out of this problem. In comparison with paper [22] , [33] , [44] we are clearly distinguishing clear background, one finger , thumb , fist , three fingers and "OK" gesture by using new approach of area ratio.

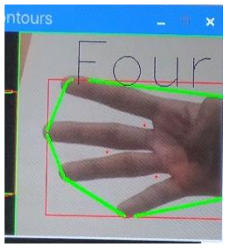
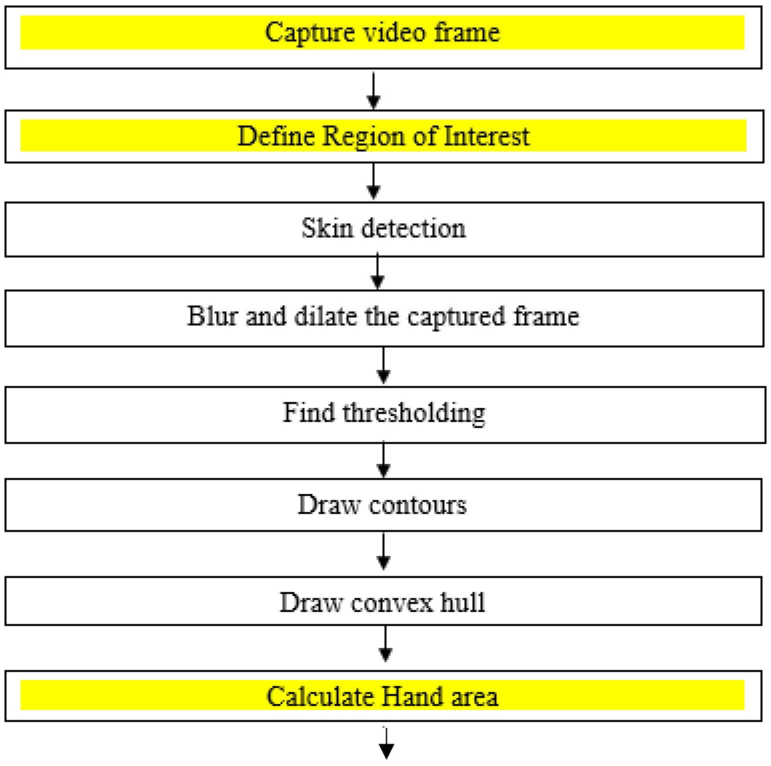


Fig 4.1 Convexity Defect

Figure 4.1 shows the red dot in between finger gap. The red dot called as convexity defect.

The algorithm that we have followed to capture hand gesture is shown in figure 17:



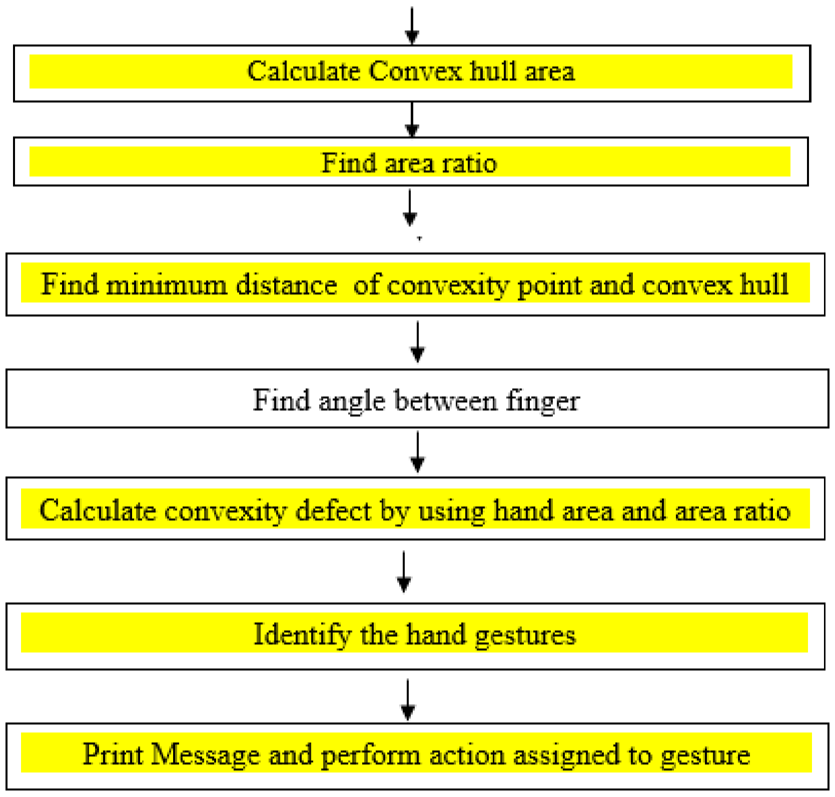

Calculate Convex hull area
Find area ratio
Find minimum distance of convexity point and convex hull
Find angle between finger
Calculate convexity defect by using hand area and area ratio
Identify the hand gestures
Print Message and perform action assigned to gesture

Fig: 4.2 Our approach for Hand gesture recognition

Figure 4.2 shows the algorithm designed by us for hand gesture recognition. Highlighted steps show our contribution in designing algorithm.

Now let's illustrate each step of algorithm.

4.1.1 Capture frames and Draw ROI

We are using gesture recognition to avoid blockage of text on white-board due to lecture's body. To avoid this situation, we are drawing a Region of Interest (ROI) box on a video streaming. ROI are samples within a data set identified for a particular purpose (see figure 4.3 ,4.4). In our system, when hand is shown as a gesture then from entire hand data our ROI is only palm hand and fingers. We do not need entire hand for hand gesture recognition.

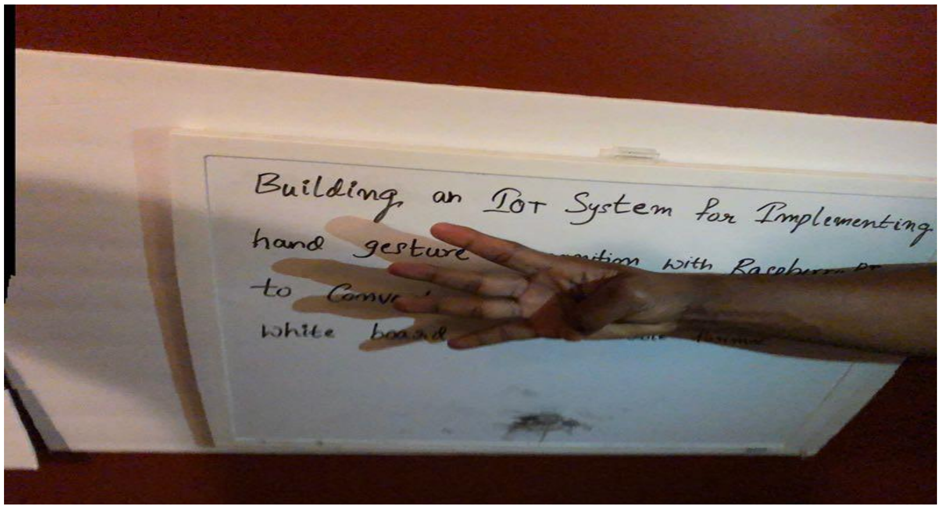


Fig 4.3: Hand Gesture

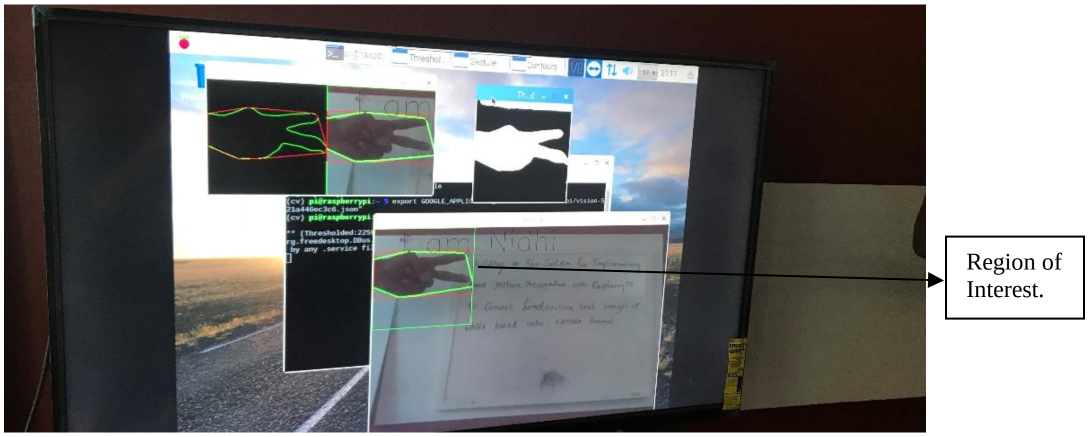


Fig 4.4: Region of Interest for hand gesture

Figure 4.3 shows that professor is showing full hand to show a gesture but form that hand we just need palm area to recognize. Figure 4.4 shows the Region of Interest for palm hand.

In this system, our main motivation to use hand gesture recognition is to reduce computational complexity. Hence, whenever lecturer is done writing on board he will show a hand gesture to camera to take a picture. Now, the problem is lecturer can make hand gestures to explain some stuff to students. We do not want a false hand gesture to be detected by our system. Also, we do not want to stop a lecturer to explain stuff by moving body or hand. Hence, we found a solution that whenever lecturer is done explaining and writing something on board, he can show a gesture to particular box at particular area of board. That particular box is called as “Gesture input box” in our system.

Raspberry Pi will initially do continuous video streaming with ROI drawn on it. This ROI will be kept at one corner of board. Our system will not detect any hand gesture until and unless it is not shown in Gesture Input box and our system will not take any pictures also until and unless the hand gesture is not shown in Gesture Input box. Hence, lecturer can freely move anywhere in class and can freely make hand/body movements while explaining. If lecturer’s body by mistake blocks or enters into ROI box then also our system will not detect false gesture and will not take false images because our ROI is only hand not any other body part or objects. The only constrain for lecturer is they have to show a gesture to particular Gesture input box area. This constrain is benefitting our system in below mentioned aspects:

- It allows lecturer to explain theory with hand/body movements which is very important in a class with handicaps and disable students.
- It is not capturing images with lecturer’s body in it as the Gesture input box has kept in such way that lecture’s body cannot block the white-board text.
- Our ROI only take palm hand so it will not detect any false hand gesture made by professor while explaining. Consequently, we will not have extra or unnecessary images of white-board since taking of images directly linked to accurate hand gestures.

Our Region Of Interest (ROI) is the palm hand region, so the palm of the hand are captured. As an example, the portion of the frame in which we are further interested is only to know that whether this pixel is in our interest area or not means just as binary image (see figure 4.5, 4.6).

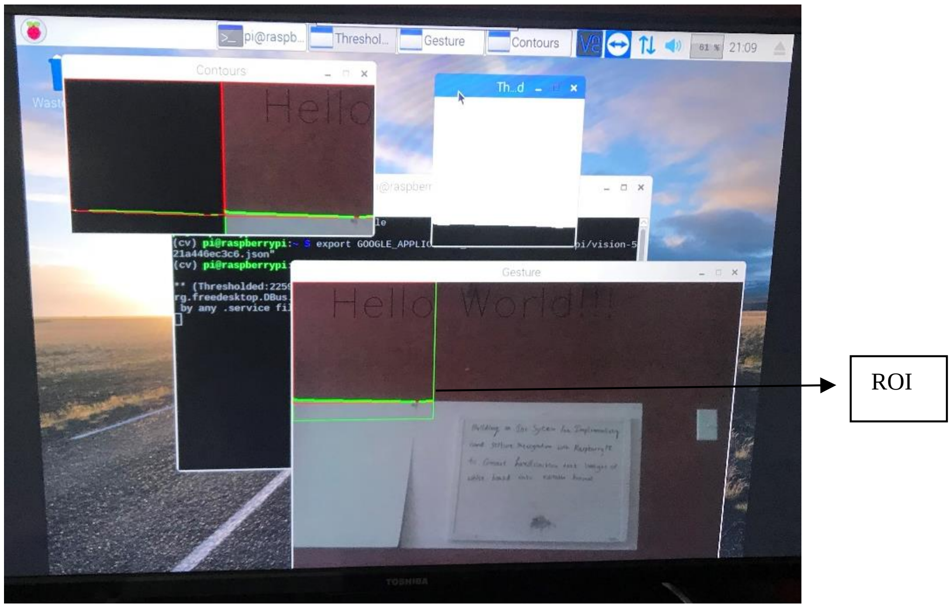


Fig 4.5: Region of Interest kept above the white board.

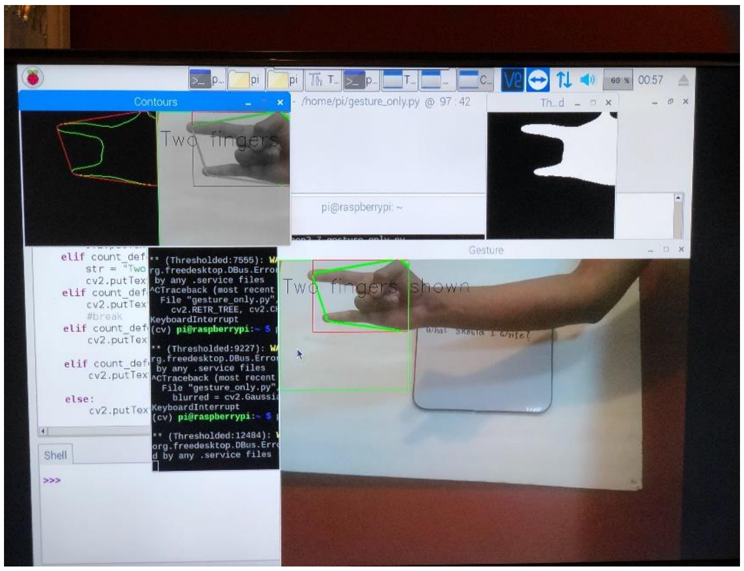


Fig 4.6: ROI box beside board

Figure 4.5 and 4.6 shows the ROI box can be kept at any corner of one board but it should be kept before the board area starts.

4.1.2 Finding skin color area

The by default color space of OpenCV is BGR (blue, green, red). In this system we are mainly working with hand palm region. Hence, we need to convert BGR color space to HSV (Hue, Saturation, Vision) color space. After that we defined the skin color range to detect the hand palm within then range. Then we are finding a mask value which will be 0 or 1, the region which has skin color will be 0 value and of white color rest will be 1 value and black color.

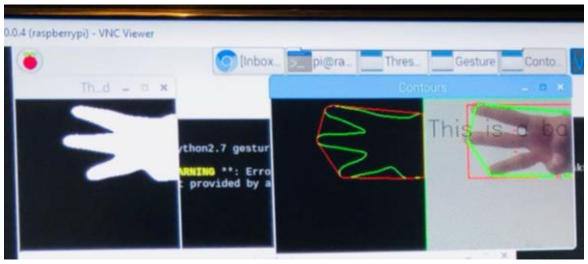
4.1.3 Blur and dilate frame

We blur the frame for smoothing and to reduce noise like shadow and details from the image (see figure 4.7). We are not interested in the details of the image but in the shape of the object to track. We are dilating a frame to enlarge few areas of hands when we have captured few gestures in improper or poor lightning [29].

I've used Gaussian Blurring on the original image. Gaussian blur is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The equation of a Gaussian function in two dimensions is [29]:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e \quad \frac{x^2+y^2}{2\sigma^2} \qquad \text{Eq. (1)}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution.



```
ython2.7 gestur
ARNING **: Erro
 provided by a
```

Fig 4.7: Shadow in hand gesture

Figure 4.7 shows that even though there is a shadow while capturing hand gesture it is clearly recognition hand gesture due to removing noise of hand gesture.

4.1.3 Image segmentation

Segmentation is perform using thresholding. Thresholding creates binary images from grey-level ones by turning all pixels below some threshold to zero and all pixels about that threshold to one. If g (x, y) is a threshold version of f (x, y) at some global threshold T, it can be defined as [26]:

$$g(x, y) = 1 \text{ if } f(x, y) \geq T$$

$$= 0 \text{ otherwise} \quad \text{Eq. (2)}$$

Thresholding operation is defined as:

$$T = M[x, y, p(x, y), f(x, y)] \quad \text{Eq. (3)}$$

In this equation, T stands for the threshold; f (x, y) is the grey value of point (x, y) and p(x, y) denotes some local property of the point such as the average grey value of the neighborhood centered on point (x, y). Otsu method is used for thresholding [26].

In Otsu's method we exhaustively search for the threshold that minimizes the intra-class variance (the variance within the class), defined as a weighted sum of variances of the two classes:

$$\sigma_\omega^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \qquad \text{Eq. (4)}$$

Weights ω_0 and ω_1 are the probabilities of the two classes separated by a threshold t, and $\sigma_0^2$and σ_1^2 are variances of these two classes (see figure 4.8).

The class probability $\omega_{0,1}(t)$ is computed from the L bins of the histogram:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i) \quad \text{Eq. (5)}$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i) \quad \text{Eq. (6)}$$

Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance:

$$\sigma_b^2(t) = \sigma^2 - \sigma_\omega^2(t) = (\omega_0\,\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$$

$$= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 \qquad \text{Eq. (7)}$$

which is expressed in terms of class probabilities ω and class means μ.While the class mean $\mu_0, 1, T^{(t)}$ is [26]:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)} \qquad \text{Eq. (8)}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)} \qquad \text{Eq. (9)}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i) \qquad \text{Eq. (10)}$$

The following relations can be easily verified:

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T \qquad \text{Eq. (11)}$$

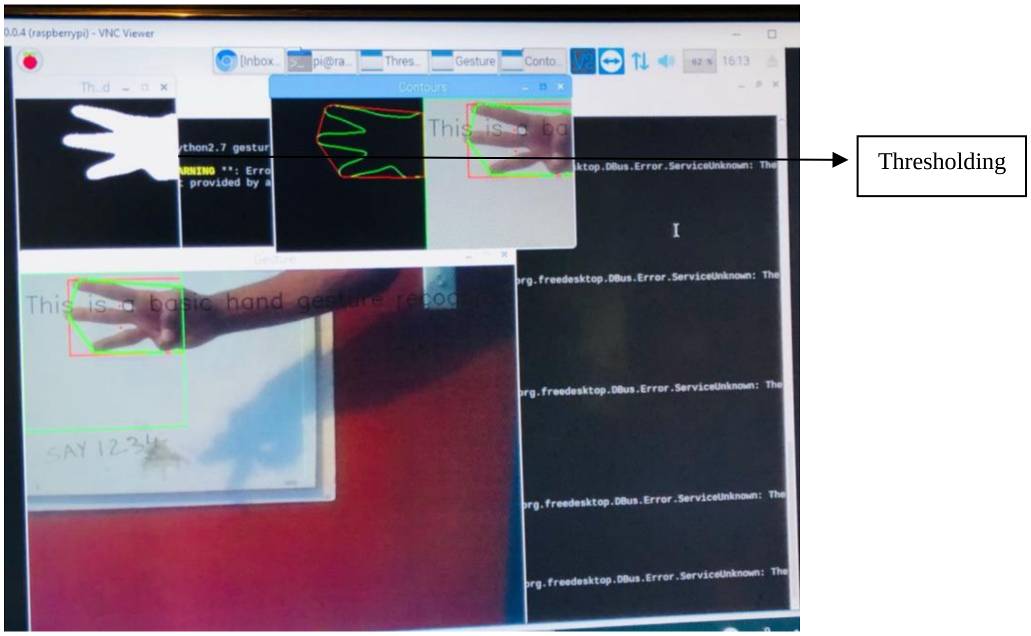
$$\omega_0 + \omega_1 = 1 \qquad \text{Eq. (12)}$$

The class probabilities and class means can be computed iteratively. This idea yields an effective algorithm.

Algorithm [26]

1. Compute histogram and probabilities of each intensity level
2. Set up initial $\omega_i(0)$ and $\mu_i(0)$
3. Step through all possible thresholds t=0, 1, 2…. maximum intensity
    1. Update ω_i and μ_i
    2. Compute $\sigma_b^2(t)$
4. Desired threshold corresponds to the maximum $\sigma_b^2(t)$

Otsu's method exhibits the relatively good performance if the histogram can be assumed to have bimodal distribution and assumed to possess a deep and sharp valley between two peaks.



```
ython2.7 gestur
ARNING **: Erro
 provided by a
ktop.DBus.Error.ServiceUnknown: The
rg.freedesktop.DBus.Error.ServiceUnknown: The
rg.freedesktop.DBus.Error.ServiceUnknown: The
rg.freedesktop.DBus.Error.ServiceUnknown: The
rg.freedesktop.DBus.Error.ServiceUnknown: The
```

Fig 4.8 After applying thresholding

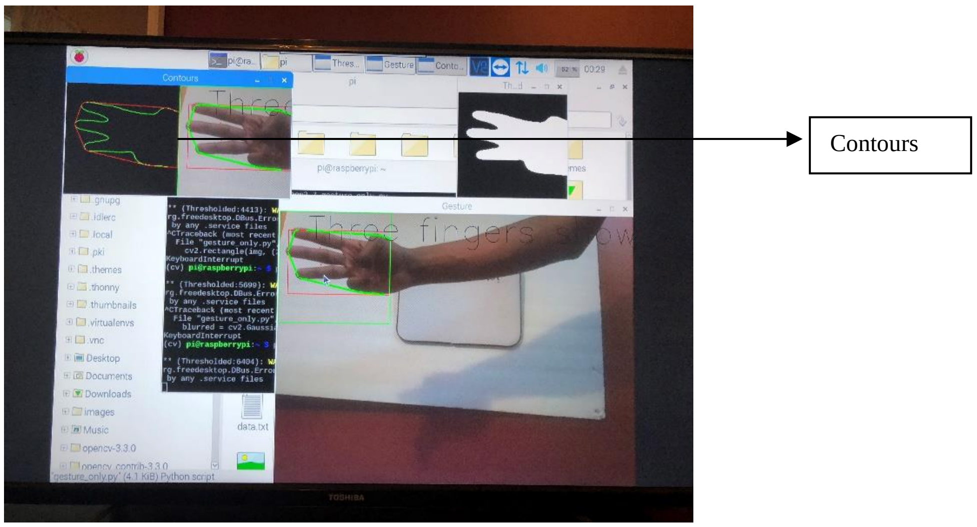
Figure 4.8 shows the thresholded frame.

By performing above step hand is easily detected. The next task is to extract features from that hand to recognize gestures related with that. Quality of extracted features decides accuracy and efficiency of gesture recognition system.

4.1.4 Draw Contours and convex hull

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity (see figure 4.9). The contours are a useful tool for shape analysis and object detection and recognition. Before finding contours, apply threshold or canny edge detection on the image [31].

findContours() function is used to find contours in image. Since OpenCV 3.2, this function no longer modifies the source image but returns a modified image as the first of three return parameters. In OpenCV, finding contours is like finding white object from black background. So, remember, object to be found should be white and background should be black. Contour Approximation will remove small curves, there by approximating the contour more to straight line. This is done using cv2.approxPolyDP() function. So, when you try to find the contours, you will get all the curves also. But with contour approximation, you can avoid all those problems and approximates it to a perfect square [31].



```
** (Thresholded:4413): W
rg.freedesktop.DBus.Erro
 by any .service files
^CTraceback (most recent
  File "gesture_only.py"
    cv2.rectangle(img, (
KeyboardInterrupt
(cv) pi@raspberrypi:~ $

** (Thresholded:5699): W
rg.freedesktop.DBus.Erro
 by any .service files
^CTraceback (most recent
  File "gesture_only.py"
    blurred = cv2.Gaussia
KeyboardInterrupt
(cv) pi@raspberrypi:~ $

** (Thresholded:6404): W
rg.freedesktop.DBus.Erro
 by any .service files
```

Fig 4.9. Image with Contours

Figure 4.9 shows the contours of hand gesture frame.

Once the approximation is over, finding convex Hull is the next step. This will look similar to contour approximation, but it is not same. Here, cv2.convexHull() function checks a curve for convexity defects and corrects it. Generally speaking, convex curves are the curves which are always bulged out, or at-least flat [31]. And if it is bulged inside, it is called convexity defects. Any deviation of the object from this hull can be considered as convexity defect (see figure 4.10). OpenCV comes with a ready-made function for this, cv2.convexityDefects().

hull =cv2.convexHull(cnt,returnPoints =False)

defects =cv2.convexityDefects(cnt, hull)

Notice that "returnPoints = False" in first line to get indices of the contour points, because input to convexityDefects() should be these indices, not original points. It returns a defects structure, an array of four values - [start point, end point, farthest point, approximate distance to farthest point]. We can visualize it using an image [31]. We draw a line joining start point and end point, then draw a circle at the farthest point.

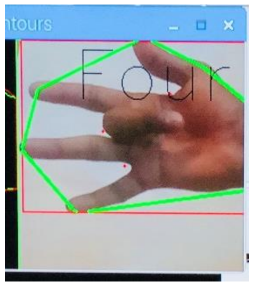


Fig 4.10: Convex hull

Figure 4.10 shows a green pentagon around the hand region. That pentagon is called as convex hull.

4.1.5 Convexity Defect

After finding convex hull it is very important to find convex hull area and hand area. With the help of these two areas we can find the area ratio. Every gesture will have their own unique area ration according that only we can clearly identify all gestures. Hence why, finding area ratio is important [31].

Area ratio=((areahull-areacnt)/areacnt) *100   Eq. (13)

Where, areahull is the area of convex hull, areacnt is area of hand.

We will now find the convex points and the defect points. The convex points are generally, the tip of the fingers. But there is other convex point too [31]. So, we find convexity defects, which is the deepest point of deviation on the contour (see figure 4.11). By this we can find the number of fingers extended and then we can perform different functions according to the number of fingers extended. The gap between two fingers called as an angle between two fingers. This angle will never be more than 90 degrees [31]. Hence, if we find the angle greater 90 degrees in convex hull will be discarded and if we find the angle less than 90 degree then we will draw a convexity defect point between fingers.

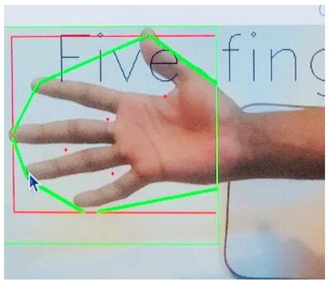


Fig 4.11: Convexity Defect

Figuer 4.11 shows the red point inside green pentagone. The red points are called as Convexity defetcs.

If person will have fat and short fingers then the gap between fingers will be very less or if person don't extend the fingers proprely then aslo the the gap between the finger will be very less. At this point of time to find a convexity difect the minimum distance between finger angle and convex hull border will be counted [31]. The figure 4.12 help in understaning this scenario better.

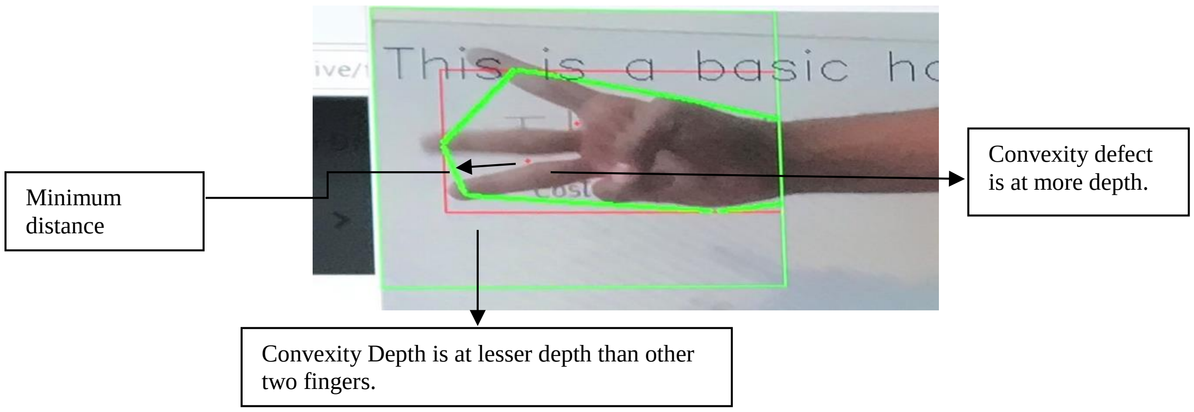
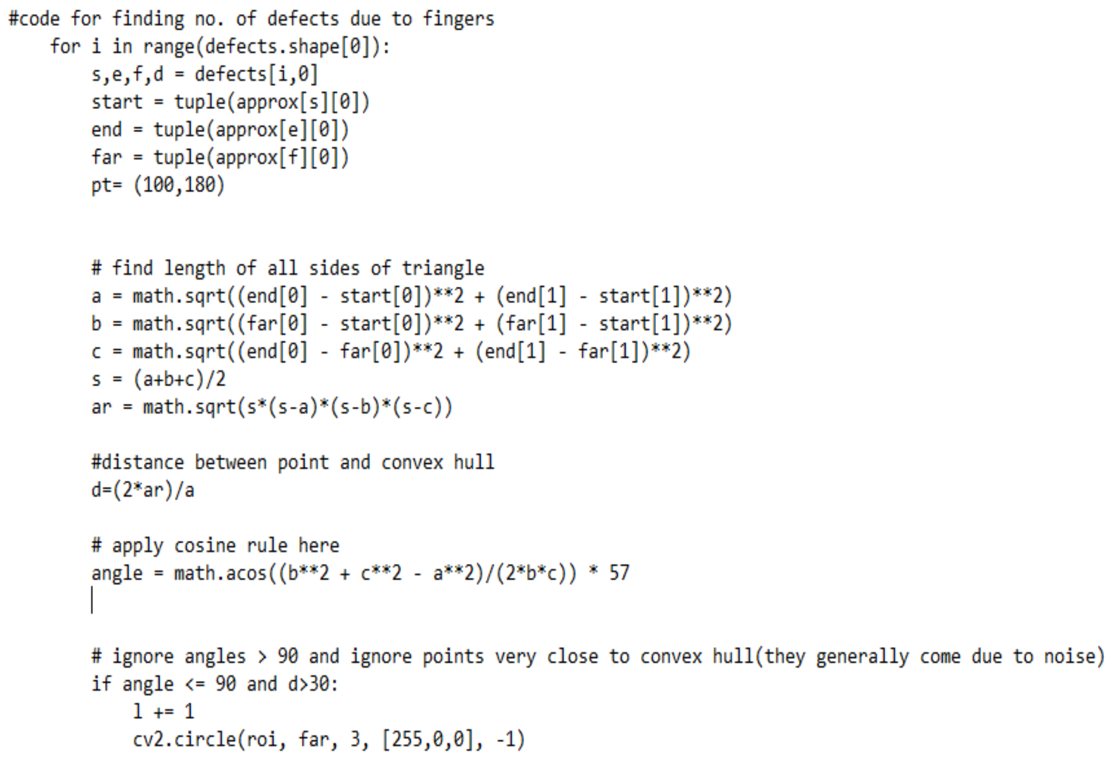

This is a basic h
Minimum distance
Convexity defect is at more depth.
Convexity Depth is at lesser depth than other two fingers.

Fig 4.12: Gap affects the height of convexity depth

Figure 4.12 shows that there is a more gap between 1st and 2nd finger so the convexity defect is at lowest depth of the gap. There is a less gap between 2nd and 3rd fingers so the convexity defect is at little higher distance then other two fingers [31]. In this case, the minimum distance between finger angle and convex hull border will be calculated to locate convexity defect point [31]. The arrow labeled as minimum distance shows the gap between finger angle and convex hull [31].



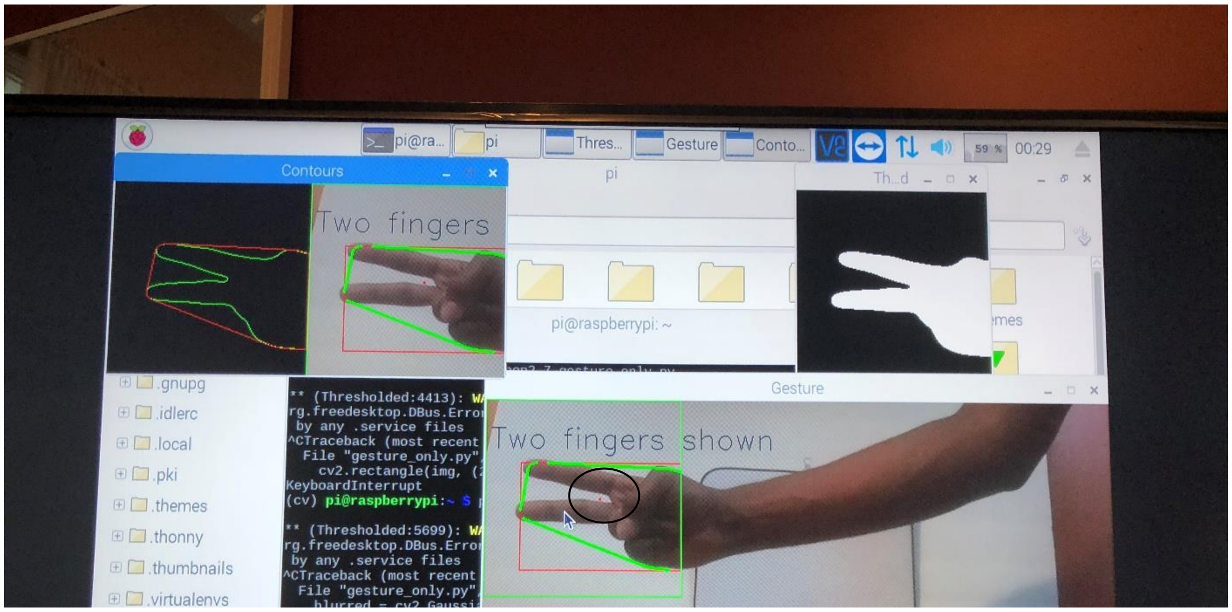
```
#code for finding no. of defects due to fingers
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(approx[s][0])
        end = tuple(approx[e][0])
        far = tuple(approx[f][0])
        pt= (100,180)


        # find length of all sides of triangle
        a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
        b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
        c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
        s = (a+b+c)/2
        ar = math.sqrt(s*(s-a)*(s-b)*(s-c))

        #distance between point and convex hull
        d=(2*ar)/a

        # apply cosine rule here
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57


        # ignore angles > 90 and ignore points very close to convex hull(they generally come due to noise)
        if angle <= 90 and d>30:
            l += 1
            cv2.circle(roi, far, 3, [255,0,0], -1)
```

Fig 4.13 Shows the code snippet

Figure 4.32 shows the code snippet to find number convexity defects, to find minimum distance between convex hull and convexity defect, to find angle between two fingers [31].

4.1.6 Finger detection

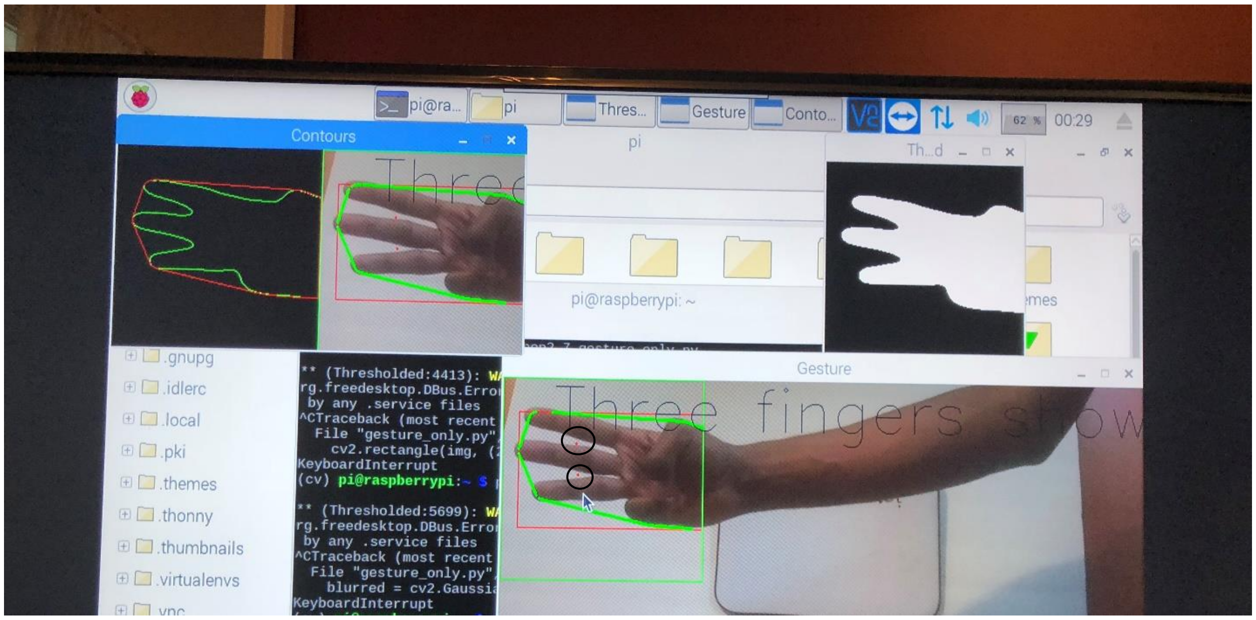
Till now we understood that we are calculating angle between fingers and finding convexity defect. Now, we will see how convexity defect is helpful in detecting fingers. If we want to detect two fingers then we will have one angle (one gap) between two fingers means we will have one convexity defect. Same for three fingers, we will have two angles (two gaps) between three fingers means we will have two convexity defects [31]. For four fingers, we will have three angles (three gap) between four fingers that means we will have three convexity defects between four fingers. For five fingers we will have four angles (four gap) between five fingers hence we will have four convexity defects (see figure 4.14, 4.15, 4.16, 4.17). Hence by this we understood that the number of fingers we show, we will have one less number of convexity defects from the number of fingers shown [31].



```
** (Thresholded:4413): WA
rg.freedesktop.DBus.Erro
 by any .service files
^CTraceback (most recent
  File "gesture_only.py"
    cv2.rectangle(img, (
KeyboardInterrupt
(cv) pi@raspberrypi:~ $

** (Thresholded:5699): WA
rg.freedesktop.DBus.Erro
 by any .service files
^CTraceback (most recent
  File "gesture_only.py"
```

Fig 4.14: two finger gesture.

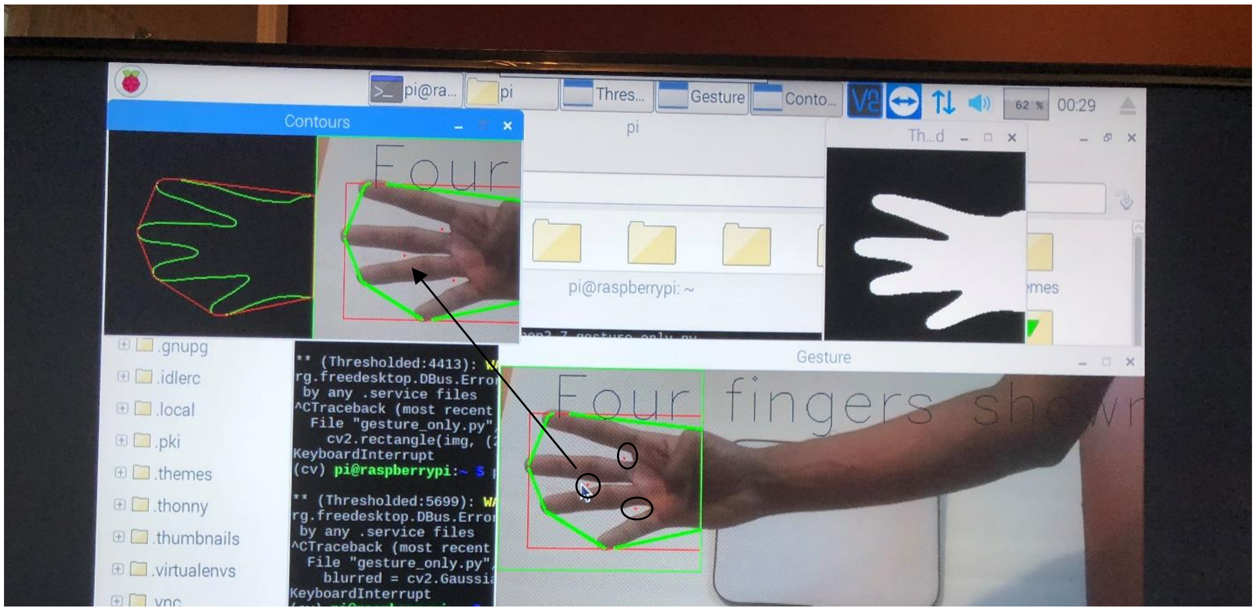
Figure 4.14 shows that if we show two fingers then we will have one convexity defect (highlighted with circle).



```
** (Thresholded:4413): WA
rg.freedesktop.DBus.Erro
 by any .service files
^CTraceback (most recent
  File "gesture_only.py"
    cv2.rectangle(img, (
KeyboardInterrupt
(cv) pi@raspberrypi:~ $

** (Thresholded:5699): WA
rg.freedesktop.DBus.Erro
 by any .service files
^CTraceback (most recent
  File "gesture_only.py"
    blurred = cv2.Gaussi
KeyboardInterrupt
```

Fig 4.15: three finger gesture

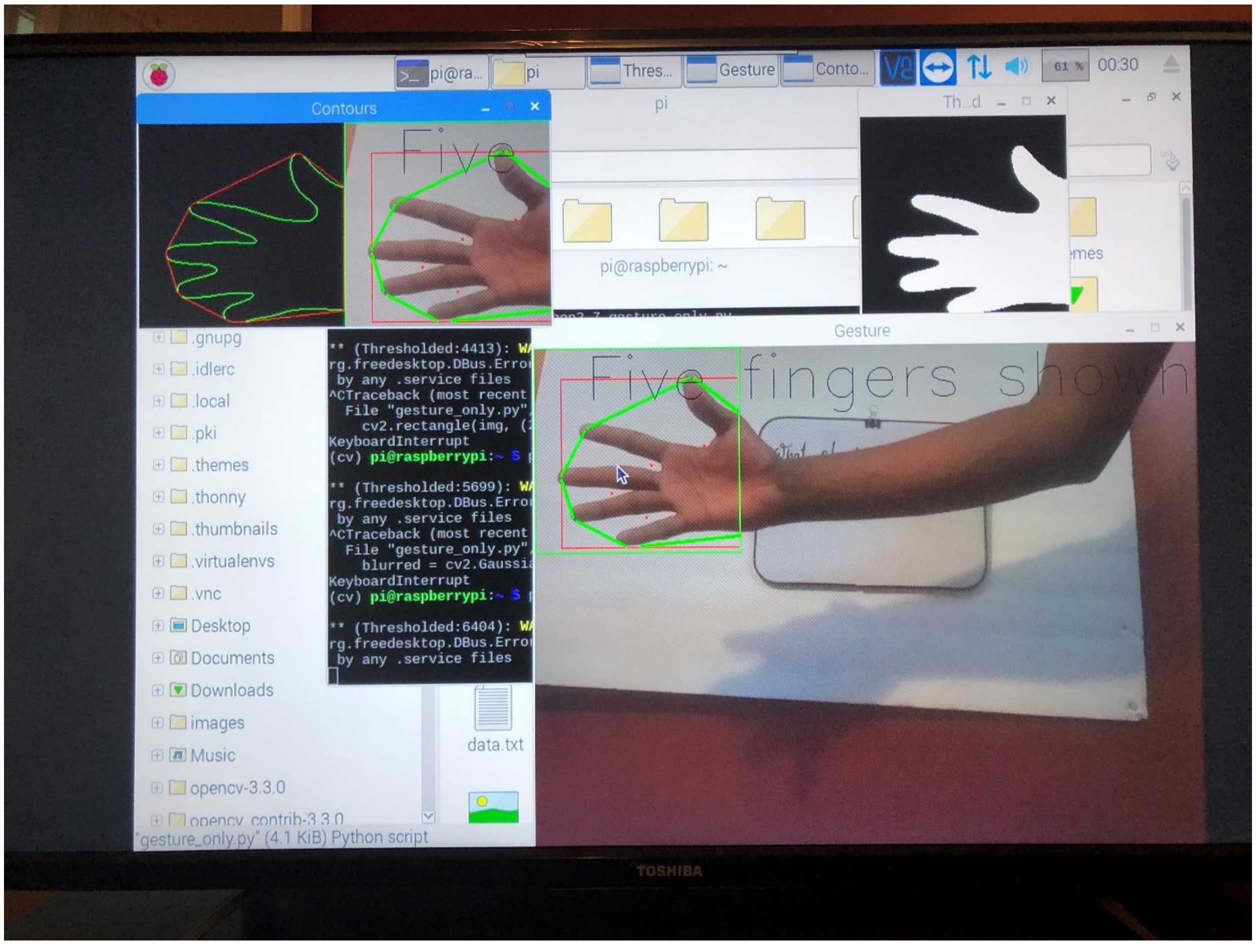
Figure 4.15 shows that if we show three fingers then we will have two convexity defects (highlighted with circle).



```
** (Thresholded:4413): WA
rg.freedesktop.DBus.Error
 by any .service files
^CTraceback (most recent
  File "gesture_only.py",
    cv2.rectangle(img, (2
KeyboardInterrupt
(cv) pi@raspberrypi:~ $ p

** (Thresholded:5699): WA
rg.freedesktop.DBus.Error
 by any .service files
^CTraceback (most recent
  File "gesture_only.py",
    blurred = cv2.Gaussia
KeyboardInterrupt
```

Fig 4.16: four finger gesture.

Figure 4.16 shows that if we show four fingers then we will have three convexity defects (highlighted with circle).



```
** (Thresholded:4413): WA
rg.freedesktop.DBus.Error
 by any .service files
^CTraceback (most recent
  File "gesture_only.py",
    cv2.rectangle(img, (2
KeyboardInterrupt
(cv) pi@raspberrypi:~ $ p

** (Thresholded:5699): WA
rg.freedesktop.DBus.Error
 by any .service files
^CTraceback (most recent
  File "gesture_only.py",
    blurred = cv2.Gaussia
KeyboardInterrupt
(cv) pi@raspberrypi:~ $ p

** (Thresholded:6404): WA
rg.freedesktop.DBus.Error
 by any .service files
```

Fig 4.17: five finger gestures.

Figure 4.17 shows that if we show five fingers then we will have four convexity defects (highlighted with circle).

After seeing this we can understand that if we are showing one finger or fist then there will not have any gap or angle hence the complexity defect count will be zero. Hence, for five fingers we will have only 4 complexity defects. Now, the challenge is if ROI will not have any gestures and ROI box will be empty then we will have complexity defect as zero. If we show one finger or fist also then the complexity defect will be zero. So, how to detect that ROI box is empty or ROI box has one finger or fist. To clearly identify the difference between empty ROI box, one finger and fist we are using hand area and area ratio. We have done numerous trial and error methods and we found accurate number to clearly find the difference between empty ROI box, one finger, fist and thumb. For all these four-scenario complexity defect will be 0. Same situation is with 3 fingers also. In 3 fingers for "OK" gesture also there will 2 convexity defects and for three fingers also there will be 2 convexity defects [31].

To clearly distinguish all these gestures, we are fixing area ratio and hand area values. This value will work with all kind of fingers [31]. To recognize only an empty ROI box, we are using hand area because there will not be any hand for empty box or hand area will be very minimal [31]. Hence for Hand area =2000 we are getting empty box. Rest of all gestures to clearly distinguish all gestures we are using area ratio values which is mentioned in table 4.1 (for result see chapter 6).

Gestures	**Area Ratio**	**Hand Area**
ROI empty box	-	<2000
Fist	<12	-
Thumb	<17.5	-
One finger	>18	-
Three	<27	-
OK	>27	-

Table 4.1: Calculated values to clearly distinguish all same gestures [31].

Table 4.1 shows value of area ratio assigned for a particular gesture to recognize.

Now, after recognizing all the fingers correctly, we need to print messages for the detected fingers and all the functions assigned with gesture should work when particular gesture is shown . We have assigned messages which is mentioned in table 4.2 to all gestures.

Gestures	**Message Print**	**Function**
ROI empty box	Show gesture	
One finger	One finger shown	
Two Fingers	Two fingers shown	
Three Fingers	Three fingers shown	Takes picture for mathematic Formula conversion
Four Finger	Four Fingers shown	Take picture for Image to text conversion
Five Fingers	Five Fingers shown	
Fist	Fist shown	
Ok	It's okay	
Thumb	All is well	Exit from entire system

Table 4.2: Message displayed when gestures are recognized.

Table 4.2 shows the messages that display while gesture is recognized. Also, table shows a task system has to perform while a particular gesture is shown.

Raspberry Pi takes picture of white-board after the third and fourth gestures are shown. Four fingers are shown when lecturer has written special characters, numbers, alphabets and simple mathematical formulas. Three finger gestures are shown when complex mathematical formulas are written on board. To understand simple and complex mathematical function for our system see subsection 4.2.

4.2 Deep machine learning for image to text conversion

After capturing images through Raspberry Pi, next task is to extract text from images (see figure 4.18). For classification of text we have built, trained and tested a convolutional neural network.

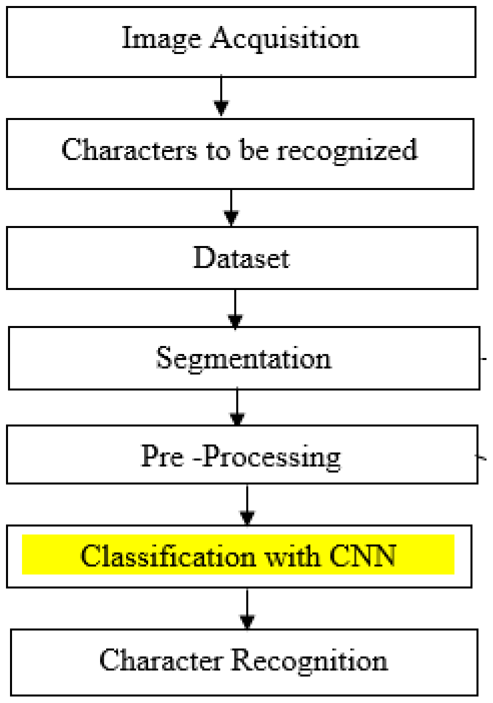


Fig 4.18: Steps followed for image to text conversion

Figure 4.18 shows the procedure followed to extract text from captured images. Highlighted step shows that we have built convolutional neural network to classify text in images.

4.2.1 Image Acquisition:

This is the way toward gaining advanced arrangement pictures from physical record or image that can be additionally subjected to image handling procedures and can be additionally utilized as a part of the application. In our system image is acquired through Raspberry Pi.

4.2.2About characters to be recognized

The modern English alphabet is a Latin alphabet consisting of 26 letters, each having an uppercase and a lowercase form. The English language was first written in the Anglo-Saxon futhorc runic alphabet, in use from the 5th century. This alphabet was brought to what is now England, along with the promo-form of the language itself, by Anglo-Saxon settlers.

Capital Alphabets

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

Small Alphabets

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

Table 4.3: alphabets

Table 4.3 shows the alphabets supported in our system for image to text conversion.

The exact shape of printed letters varies depending on the typeface (and font). The shape of handwritten letters can differ significantly from the standard printed form (and between individuals), especially when written in cursive style.

Numerals

0	1	2	3	4	5	6	7	8	9

Table 4.4: numerals

Table 4.4 shows the numerals that are supported in our system for image to text conversion.

Special Characters

~	`	!	@	#	$	%	^	&	*	(	)	_	-	+	=	{	}	[	]	:	;	'	"	|	\	<	>	,	.	?	/

Table 4.5: Special Character

Table 4.5 shows the special characters that are supported in our system for image to text conversion.

So, the total number of characters to be recognized are 64 in that 26 Capital Alphabets, 26 small alphabets, 10 numerals.

4.2.3 About dataset

For training and testing purpose, two types of datasets are used which are

- The MNIST dataset
- The EMNIST Database

The MNIST Database

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems (see figure 4.19) [1][2]. The database is also widely used for training and testing in the field of machine learning [3][4]. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments [5]. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels [5].

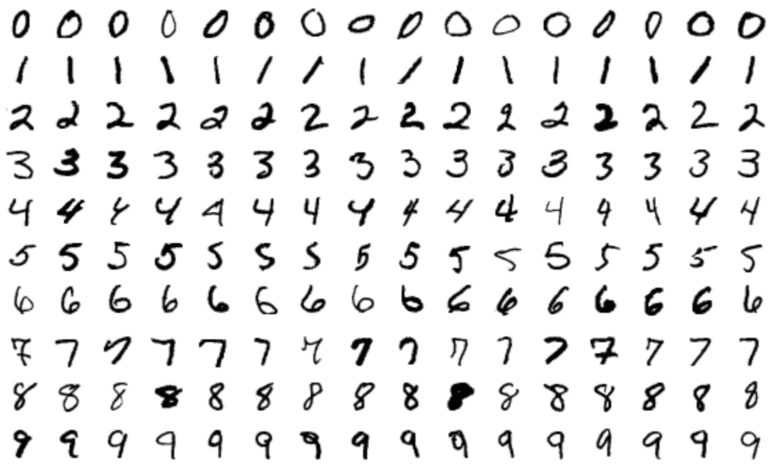


Fig 4.19: Sample Images from MNIST test dataset

The figure 4.19 shows the Sample of MNIST dataset.

The MNIST database contains 10 million images.Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset [7]. The digits have been size-normalized and centered in a fixed-size image.

<u>EMNIST dataset:</u>

The EMNIST dataset is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 28x28 pixel image format and dataset structure that directly matches the MNIST dataset. Special Database 19 contains NIST's entire corpus of training materials for handprinted document and character recognition. It publishes Handprinted Sample Forms from 3600 writers, 810,000-character images isolated from their forms, ground truth classifications for those images, reference forms for further data collection, and software utilities for image management and handling (see figure 4.20).

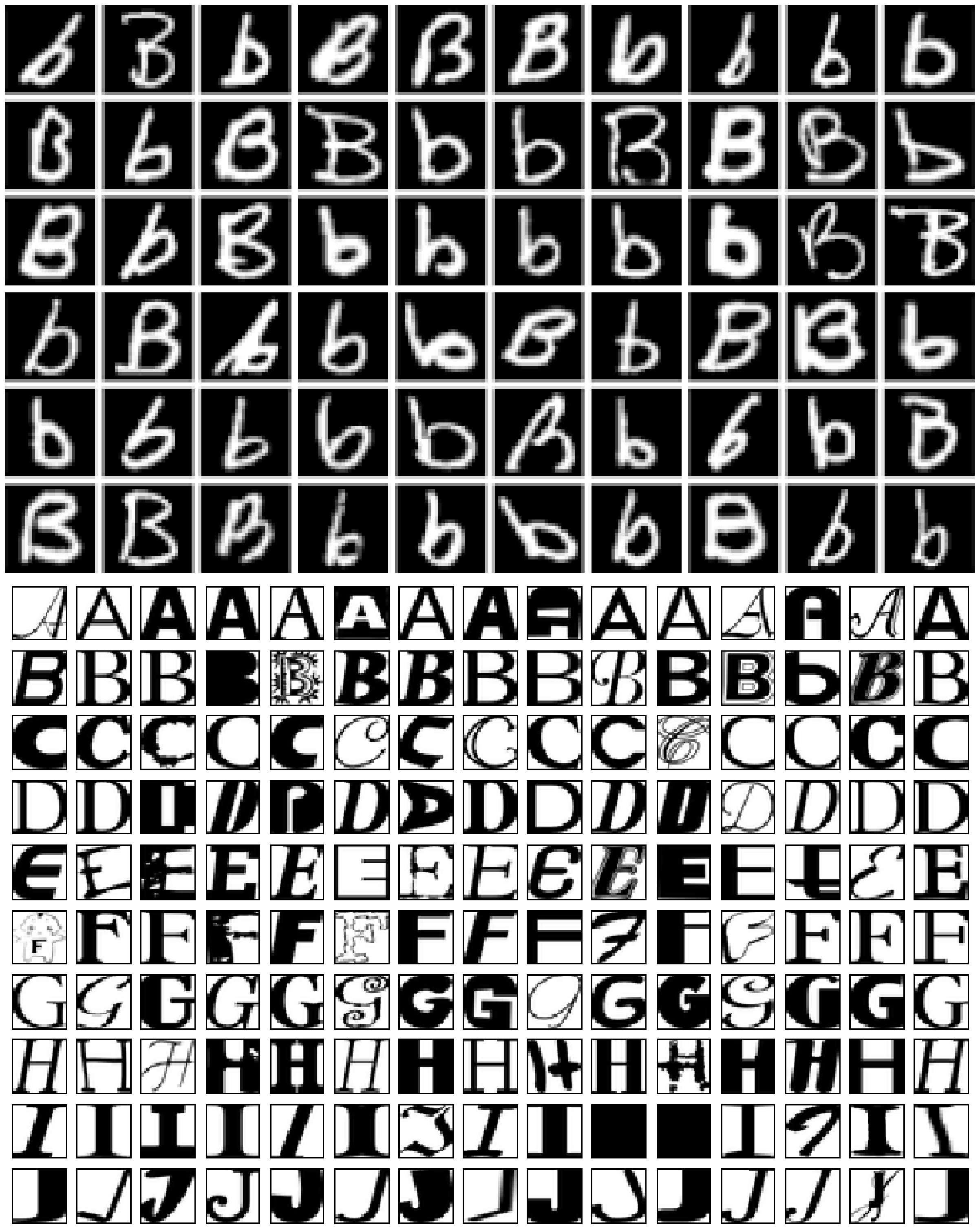


Fig 4.20: Sample of EMNIST dataset.

Figure 4.20 shows the sample of EMNIST dataset.

The features of this database are:

- Final accumulation of NIST's handprinted sample data
- Full page HSF forms from 3600 writers
- Separate digit, upper and lower case, and free text fields
- Over 800,000 images with hand checked classifications

It is a suite of six datasets intended to provide a more challenging alternative to the MNIST dataset. The characters of the NIST Special Database 19 were converted to a format that matches that of the MNIST dataset, making it immediately compatible with any network capable of working with the original MNIST dataset.

4.2.4 Line, Word and Character Segmentation:

It is compulsory to convert the given input text image into lines then to words and then segmented to individual characters, so that we easily recognize these individual characters. For that we required to find connected components. Here we are giving labels to each and every connected component (see figure 4.21 to 24) . The pixels which are connected with the same values are labeled with same number. Horizontal projection and vertical projection are also included here to segment the text into lines and then words. For that histogram is used. Histogram of horizontal and vertical projections is plot. This histogram has peaks and valleys. Peaks means the pixels which are in white color and valley include black portion of image. So that valley gives the boundary of the line and words. Hand writing text recognition requires all three segmentations [36].

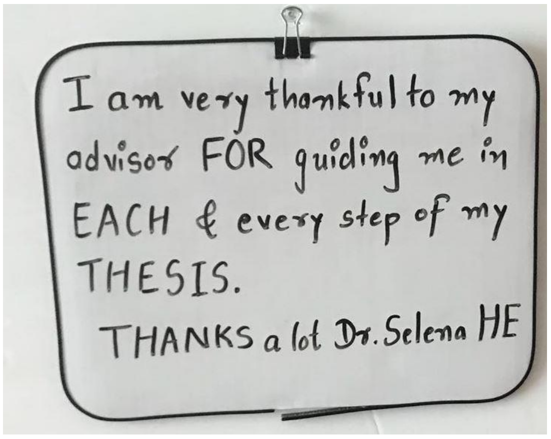

I am very thankful to my
advisor FOR guiding me in
EACH & every step of my
THESIS.
THANKS a lot Dr. Selena HE

Fig 4.21: Original Image

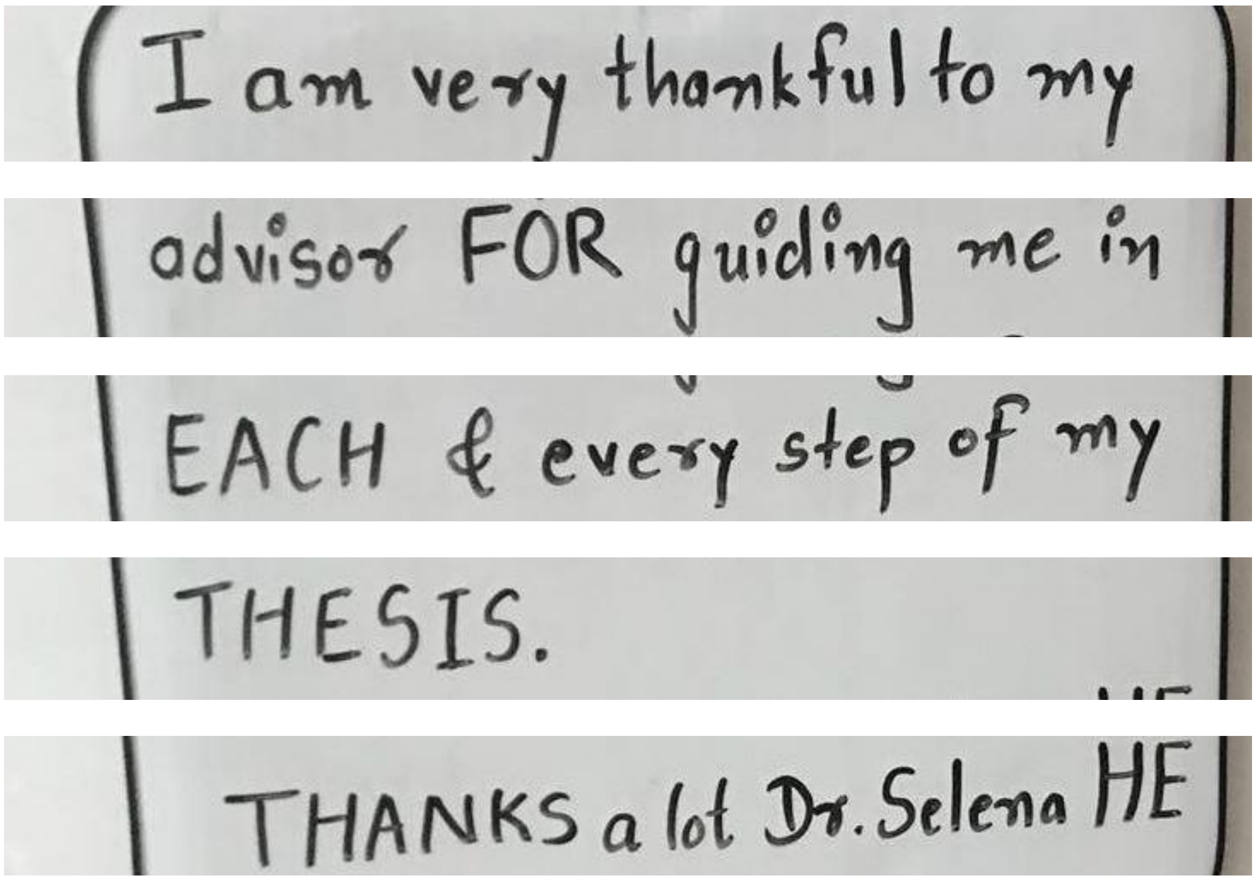

I am very thankful to my
advisor FOR guiding me in
EACH & every step of my
THESIS.
THANKS a lot Dr. Selena HE

Fig 4.22: Line Segmentation

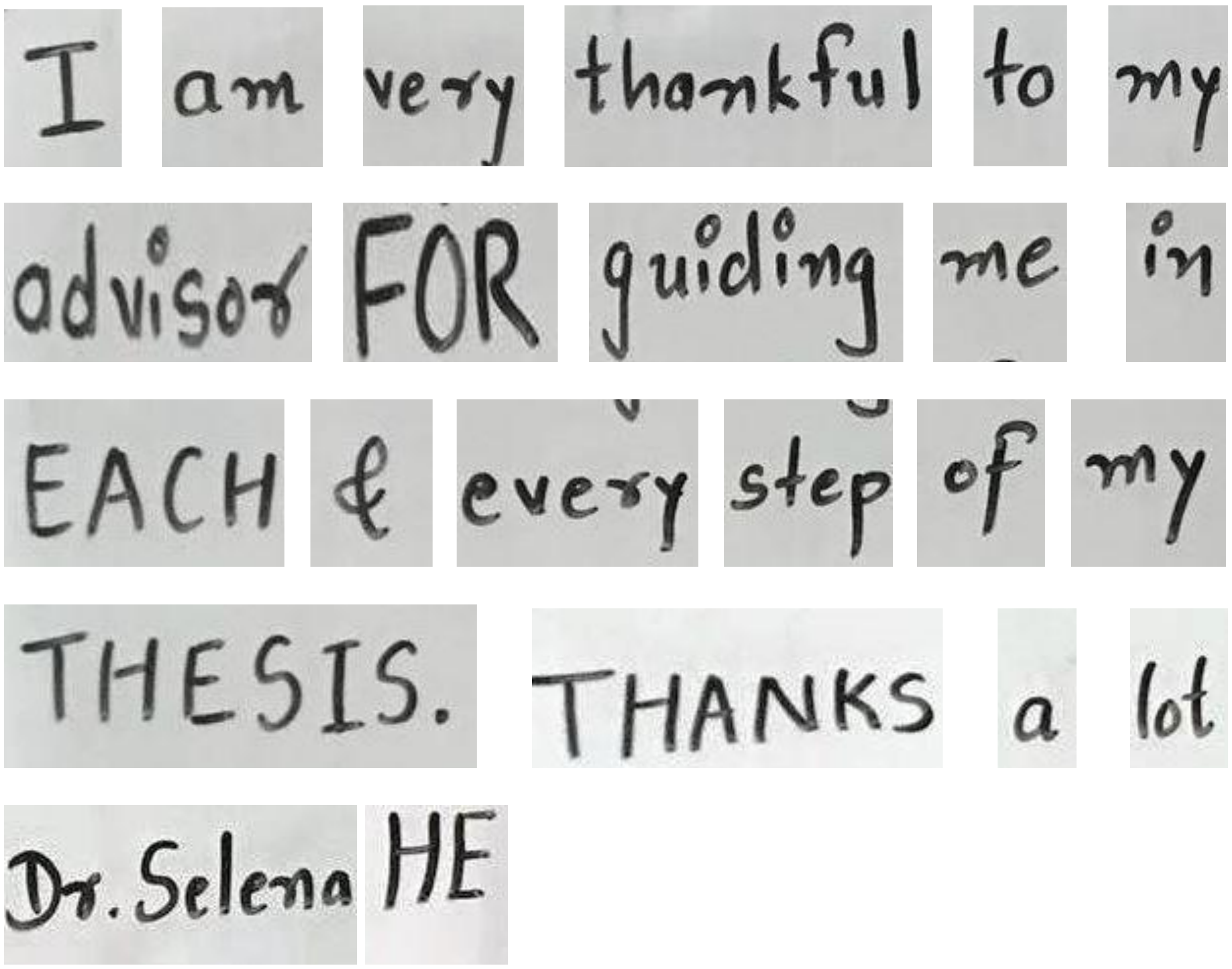


Fig 4.23 : word segmentation

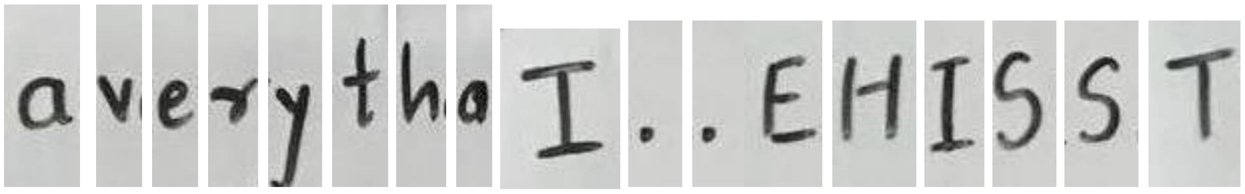


Fig 4.24: Character Segmentation

Figure 4.21 – 4.24 shows the process of Line, word , Character Segmentation. Figure 4.21 show the original image. Figure 4.22 shows the line segmentation of original image. Figure 4.23 shows the word segmentation of original image. Figure 4.24 shows the Character segmentation of original image

4.2.5 Pre-processing:

The gained image is of no utilization in the event that it isn't subjected to the pre-preparing stage. This stage makes the image reasonable for advance examination and work to be performed on that. Pre-preparing incorporates a few application particular strategies. A portion of the systems utilized in this undertaking are:

Binarization

The procedure of binarization [26] includes transformation of examined image into a grayscale image which thusly is changed over into parallel or bi-level picture. This progression is critical to recognize the foundation which is set to white and frontal area, the content to be perceived which is set to dark. Means the background is set to white and the interested region means the text on which we are working is set in black color (see figure 25). To perform this binarization we are using thresholding technique which is already discussed in the gesture recognition.

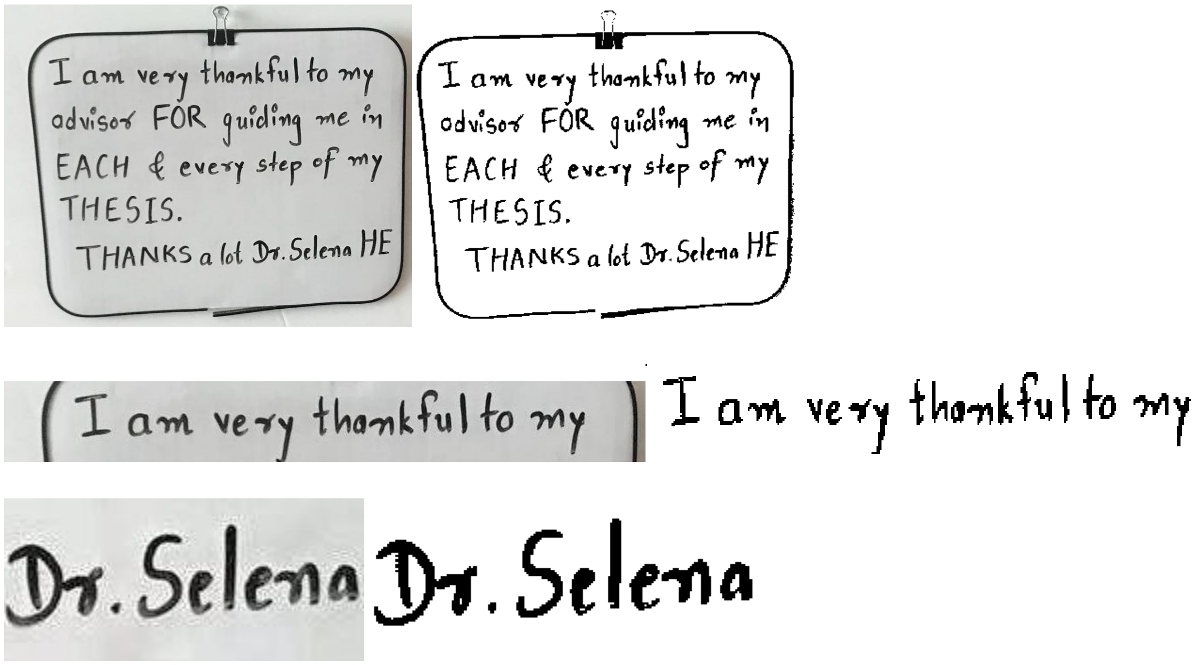

I am very thankful to my advisor FOR guiding me in EACH & every step of my THESIS.
THANKS a lot Dr. Selena HE

Fig 4.25: binarization of original images.

Figure 4.25 shows the binarization of original image.

<u>Gap Correction</u>

In segmentation, the position of the object i.e., the character in the image is found out and the size of the image is cropped to that of the template size**.** Segmentation can be external and internal. External segmentation is the isolation of various writing units, such as paragraphs, sentences or words. In internal segmentation an image of sequence of characters is decomposed into sub-images of individual character.

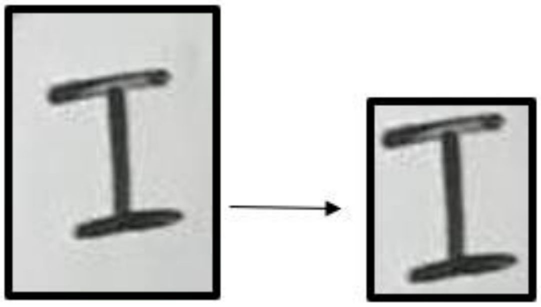


Figure 4.26: Segmented Image

Figure 4.26 shows the segmented Image for original image I.

<u>Scale down/ scale up</u>

The output image of the segmentation step is input to this operation. The input image can be of any size. In convolutional Neural Network, the size of input image is 28x28. To make the image of a fixed size of 28x28, this operation is performed. If the image is smaller than the decided size (28x28), then up sampling (scale up) operation is performed. If the image is bigger than the decided fixed size, then down sampling (scale down) operation is performed. The output of this task is an image of 28x28 size.

4.2.6. Classification through Conventional Neural Network:

After segmenting entire image into single characters, we need to classify that in which class each character belongs. The segmented character may belong to alphabets, numbers or character. Now., let's understand how the classification of character works.

<u>4.2.6.1 Model of Convolutional Neural Network</u>

To classify the character, we have built a Convolutional neural network. The model of convolutional neural network is shown in figure.

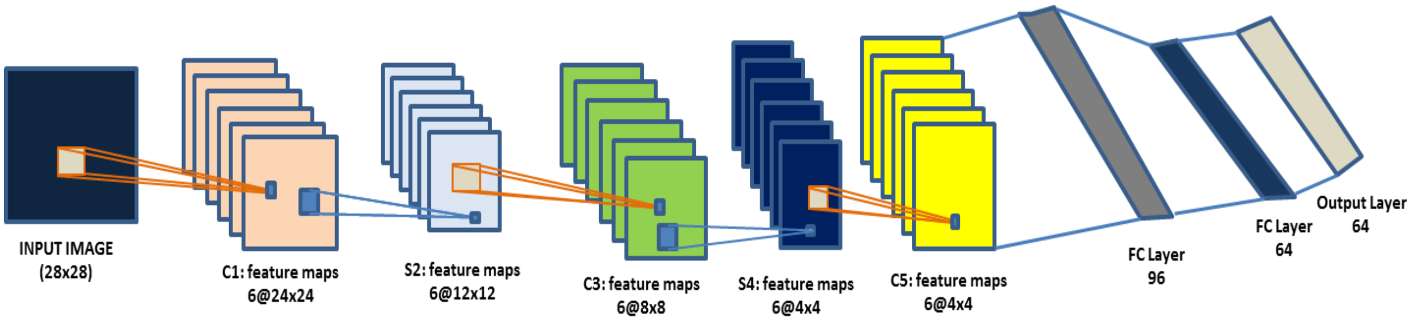


Figure 4.27: Model of Convolutional Neural Network

In figure 4.27, the convolutional layers are labelled Cx, Sub-sampling layers are labelled Sx and Fully connected layers are labelled Fx, where x is the layer index.

As shown in figure 4.27, the model of CNN comprises 8 layers excluding input layer. All of which contains trainable parameters (weights).

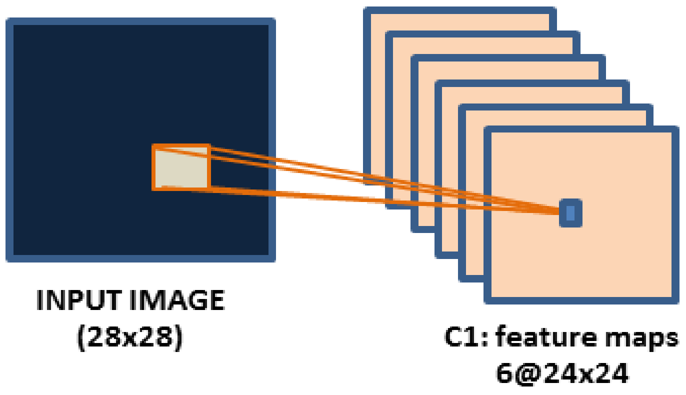


Figure: 4.28 Convolutional layer and input layer

Figure 4.28 shows the Convolutional layer and input layer.

The input is a 28x28 pixel image. This is significantly larger than the largest character in the database. The size of the input is chosen 28x28 because the size of training samples in MNIST and EMNIST dataset is also 28x28. INPUT layer will hold the raw pixel values of the image, in this case an image of width 28, height 28 with single color channel as the input image is pre-processed and binarized. With the local receptive fields (filters), neurons can extract elementary visual features such as oriented edges, end-point corners.

These features are then combined by sub sequent layers in order to detect higher order features. Distortions or shifts of the input can cause the position of salient features to vary. In addition, elementary feature detectors that are useful on one part of the image are likely to be useful across the entire image. This knowledge can be applied by forcing a set of units, whose receptive fields are located at different places on the image, to have identical weight vectors.

Units in a layer are organized in planes within which all the units share same kind of weights. The set of output in of the units in such a plane is called feature map. Unit in a feature map are all constrained to perform the same operation on different parts of the image.

A complete convolutional layer is a composed of several feature maps (with different weight vectors), so that multiple features can be extracted at each location. As shown in figure

4.30, a unit in a feature map has 25 inputs connected to a 5 by 5 area in the input called the receptive field of the input. Each unit has 25 inputs and therefore 25 trainable coefficients plus a trainable bias. The receptive fields of the contiguous units in a feature map are central on correspondingly contiguous units in the previous layer. Therefore, receptive fields of the neighboring field overlap.

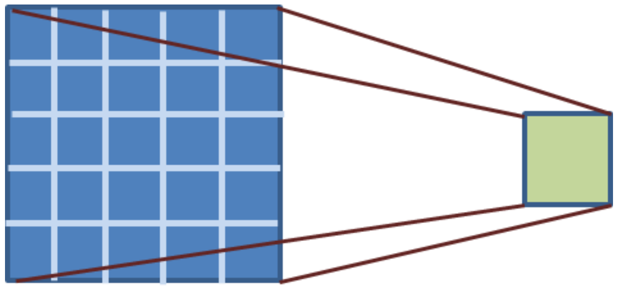


Figure 4.29: 5x5 input connected to single unit of a feature map.

Figure 4.29 shows the 5*5 input connected to single unit of a feature map.

All the units in the feature map share the set of 25 weights and the same bias so it detects the same features at all the possible locations on the input. The other feature maps in the same layer uses different sets of weights and biases, thereby extracting different types of features from the same image is possible. In our model, at each input location 6 different types of features are extracted by 6 units in identical locations in the 6 feature maps. With the local receptive fields, neurons can extract elementary visual features such as oriented edges, end-point corners.

A sequential implementation of a feature map would scan the input image with a single unit that has a local receptive field, and store the states of this unit at corresponding locations in the feature map. This operation is equivalent to a convolution, followed by an additive bias and a squashing function.

The kernel of the convolution is the set of connection weights used by the units in the feature map. Layer C1 is a convolutional layer with 6 feature maps. Each unit in each feature map is connected to a 5x5 neighborhood in the input.

C1 contains 156 [(5x5x6) +6] trainable parameters. Where 5x5 is the size of receptive field. There are 6 receptive fields (filters) and for every receptive field there is one trainable coefficient called bias. Bias is used to adjust the values of the training parameters. The size of the feature map is 24x24 which prevents connection from the input from falling of the boundary. Once a feature has been detected, its exact location becomes less important. Only its approximate position relative to another feature map is relevant.

A simple way to reduce precision with which the positions of the distinctive features are encoded in a feature map is to reduce the spatial resolution of the feature map. This can be achieved with a so- called sub-sampling layer which performs a local averaging and sub sampling, reducing the resolution of the feature map and reducing the sensitivity of the output to shifts and its distortions. Sub sampling layer is also known as POOL layer. Here max pooling is used.

Layer S2 is a sub-sampling layer with 6 feature maps of size 12x12. This layer comprises feature maps, one for each feature map in the previous layer. Each unit in each feature map is connected to a 2x2 neighborhood in the corresponding feature map in C1. The four inputs to a unit in S2 are averaged, then multiplied by a trainable coefficient, and added to a trainable bias. The result is passed through a sigmoidal function. The 2x2 receptive field are non-overlapping therefore feature maps in S2 have half the number of rows and columns as feature maps in C1.

The trainable coefficient and bias control the effect of the sigmoid non-linearity. If the coefficient is small then the unit operates in a quasi-linear mode and the sub sampling layer merely blurs the input. If the coefficient is large, sub sampling units can be seen as performing a “noisy OR” or a “noisy AND” function depending on the value of the bias. Layer S2 has 70 trainable parameters.

Layer C3 is a conventional layer with 6 feature maps of size 8x8. Each unit in each feature map is connected to several 5x5 neighborhoods at identical locations in a subset of S2's feature map.

Layer S4 is a sub-sampling layer with 6 feature maps of size 4x4. Each unit in each feature map is connected to a 2x2 neighborhood in the corresponding feature map in C3, in a similar way as C1 and S2.

Layer C5 is a conventional layer with 6 feature maps of size 4x4. This layer comprises feature maps, one for each feature map in the previous layer. From pool layer to convolutional layer C5 mapping is 2x2.

Layer 6 and 7 are fully connected layers. First fully connected layer contains 96 neurons and second fully connected layer contains 64 neurons.

Since, all the weights are learned with back- propagation, convolutional networks can be seen as synthesizing their own feature extractor. The weight sharing technique has the interesting side effect of reducing the capacity of the machine and reducing the gap between test errors and training error.

As in classical neural networks, units in layers up to F7 compute a dot product between their input vector and their weight vector. And a bias is also added to balance the output.

This weighted sum denoted as a_i for unit i, is then passed through a sigmoid squashing function to produce the state of unit I, denoted by x_i

$$x_i = f(a_i) \quad \text{Eq. (14)}$$

The squashing function is a scaled hyperbolic tangent.

$$F(a) = A \tanh(Sa) \quad \text{Eq. (15)}$$

Where A is a amplitude of the function and S determines the slope of the origin. The function f is odd, with horizontal asymptotes at +A and –A. The constant A is chosen to be 10 x e^{-6}.

Finally, the output layer contains 64 neurons. After experimenting the batch sizes. 64 batch size is chosen. The SoftMax function is often used in the final layer of a neural network-based classifier [34]. SoftMax Regression is a generalization of logistic regression that we can use for multi-class classification.

These are commonly trained under a log loss (or cross-entropy) regime, giving a non-linear variant of multinomial logistic regression [34].

$$P(y=j \mid z^{(i)}) = \phi_{softmax}(z^{(i)}) = \frac{e^{z^{(i)}}}{\sum_{j=0}^{k} e^{z_k^{(i)}}}, \qquad \text{Eq. (16)}$$

where we define the net input z as

$$z = w_0x_0 + w_1x_1 + \ldots + w_mx_m = \sum_{l=0}^{m} w_lx_l = \mathbf{w}^T\mathbf{x}. \qquad \text{Eq. (17)}$$

Here w is the weight vector, x is the feature vector of 1 training sample, and w0 is the bias unit.

4.2.6.2 Training , testing and validation Module

In training module, we have taken 100,000 images from MNIST dataset and 100,000 images from EMNIST dataset. We total 200,000 images and 64 classes hence, each class will have approximately 3000 images. The number epochs (no. of iteration we need to train a machine) our machine needed to generate proper result is nearly 1000. Our batch size to train a machine is 64. The batch size is defined as number of images needed to train. The learning rate of my machine is 10*e^{-6}. Batches , epochs and learning rate all are hyper parameters. We have tried training machine first with 16 batch size then 32 then 64. For batch size 64 we got desired output. Similarly for machine learning rate we got best results for e^{-6} after seeing results for e^{-2} , e^{-4}.

Testing module deals with the test images. Test images are obtained by splitting the dataset. Here MNIST and EMNIST datasets are used to train the conventional neural network. Total 200,000 images are used for training and 20,000 images are used for testing purpose. It will first pre-process the input image and it will classify the unlabeled test data. Test data is not labelled in the sense it should be recognized by the machine. Labels are assigned to each of test images by the network and then the accuracy is measured.

In the post processing stage of the proposed system, CNN outputs mapped to the character Unicode. The output of the classifier will be some integer labels. This integer label should be converted into corresponding character Unicode. The Unicode is written in a text file.

4.3 Mathematic Equations

To convert mathematical equation into text we are using steps shown in figure.

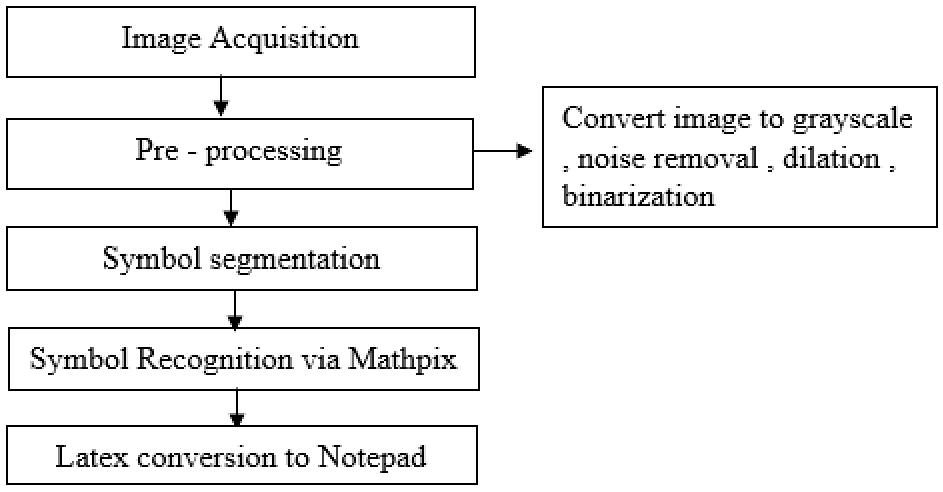


Image Acquisition
Pre - processing
Convert image to grayscale , noise removal , dilation , binarization
Symbol segmentation
Symbol Recognition via Mathpix
Latex conversion to Notepad

Fig 4.30 Image to text conversion for mathematical equation.

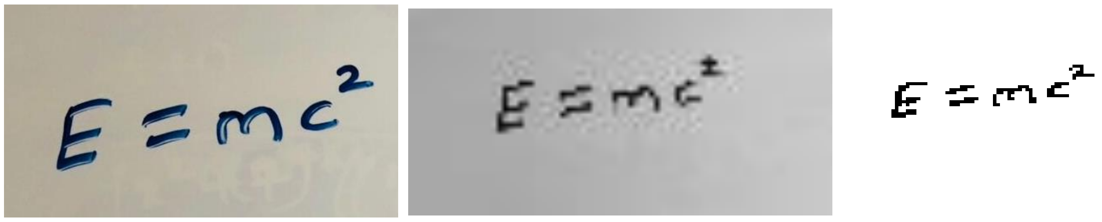
Figure 4.30 shows the procedure followed to convert mathematical equations into text.

4.3.1 Image Acquisition

To capture images, we are using Raspberry Pi camera, when lecturer shows three fingers Raspberry Pi will take a picture of white board.

4.3.2 Pre- processing

After picture is captured we are converting image to grayscale, and after converting to grayscale , we are removing noise with median filter , then we binarization a picture (see figure).



(a) (b) (c)

Fig 4.31 Pre-processing Task

Figure 4.31 shows the pre-processing task of image processing. Figure 4.31(a) shows the original image. Figure 4.31(b) shows the gray scale image. Figure 4.31 (c) shows the binary image.

4.3.3 Symbol Segmentation

To segment each and every symbol we are applying canny edge detector by using OpenCV (see figure 4.32).

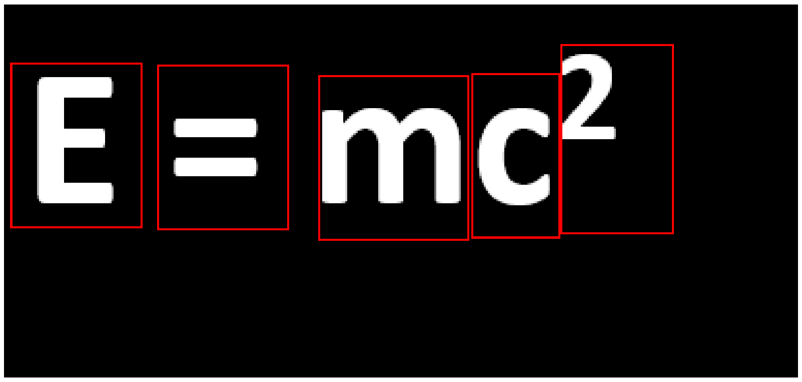


Fig 4.32 Symbol Segmentation

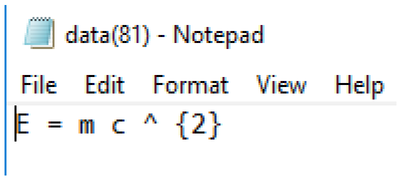
Figure 4.32 shows the canny edge detector segmentation for all the symbols.

4.3.4 Symbol recognition

For recognizing segmented symbols, we are using “Mathpix” tool [55]. Mathpix is an open source OCR engine. We integrated it into our system to identify the symbol within every segmented symbol box.

4.3.5 Image to text conversion.

After recognizing each symbol, the entire mathematical equation is provided in latex format in notepad.



```
E = m c ^ {2}
```

Fig 4.33

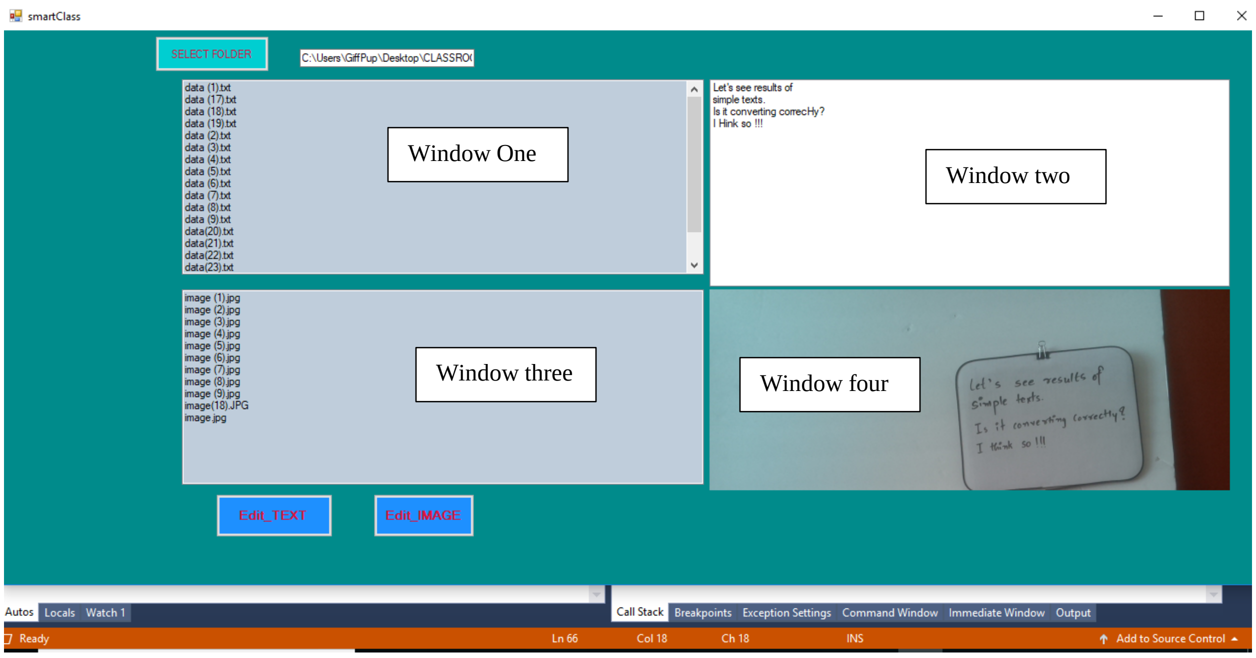
Figure 4.33 shows that $E=mc^2$ equation is converted into latex format and retrieved in notepad file.

4.4 Text file Conversion

After recognizing each character and combining all characters, words and line together next step is to put all the results in text file. All the extracted text has been stored in the text file. In our system we call text file as a data file. The Raspberry Pi, google cloud and desktop application is connected through one network. Figure 3.12 shows the diagram of IOT system. Raspberry Pi stores all the datafiles in folder named “CLASSROOM” which is in google drive. Students laptops are synced with same internet connection and same google drive account in which Raspberry Pi is connected. Hence, students can directly access to CLASSROOM folder from google drive and can retrieve all datafiles and input images through desktop application.

__Desktop Application__

Desktop application is specially designed for students (see figure 4.34). It is designed in VB.net. For our system we have made the desktop application which can only support windows machines. Through desktop application students can retrieve data files and input images in their laptop. Desktop application facilitates students to edit data files and images both. Desktop application has two buttons “EDIT_TEXT” and “EDIT_IMAGE” button. If student clicks on “EDIT_TEXT” button then desktop application will open notepad to edit text file. If students click on “EDIT_IMAGE” button then paint will open to edit an image. After editing they can store their file to laptop or google drive CLASSROOM folder. Hence, the desktop application allows students to get lecture content on their personal device within fraction of time.



C:\Users\GiffPup\Desktop\CLASSRO(

data (1).txt
data (17).txt
data (18).txt
data (19).txt
data (2).txt
data (3).txt
data (4).txt
data (5).txt
data (6).txt
data (7).txt
data (8).txt
data (9).txt
data(20).txt
data(21).txt
data(22).txt
data(23).txt

Let's see results of
simple texts.
Is it converting correcHy?
I Hink so !!!

image (1).jpg
image (2).jpg
image (3).jpg
image (4).jpg
image (5).jpg
image (6).jpg
image (7).jpg
image (8).jpg
image (9).jpg
image(18).JPG
image.jpg

Let's see results of
simple texts.
Is it converting correctly?
I think so !!!

Fig 4.34 Desktop Application

Figure 4.34 shows the screenshot of desktop application. Window one shows the list of data files retrieved. Window two shows the data file content. Window three shows the list of input images retrieved. Windows four shows the input images content. Down to all windows there are two buttons “EDIT_TEXT” and “EDIT_IMAGE” which helps in student edit the text and image in editing.

CHAPTER 5

RESULTS

5.1. Entire system results

This subsection shows the arrangement of entire system and result of all the components of system architecture.

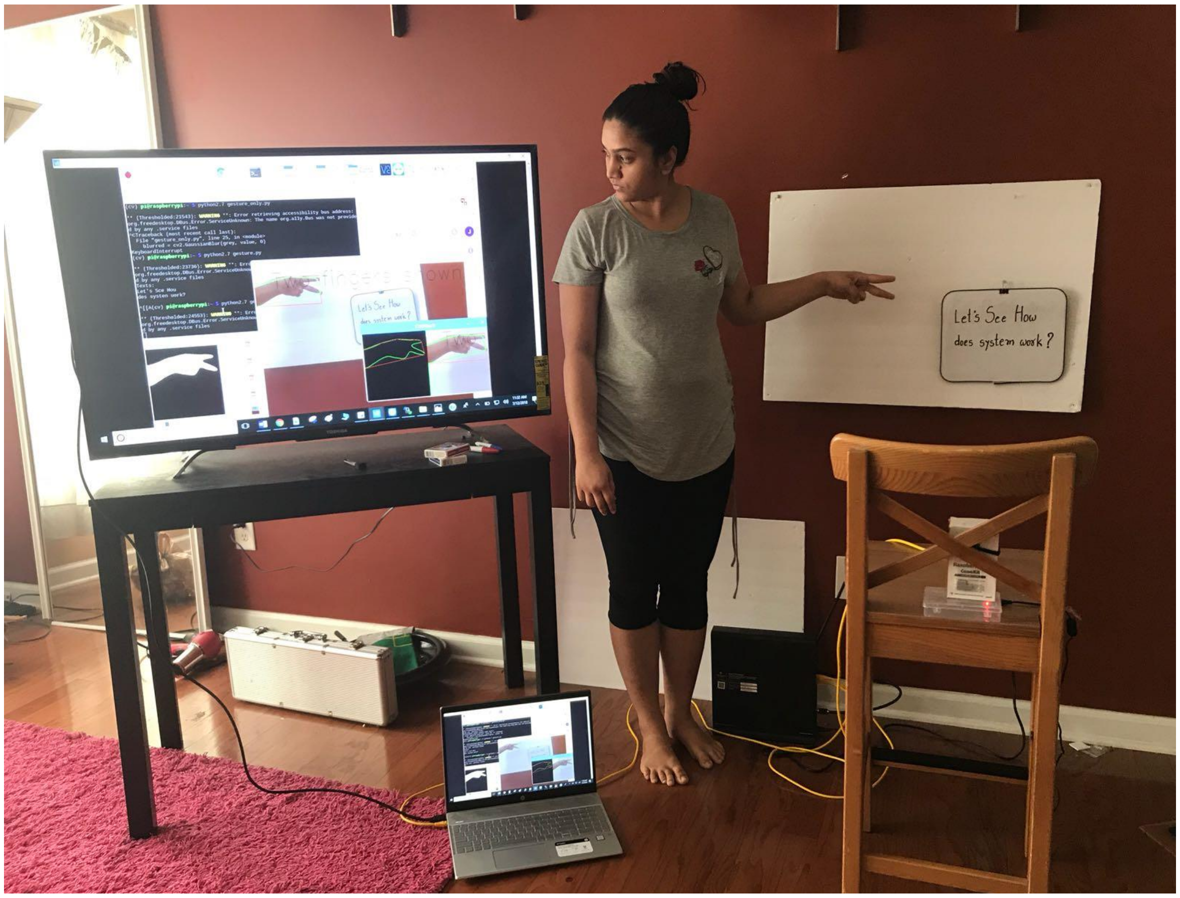
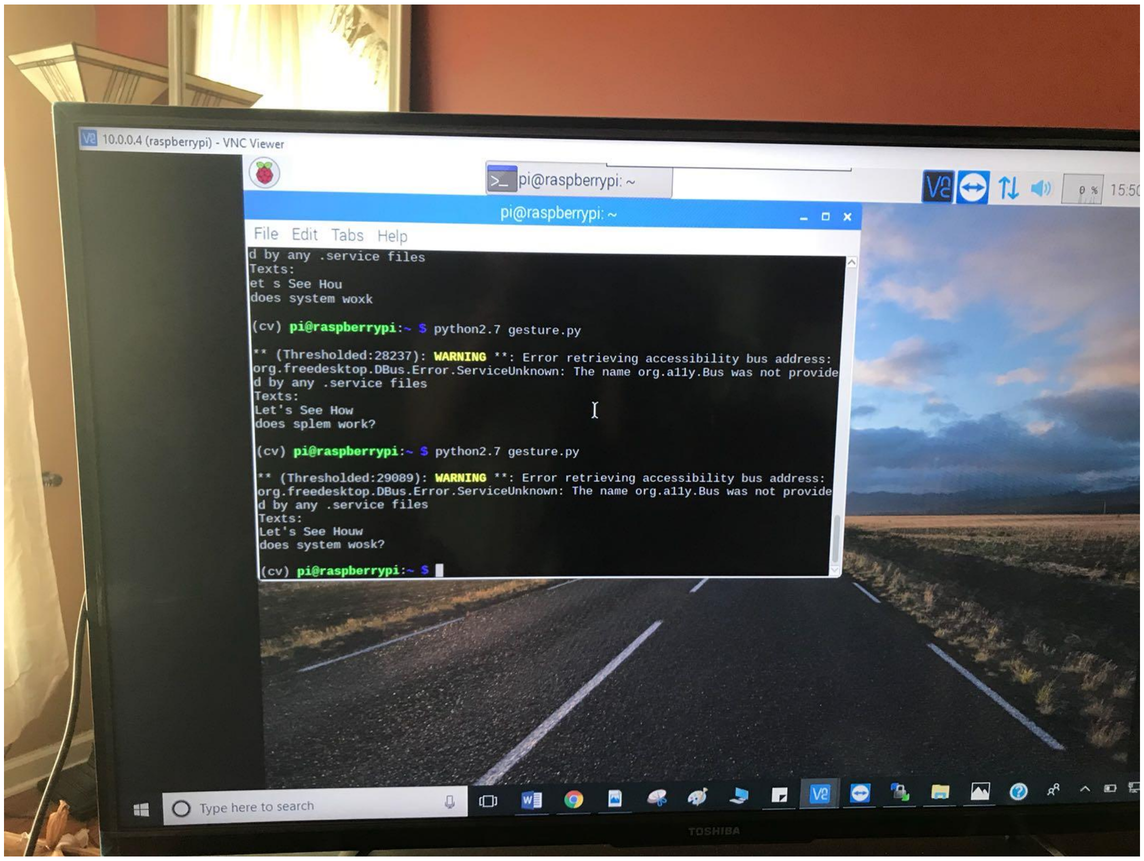


Fig 5.1 : Arrangement of entire system

Figure 5.1 shows the arrangement of white-board and Raspberry Pi. Also, this figure shows the gesture recognition implemented in this system. The text written on white board will converted to text and for that results are attached after this image.



```
d by any .service files
Texts:
et s See Hou
does system woxk

(cv) pi@raspberrypi:~ $ python2.7 gesture.py

** (Thresholded:28237): WARNING **: Error retrieving accessibility bus address:
org.freedesktop.DBus.Error.ServiceUnknown: The name org.a11y.Bus was not provide
d by any .service files
Texts:
Let's See How
does splem work?

(cv) pi@raspberrypi:~ $ python2.7 gesture.py

** (Thresholded:29089): WARNING **: Error retrieving accessibility bus address:
org.freedesktop.DBus.Error.ServiceUnknown: The name org.a11y.Bus was not provide
d by any .service files
Texts:
Let's See Houw
does system wosk?

(cv) pi@raspberrypi:~ $
```

Fig 5.2 Image to text conversion

Figure 5.2 shows the text extracted from the white board text shown in previous image.

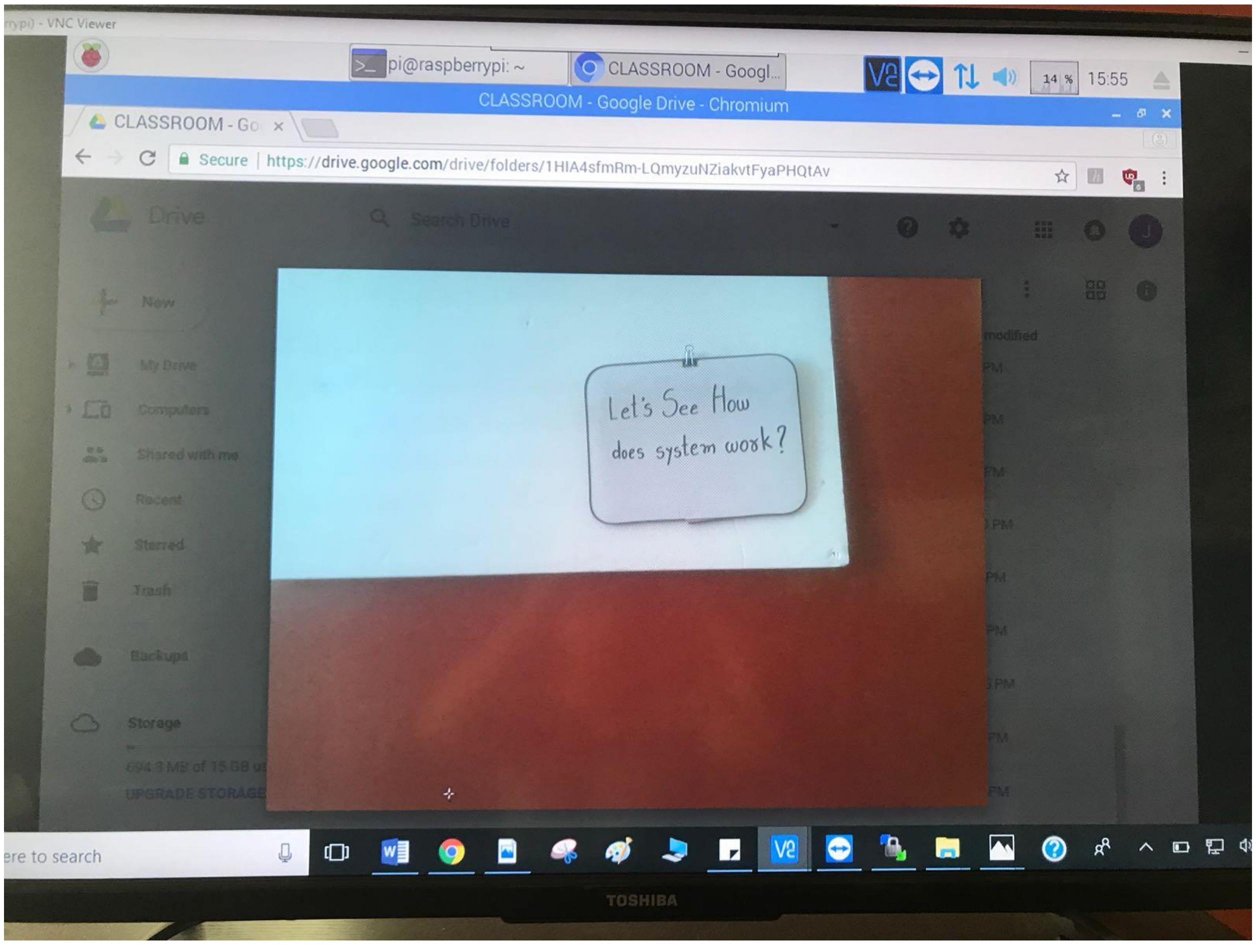
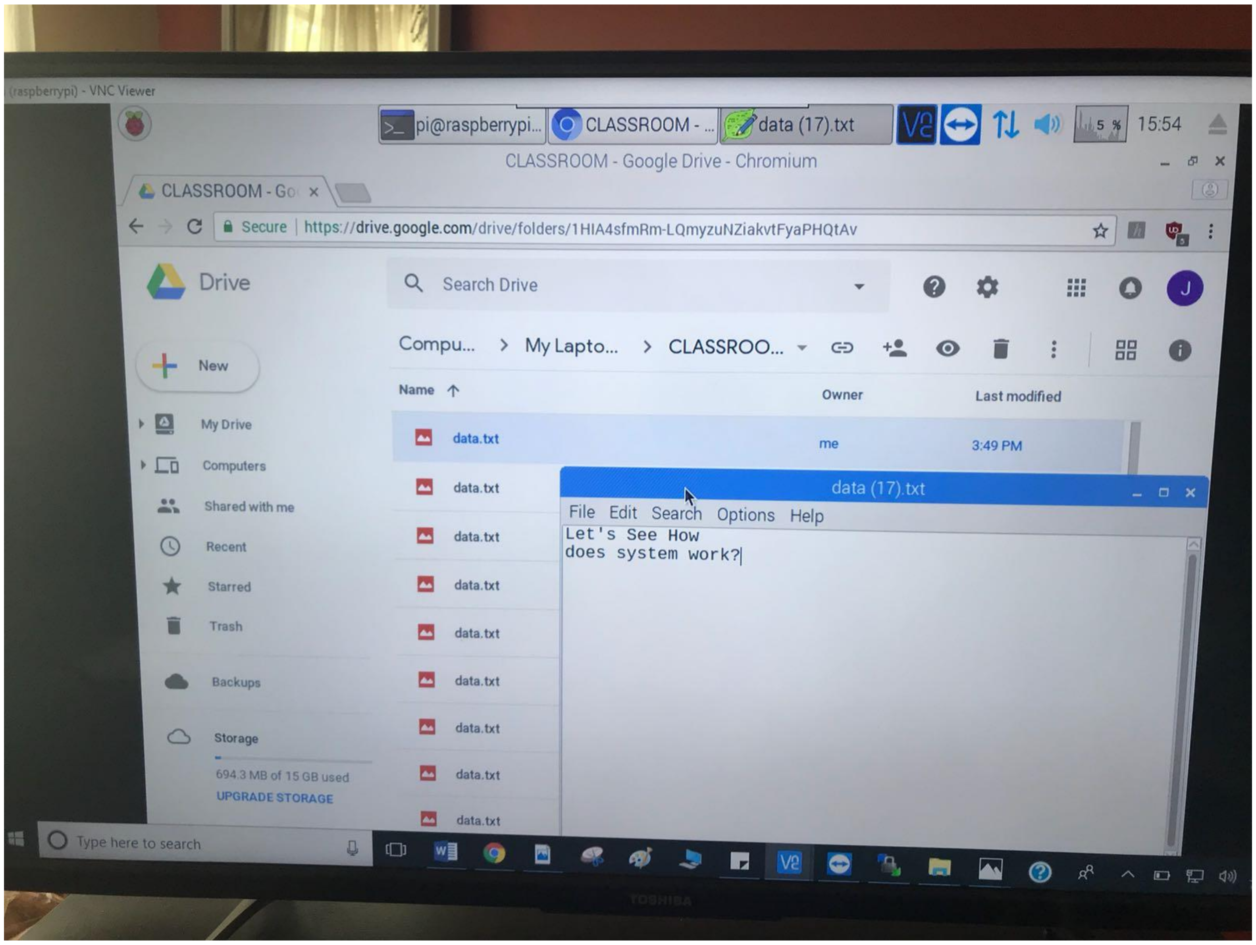

pi@raspberrypi: ~
CLASSROOM - Google Drive - Chromium
https://drive.google.com/drive/folders/1HIA4sfmRm-LQmyzuNZiakvtFyaPHQtAv
Let's See How does system work?
TOSHIBA

Fig 5.3 Image captured by Raspberry Pi

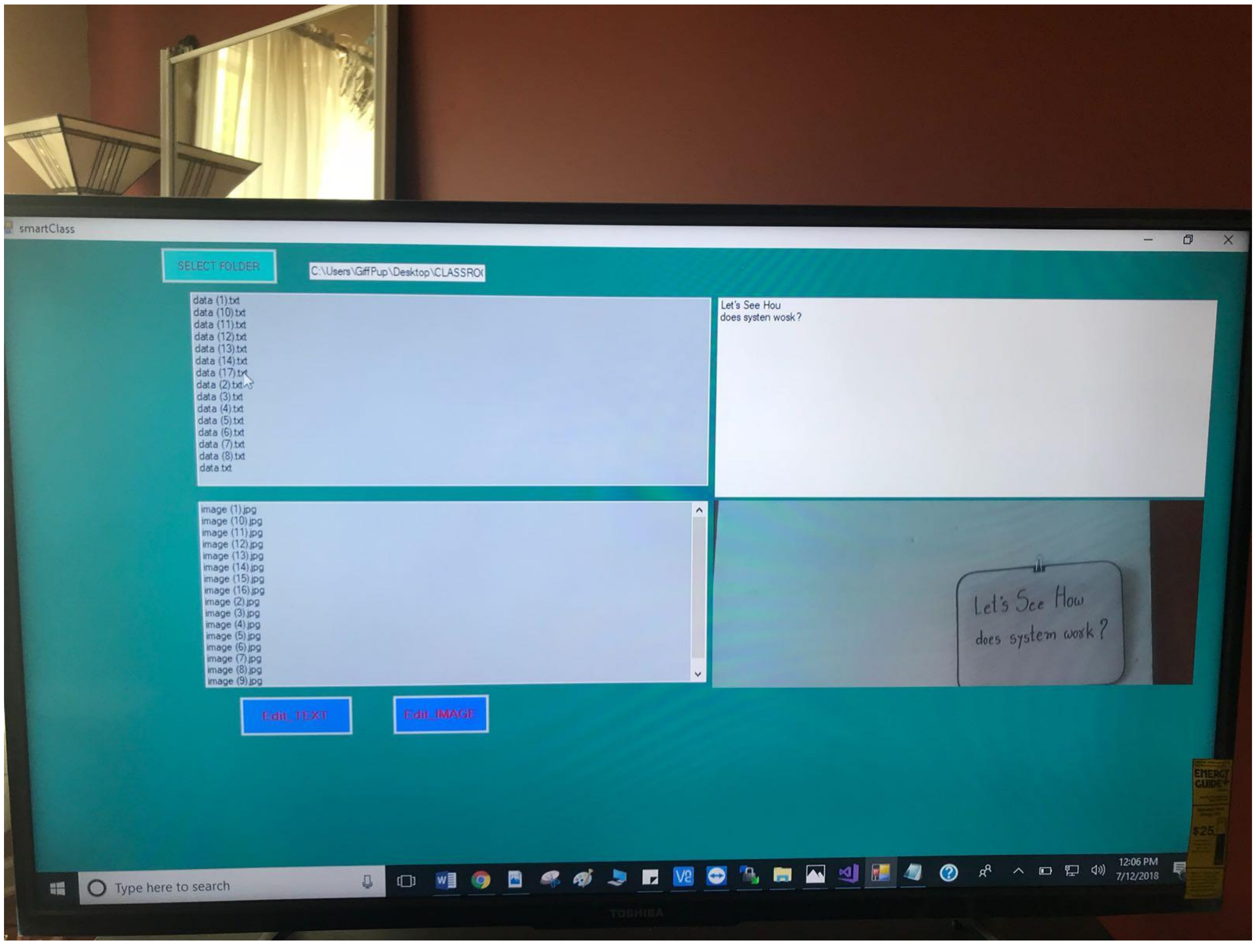
Figure 5.3 shows the white board picture clicked through Raspberry Pi after showing gesture. This images is stored in google drive folder CLASSROOM. We can see that image is opened from CLASSROOM folder of google drive.


CLASSROOM - Google Drive - Chromium
https://drive.google.com/drive/folders/1HIA4sfmRm-LQmyzuNZiakvtFyaPHQtAv
Drive
Search Drive
Compu... > My Lapto... > CLASSROO...
Name
Owner
Last modified
data.txt
me
3:49 PM
data (17).txt
File Edit Search Options Help

```
Let's See How
does system work?
```

Fig 5.4 Data File

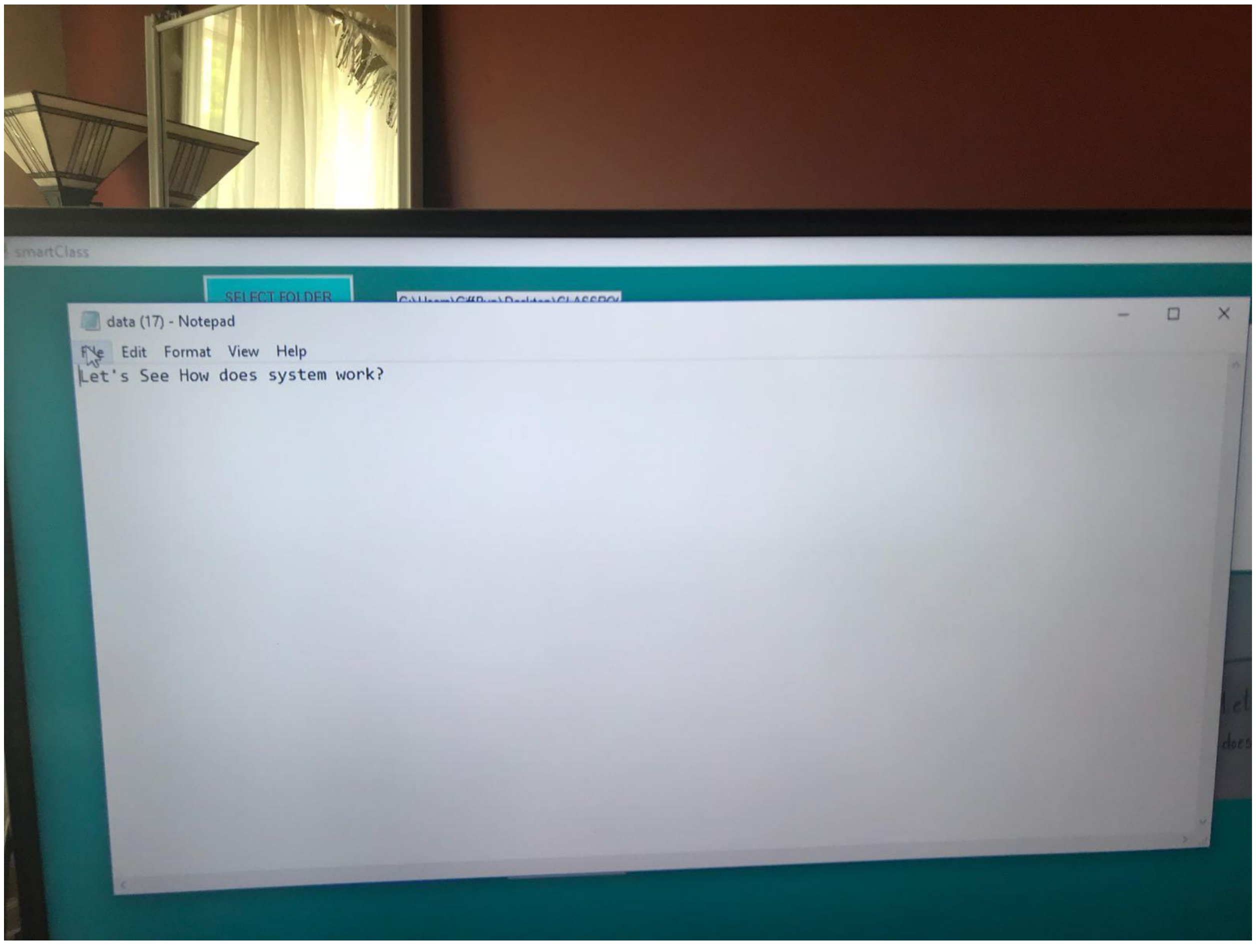
Figure 5.4 shows the data file which contains extracted text from the image. This data file is stored in google drive folder CLASSROOM. We can see that data file is opened from CLASSROOM folder of google drive.



```
C:\Users\GiffPup\Desktop\CLASSRO(

data (1).txt
data (10).txt
data (11).txt
data (12).txt
data (13).txt
data (14).txt
data (17).txt
data (2).txt
data (3).txt
data (4).txt
data (5).txt
data (6).txt
data (7).txt
data (8).txt
data.txt

image (1).jpg
image (10).jpg
image (11).jpg
image (12).jpg
image (13).jpg
image (14).jpg
image (15).jpg
image (16).jpg
image (2).jpg
image (3).jpg
image (4).jpg
image (5).jpg
image (6).jpg
image (7).jpg
image (8).jpg
image (9).jpg

Let's See Hou
does systen wosk?
```

Fig 5.5 Desktop Application

Figure 5.5 shows the desktop application built for students. In this application students will get white board picture taken from Raspberry Pi camera and data files. This application allows students to edit data files and input images.



Let's See How does system work?

Fig 5.6 Function of “EDIT_TEXT” button

Figure 5.6 shows that when students click on “ EDIT _ TEXT” button of desktop application then application redirects to notepad to edit the datafile.

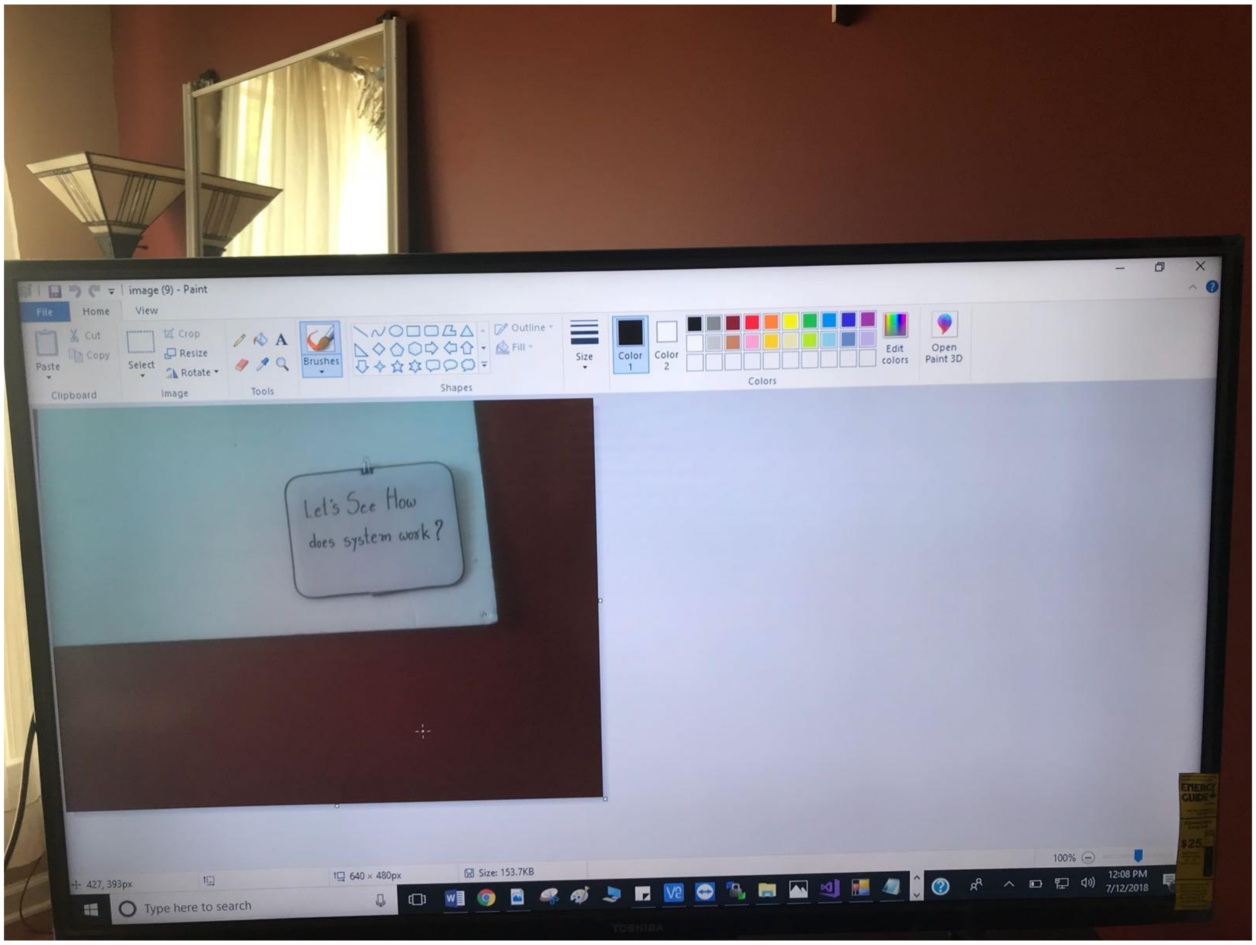


Fig 5.7 Function of "EDIT_IMAGE" button

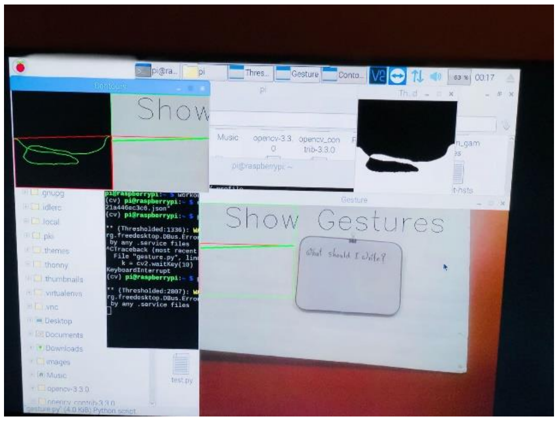
Figure 5.7 shows that when students click on " EDIT_IMAGE" button application redirects to paint to edit the input image.

6.2 Gesture Recognition

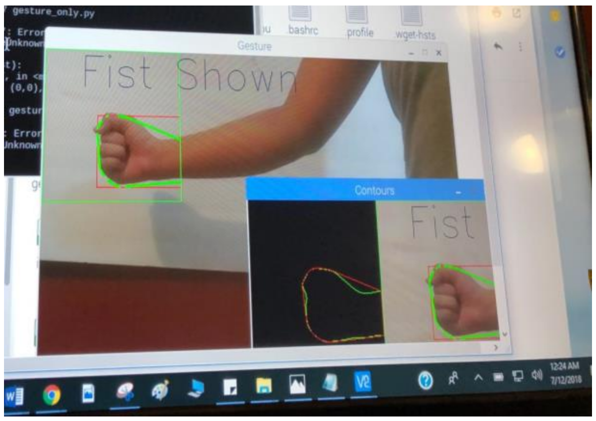
We have implemented convexity defect approach to recognize all the fingers. We have 31 combinations of hands and when do not show gesture then we will have clear background. Hence, we have total 32 test cases for hand gesture recognition.

6.2.2 When Convexity defect is zero:

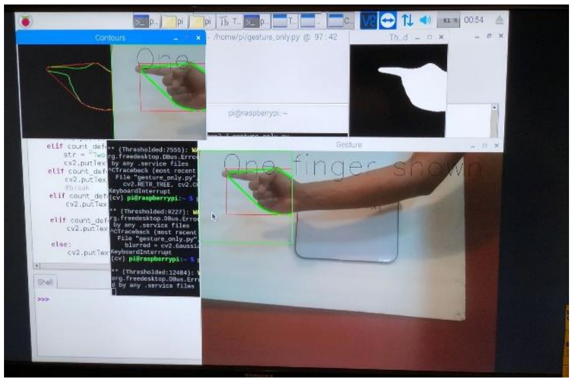
When we have convexity defect as zero then we can say that either the background is clear one finger shown or fist shown. For one finger we have 5 combinations. Among those 5 combinations we differentiate thumb and finger by area ratio.



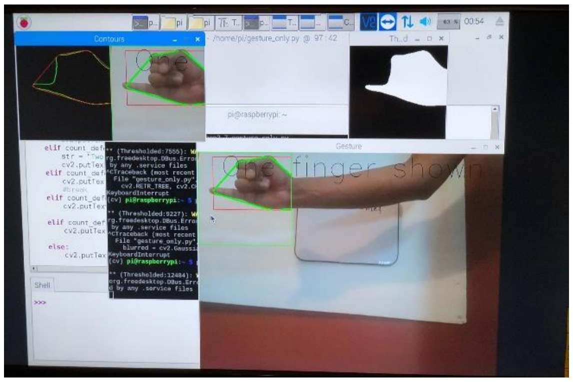
(a)



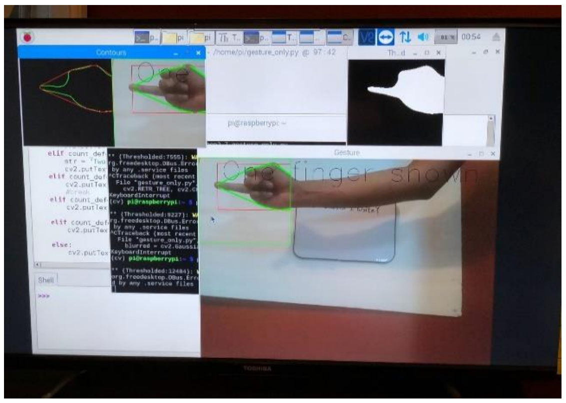
(b)



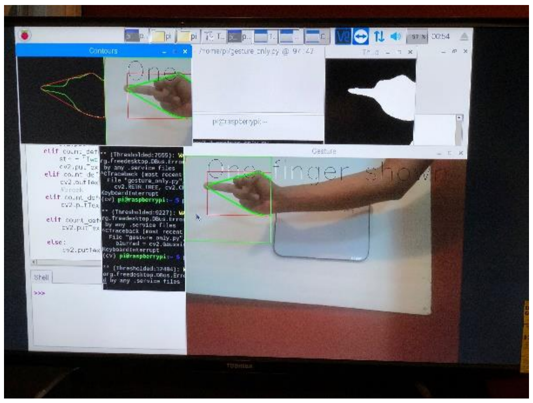
(c)



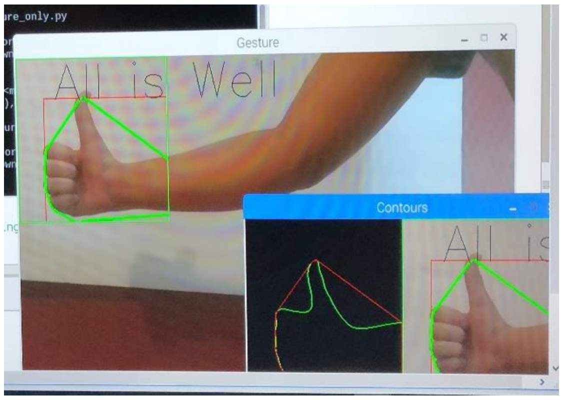
(d)



(e)



(f)



(g)

Figure 5.8 (a-g) shows clear background, fist and one figure hand gestures

Figure 5.8 (a) shows when there is no gesture in “Gesture input box” the message printed is: “show gesture”.

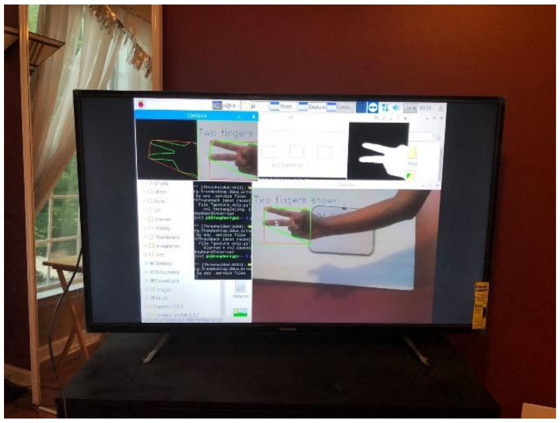
Figure 5.8 (b) shows the fist gesture.

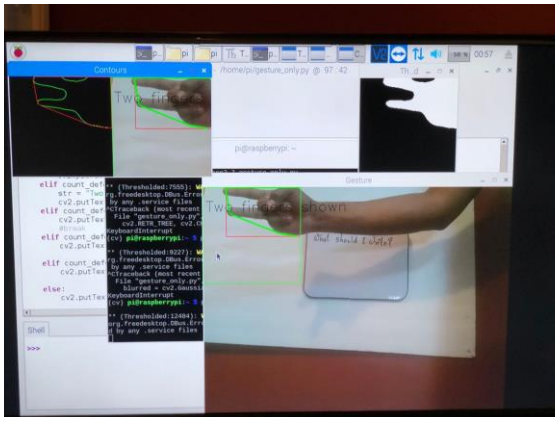
Figure 5.8 (c ,d ,e , g) shows   the all combinations of figure one.

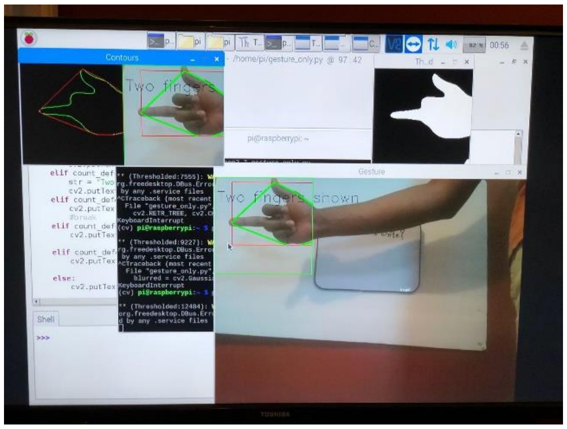
Figure 5.8 (g) shows when the thumb is detected it is printing message “ All is well ”

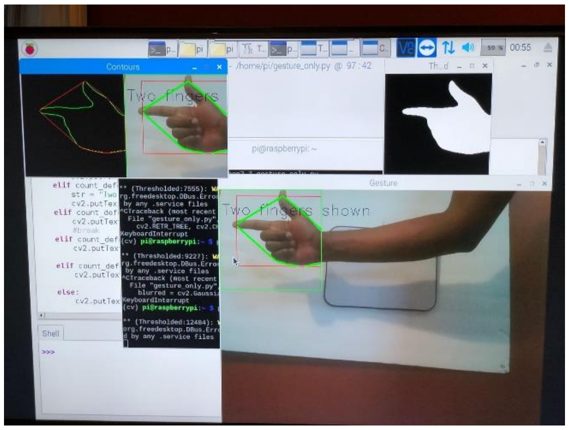
6.2.3 Two fingers shown when convexity defect is 1:

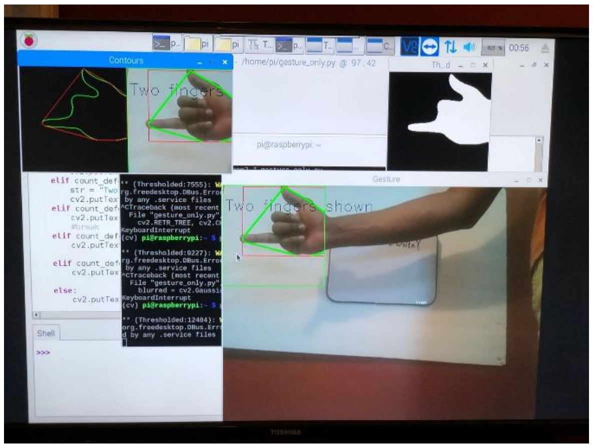
All the picture shown in section 6.2.3 are all 10 combinations of finger two.

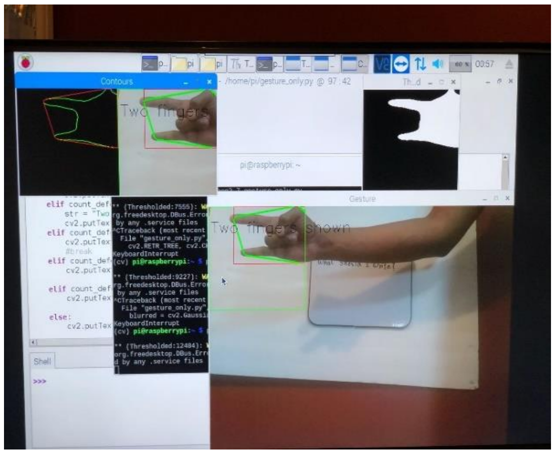


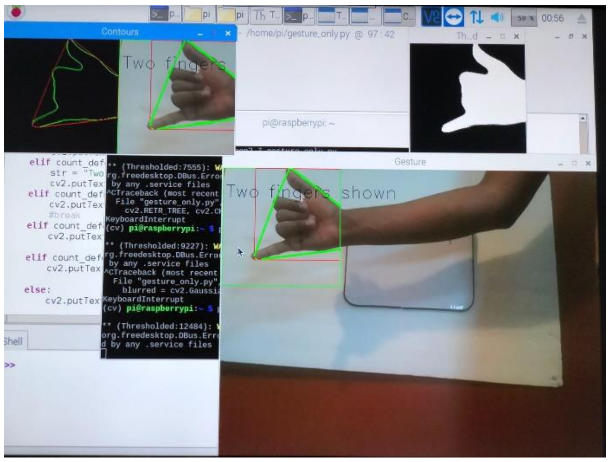


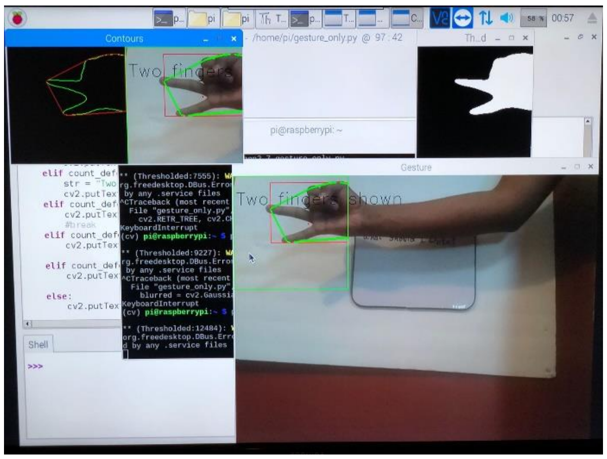


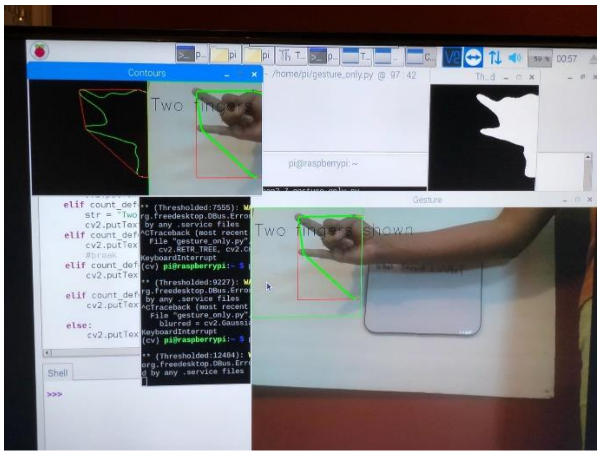


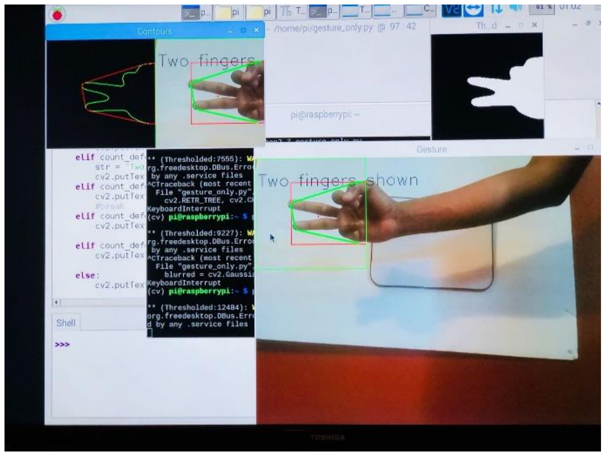




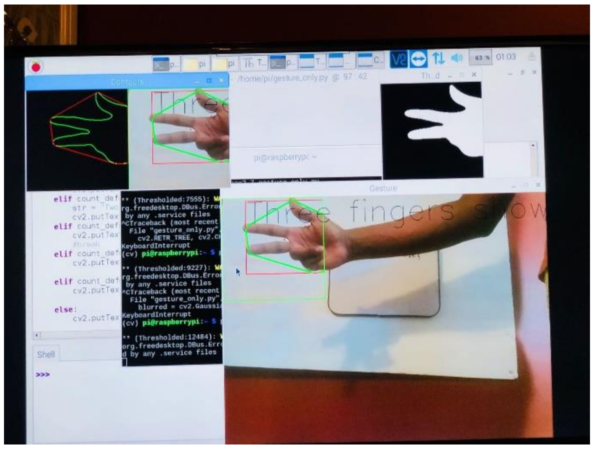


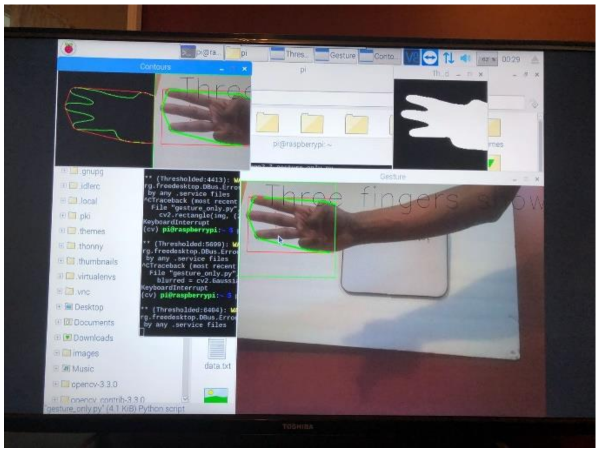


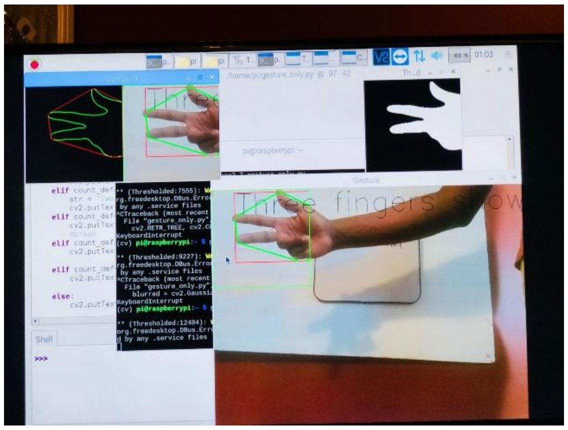
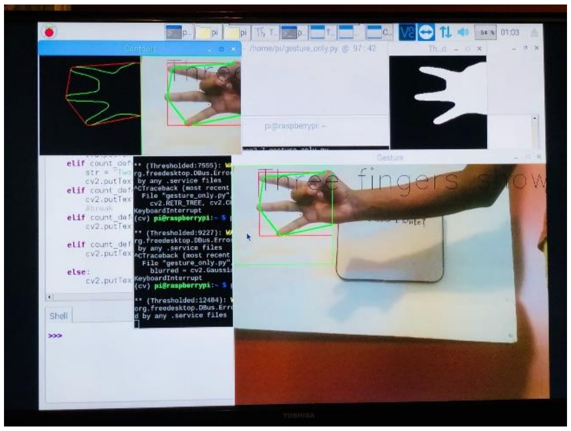
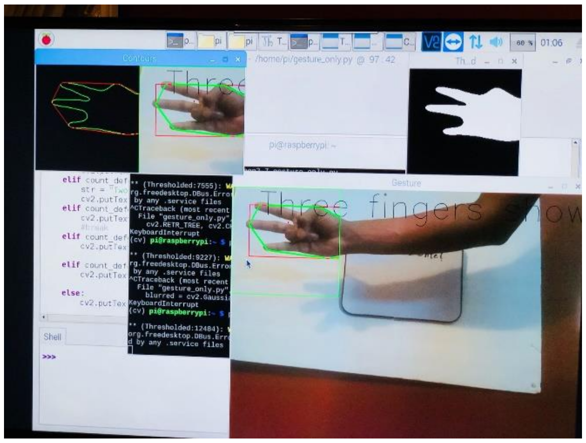
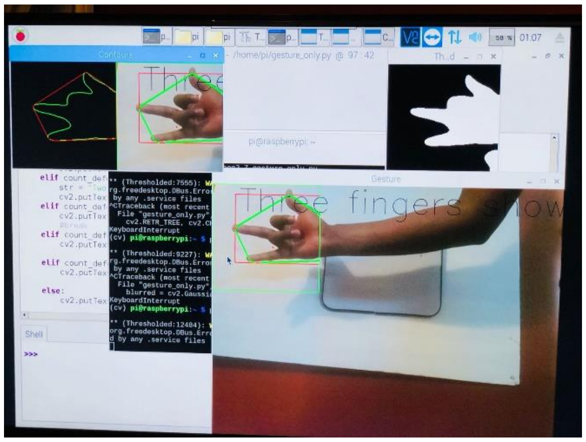
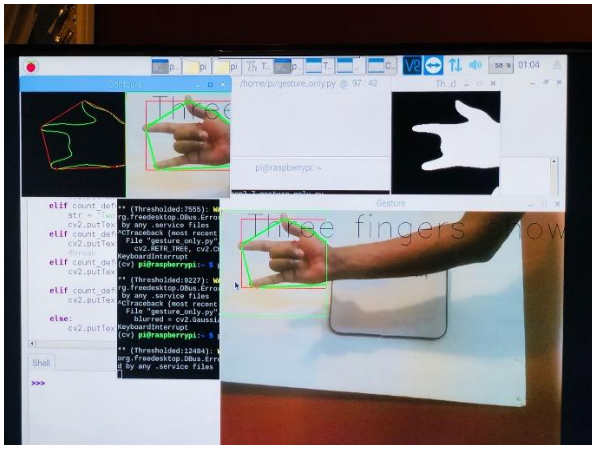
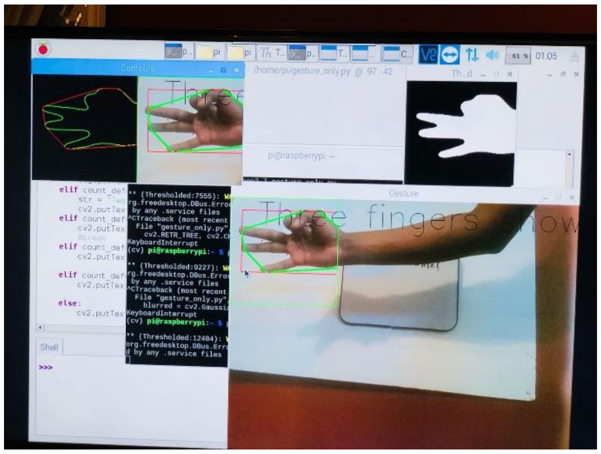


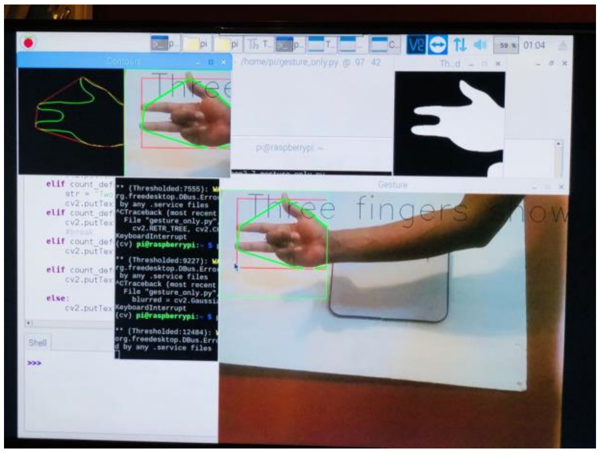


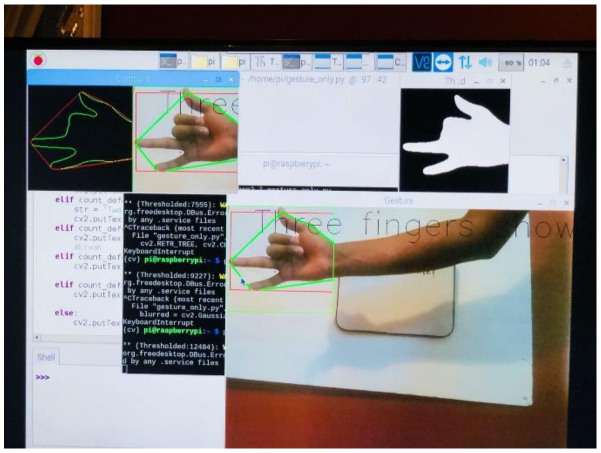
6.2.4 Three fingers shown when convexity defect is 2:






Three fingers show

Three fingers show

Three fingers show

Three fingers show

Three fingers show

Three fingers show





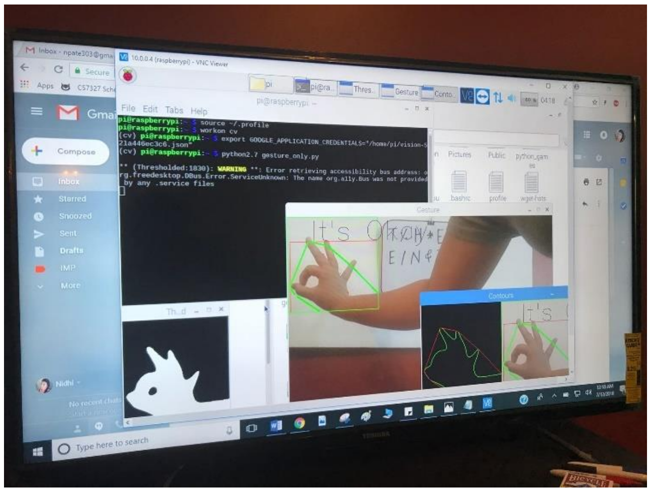
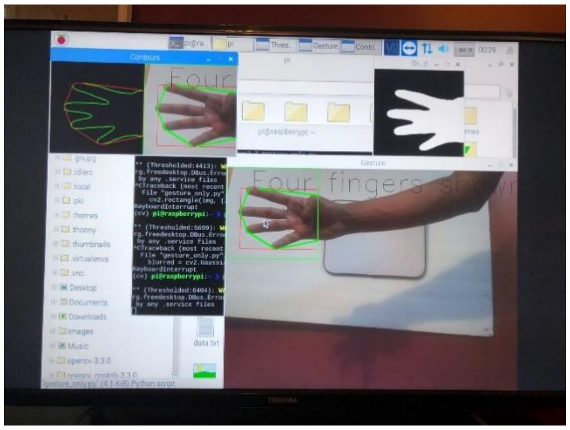
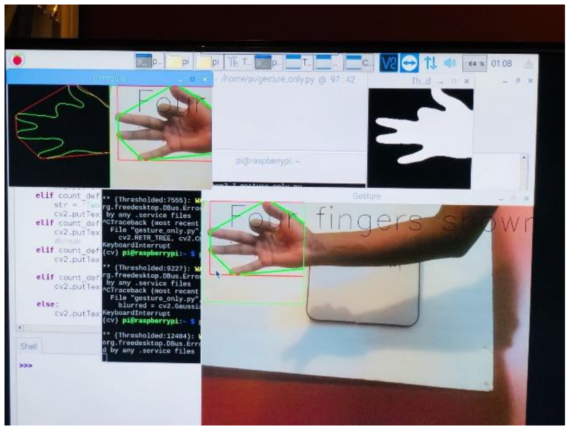


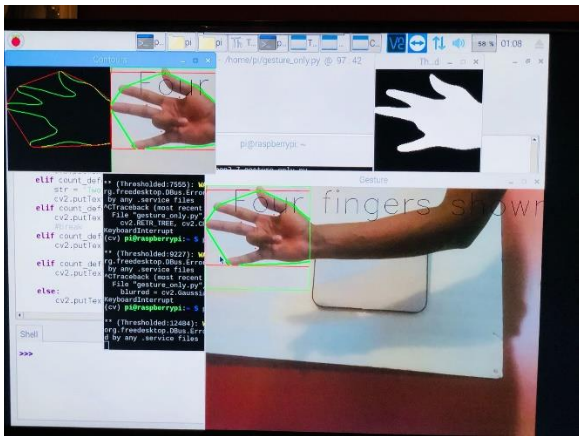
Fig 5.9 "OK" gesture

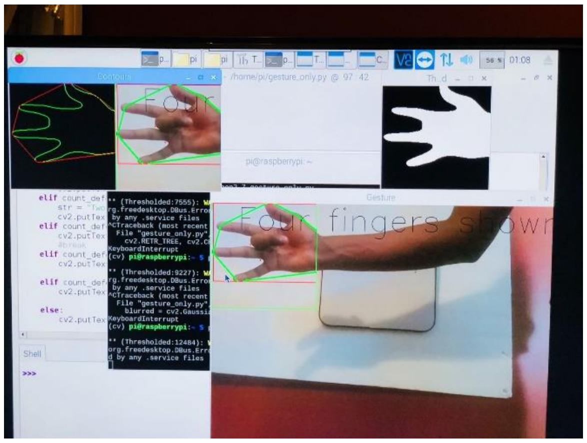
Figure 5.9 shows that to show "OK" gesture we have to use three fingers but it is clearly distinguish between "OK" gestures and three finger gestures.

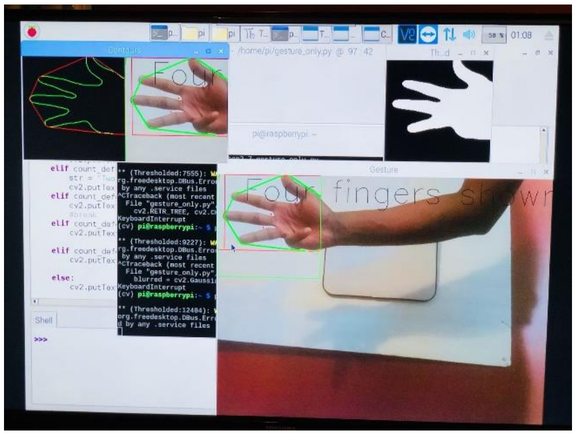
6.2.5 Four fingers shown when our convexity defect is 3:



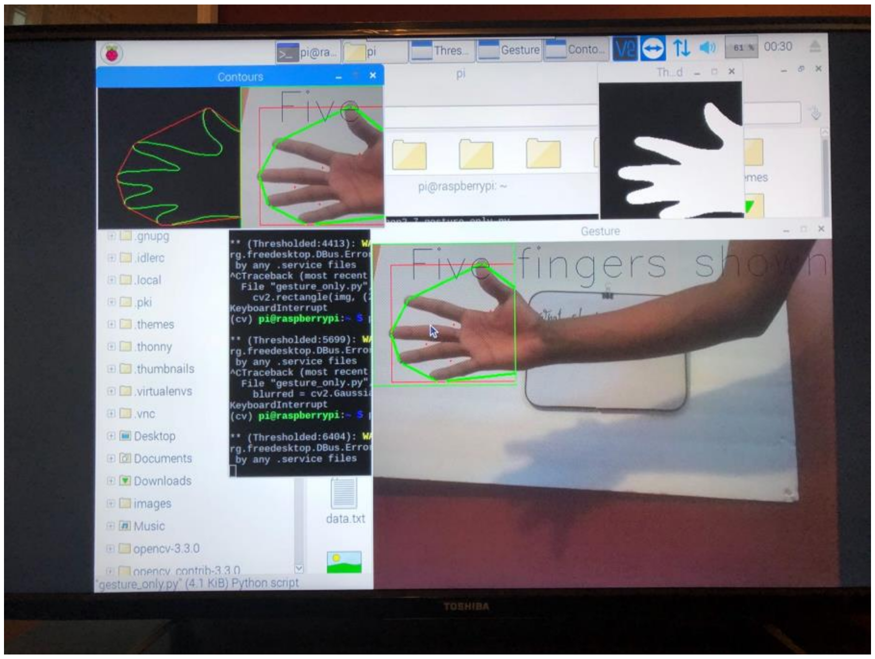








6.2.6 Five fingers shown when convexity defect is 4:

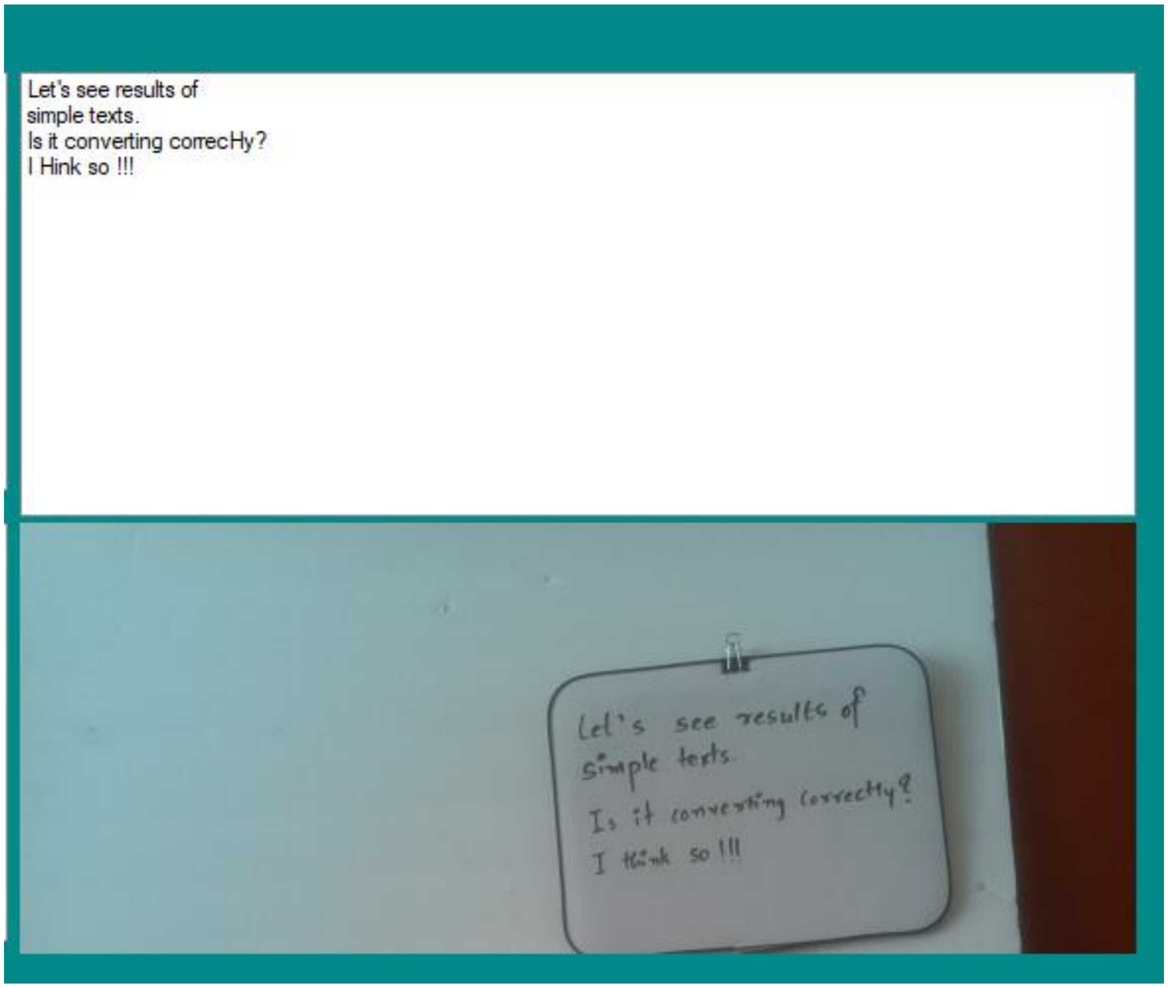


```
** (Thresholded:4413): WA
rg.freedesktop.DBus.Erro
 by any .service files
^CTraceback (most recent
  File "gesture_only.py"
    cv2.rectangle(img, (
KeyboardInterrupt
(cv) pi@raspberrypi:~ $

** (Thresholded:5699): WA
rg.freedesktop.DBus.Erro
 by any .service files
^CTraceback (most recent
  File "gesture_only.py"
    blurred = cv2.Gaussia
KeyboardInterrupt
(cv) pi@raspberrypi:~ $

** (Thresholded:6404): WA
rg.freedesktop.DBus.Erro
 by any .service files
```

6.3 Image to text Conversion

6.3.1 Text images.

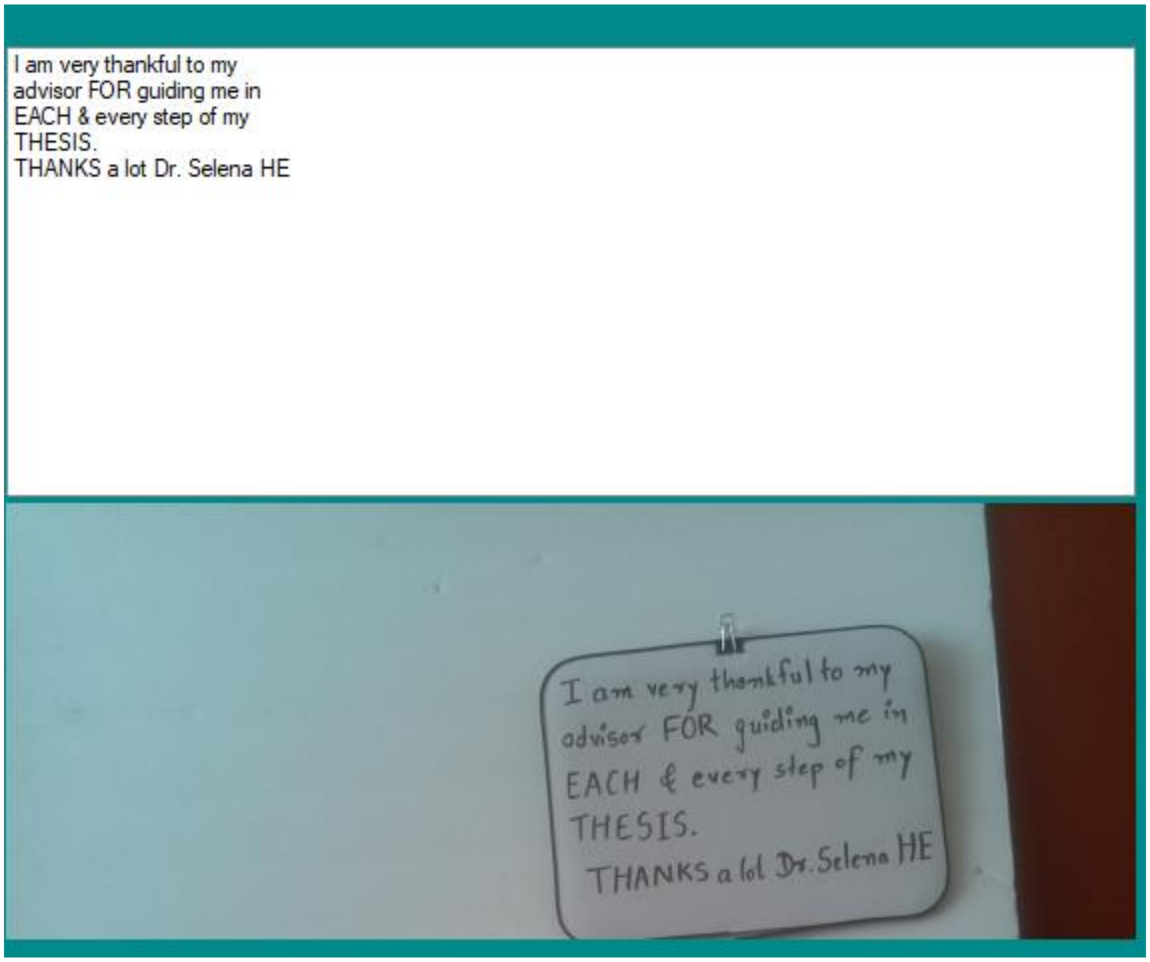
In this section all the results of image to text conversion for alphabets , numbers and special characters has been taken for taste case.



```
Let's see results of
simple texts.
Is it converting correcHy?
I Hink so !!!
```

Fig. 5.10 : Simple Text Conversion

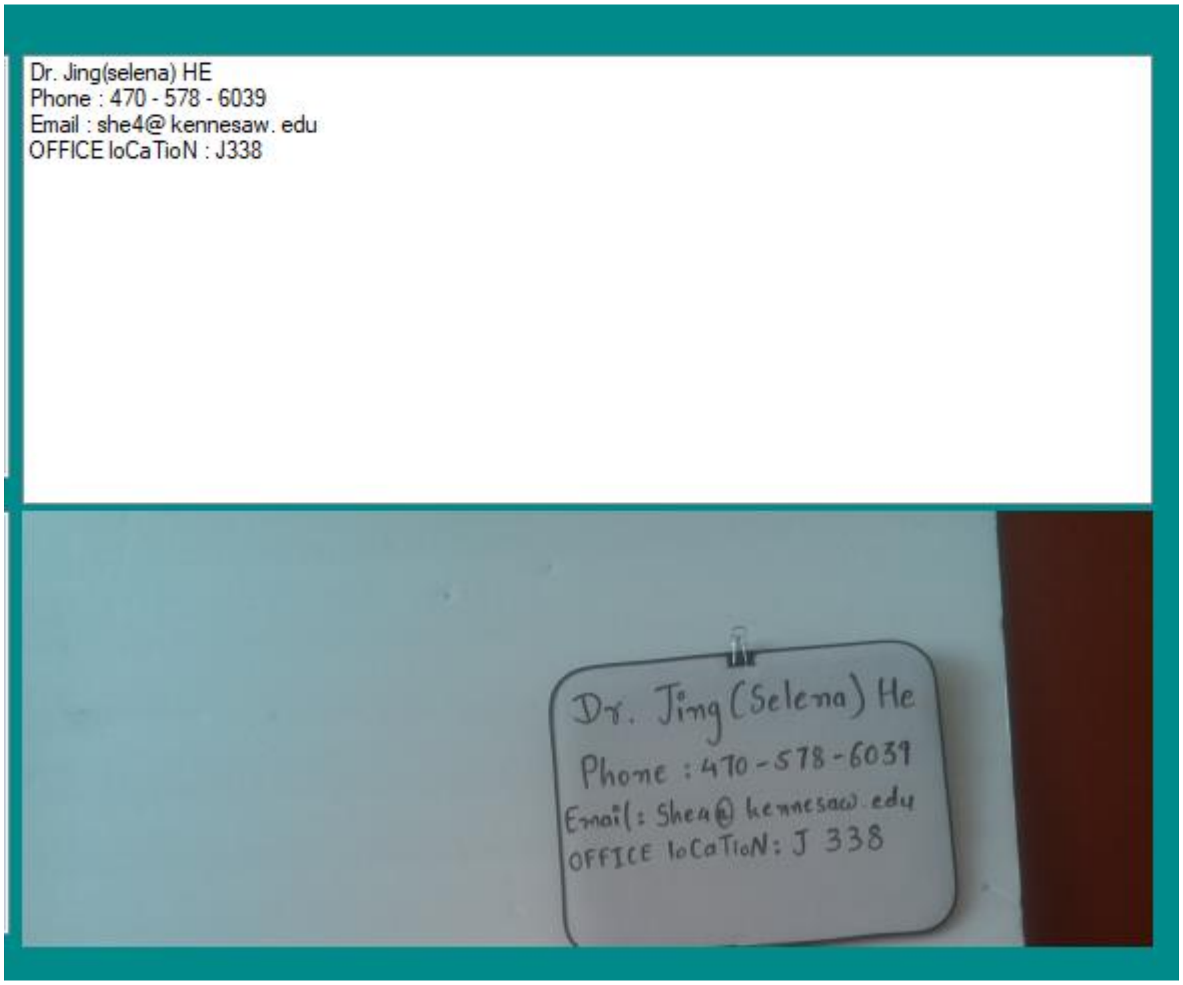
Figure 5.10 show the image to text conversion. In the captured image of white board, writer has attached two letters "t" "l" and "t" "h" so system is taking it as "H". Hence, for words "correctly" and "think" the output generated by system is " correcHy" and "Hink". System takes the nearby analyzed letter in case of confusion. So, in this way it makes few mistakes if there isn't any clarity.



I am very thankful to my
advisor FOR guiding me in
EACH & every step of my
THESIS.
THANKS a lot Dr. Selena HE

Fig. 5.11 Simple Text Conversion

Figure 5.11 show the image to text conversion. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
Dr. Jing(selena) HE
Phone : 470 - 578 - 6039
Email : she4@ kennesaw. edu
OFFICE loCaTioN : J338
```

Fig. 5.12 Simple Text Conversion

Figure 5.12 show the image to text conversion. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.

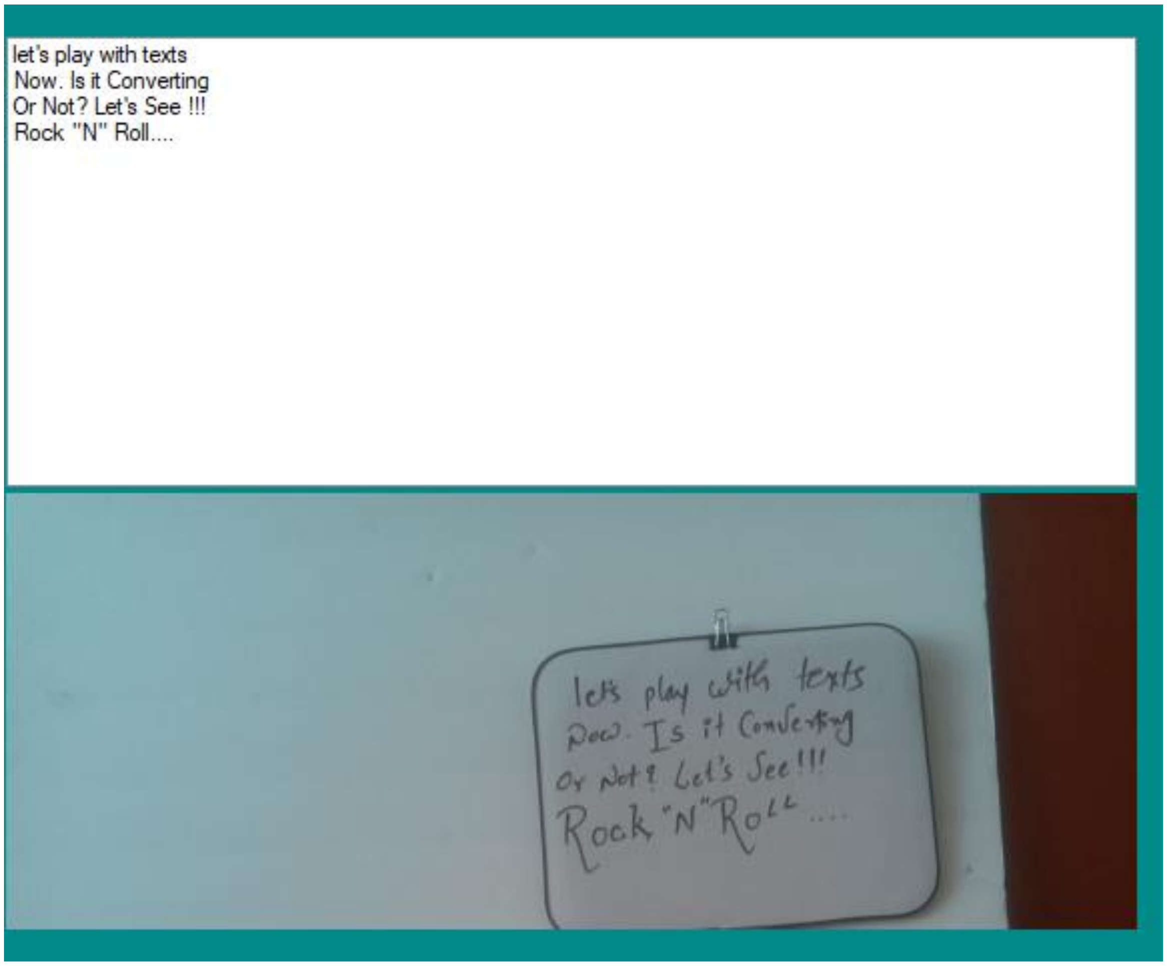
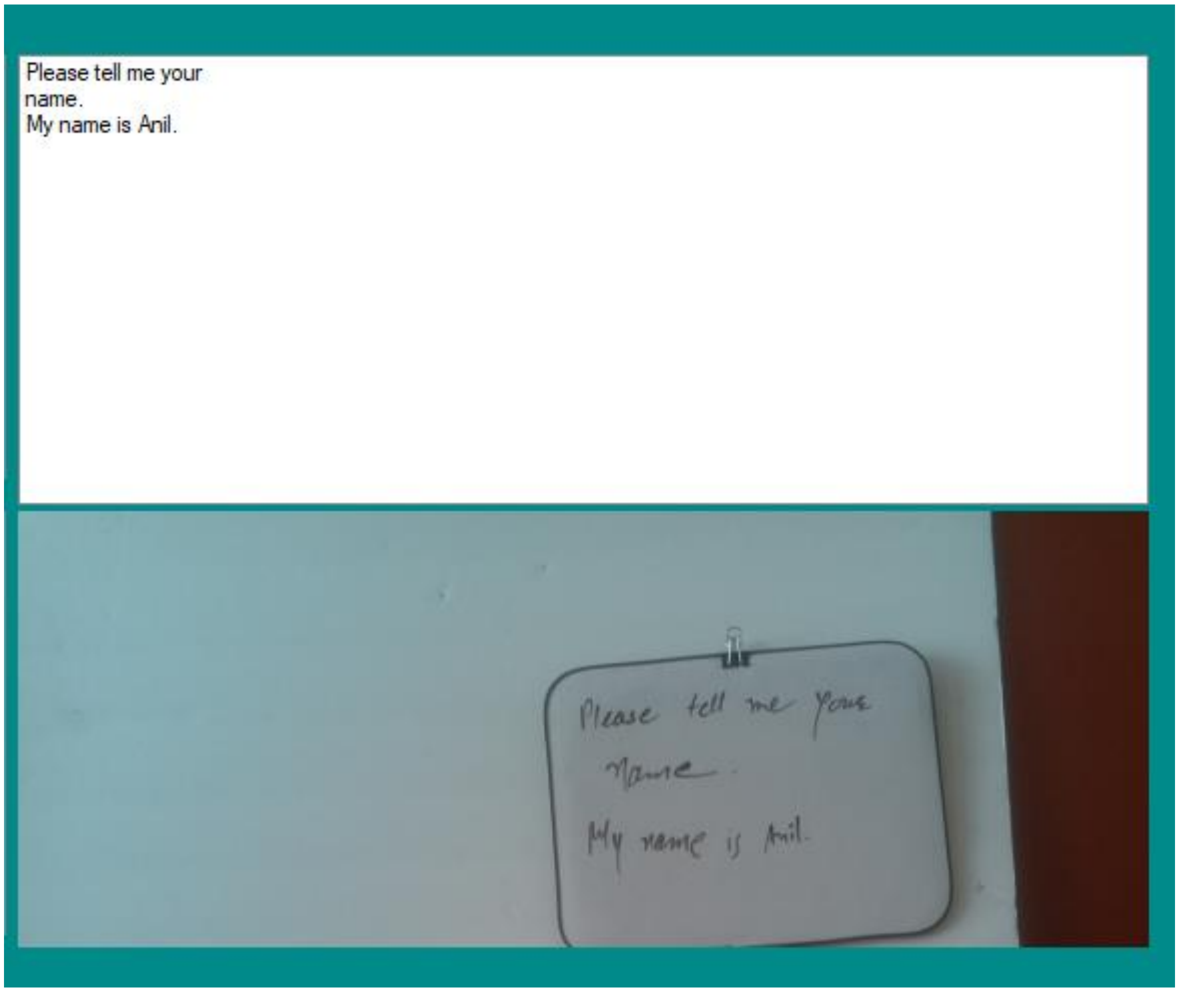

```
let's play with texts
Now. Is it Converting
Or Not? Let's See !!!
Rock "N" Roll....
```

Fig. 5.13 Simple Text Conversion

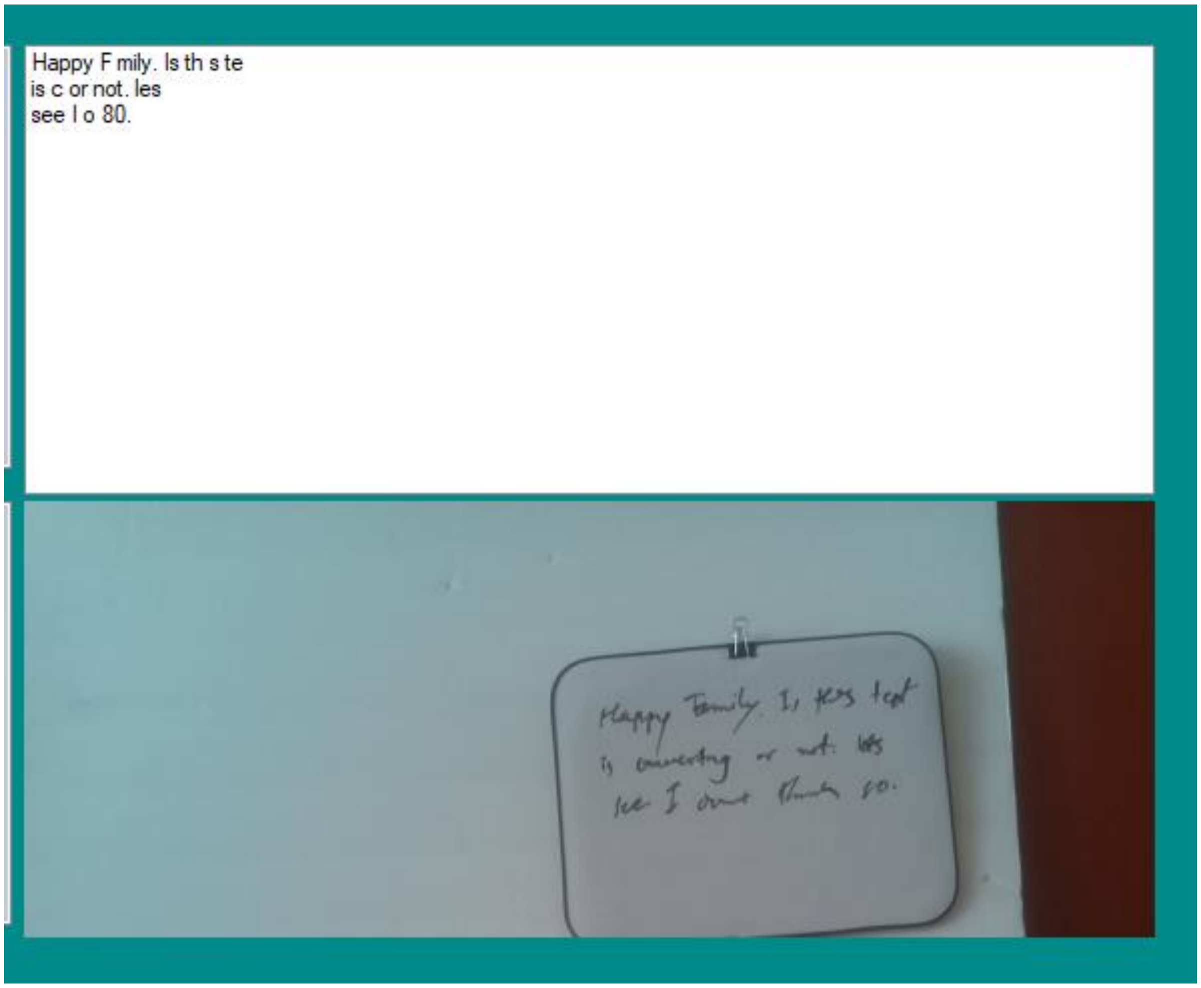
Figure 5.13 show the image to text conversion. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



Please tell me your
name.
My name is Anil.

Fig. 5.14 Complex Text Conversion

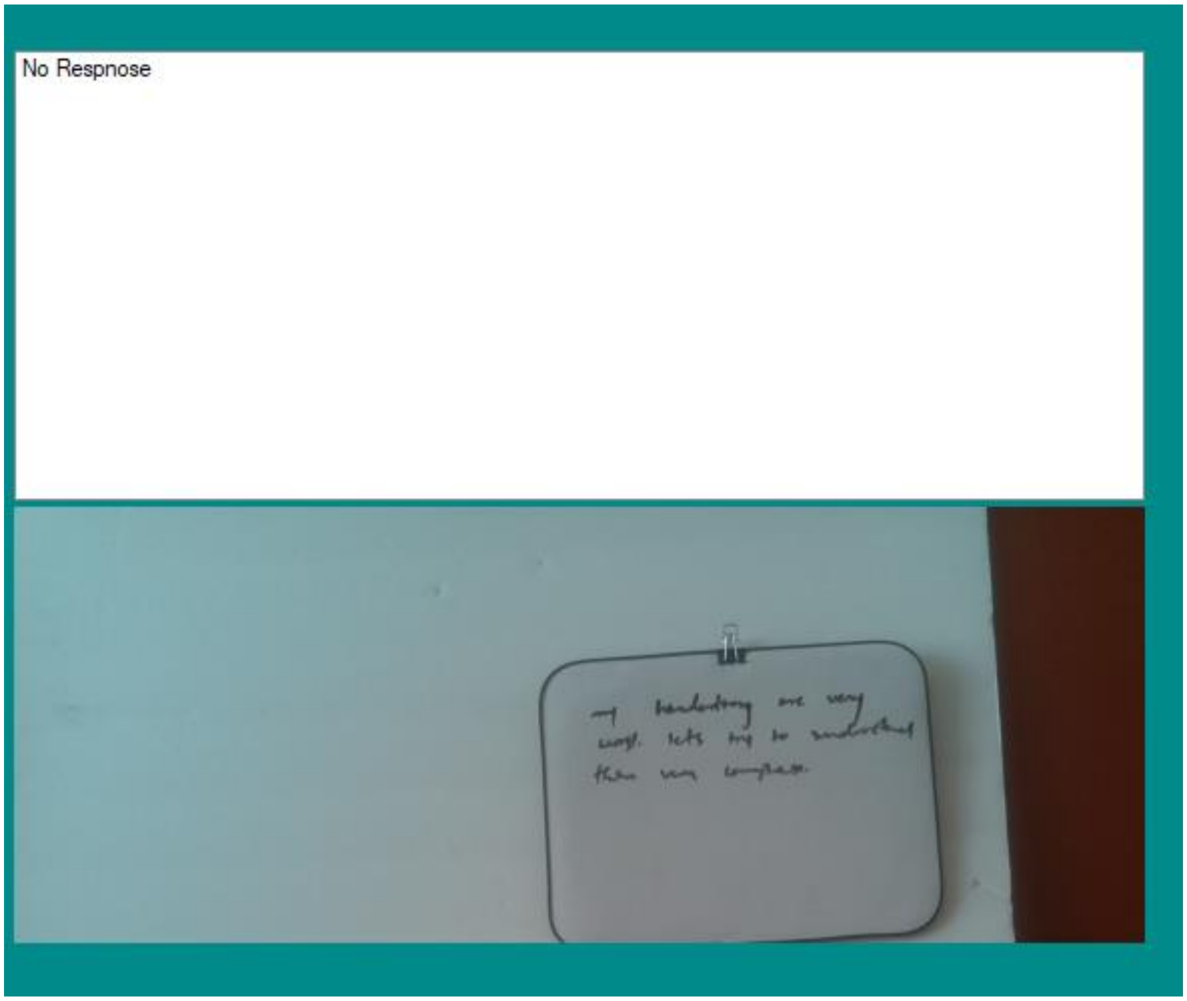
Figure 5.14 show the image to text conversion. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
Happy F mily. Is th s te
is c or not. les
see I o 80.
```

Fig. 5.15 Complex Text Conversion

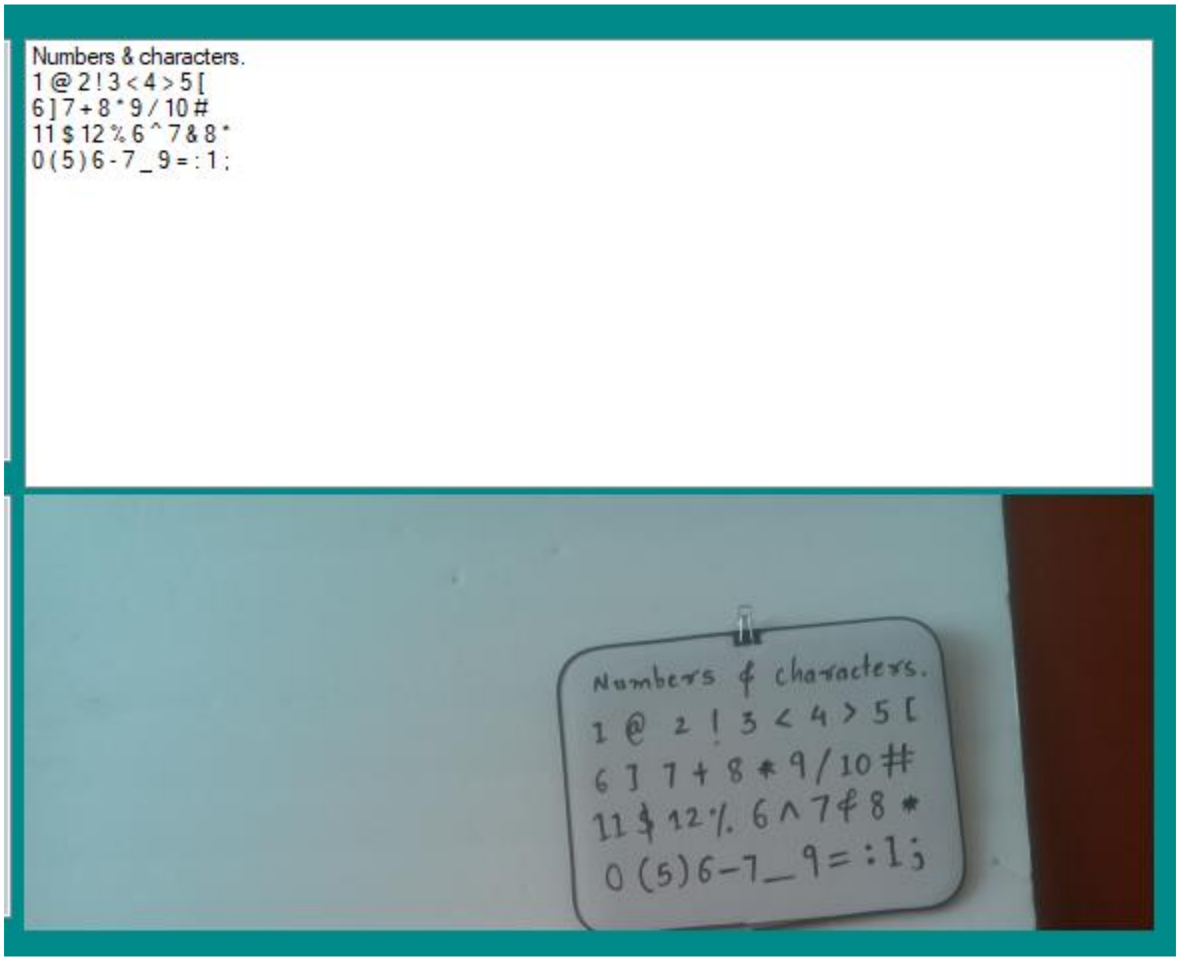
Figure 5.15 show the image to text conversion. In the captured image of white board, writer has not written in a proper format. Hence the output has many letters missing and many errors. System takes the nearby analyzed letter in case of confusion. So, in this way it makes few mistakes if there isn’t any clarity.



```
No Respnose
```

Fig. 5.16 Complex Text Conversion

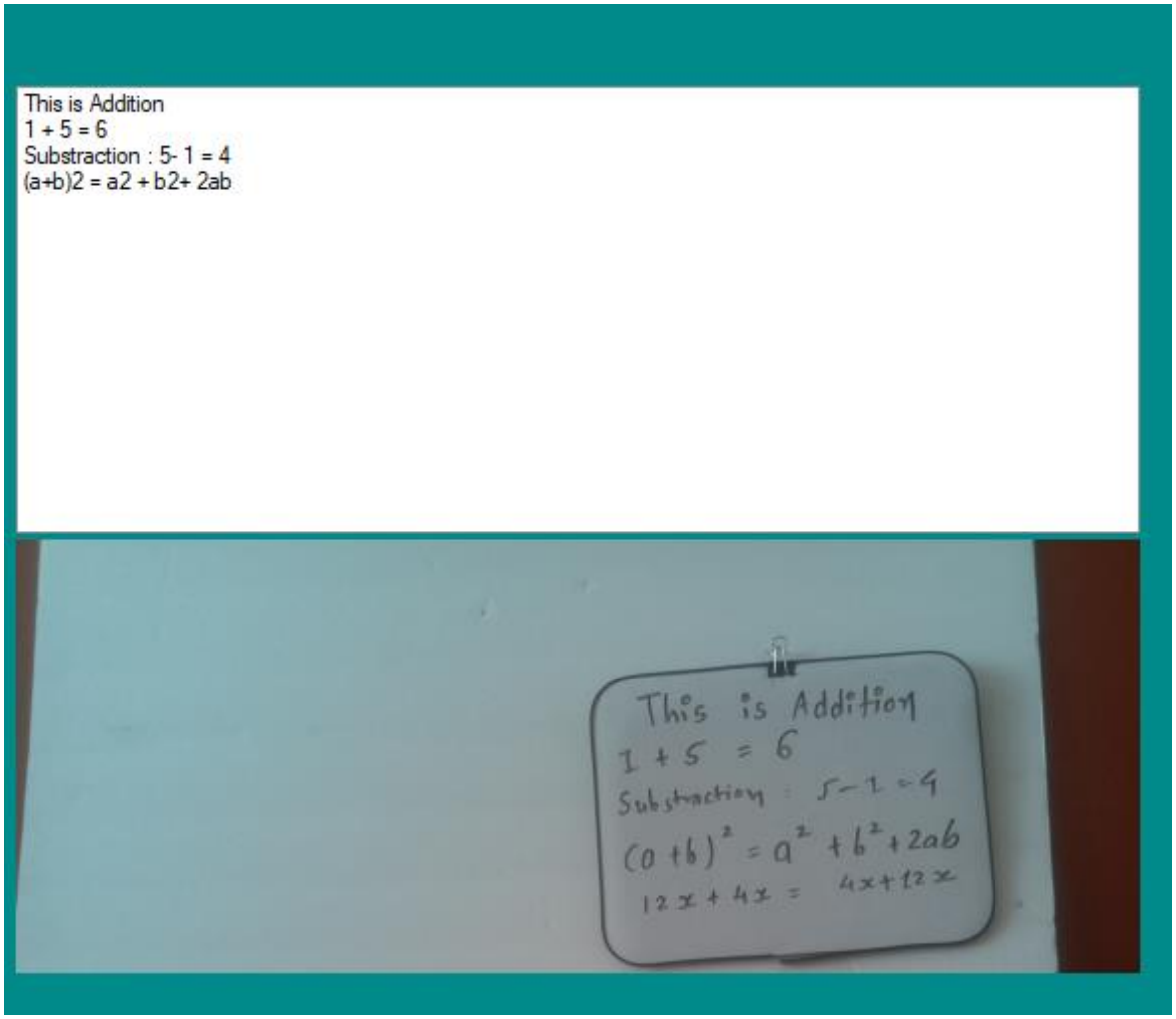
Figure 5.16 show the image to text conversion. In the captured image of white board, system couldn’t analyze none of the words. System simply gave “No Response” output. So the input from the whiteboard needs to clear to generate some output.



```
Numbers & characters.
1 @ 2 ! 3 < 4 > 5 [
6 ] 7 + 8 * 9 / 10 #
11 $ 12 % 6 ^ 7 & 8 *
0 ( 5 ) 6 - 7 _ 9 = : 1 ;
```

Fig. 5.17 Simple Text Conversion

Figure 517 show the image to text conversion. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
This is Addition
1 + 5 = 6
Substraction : 5- 1 = 4
(a+b)2 = a2 + b2+ 2ab
```

Fig 5.18 Simple Text

Figure 5.18 shows the image to text conversion. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.

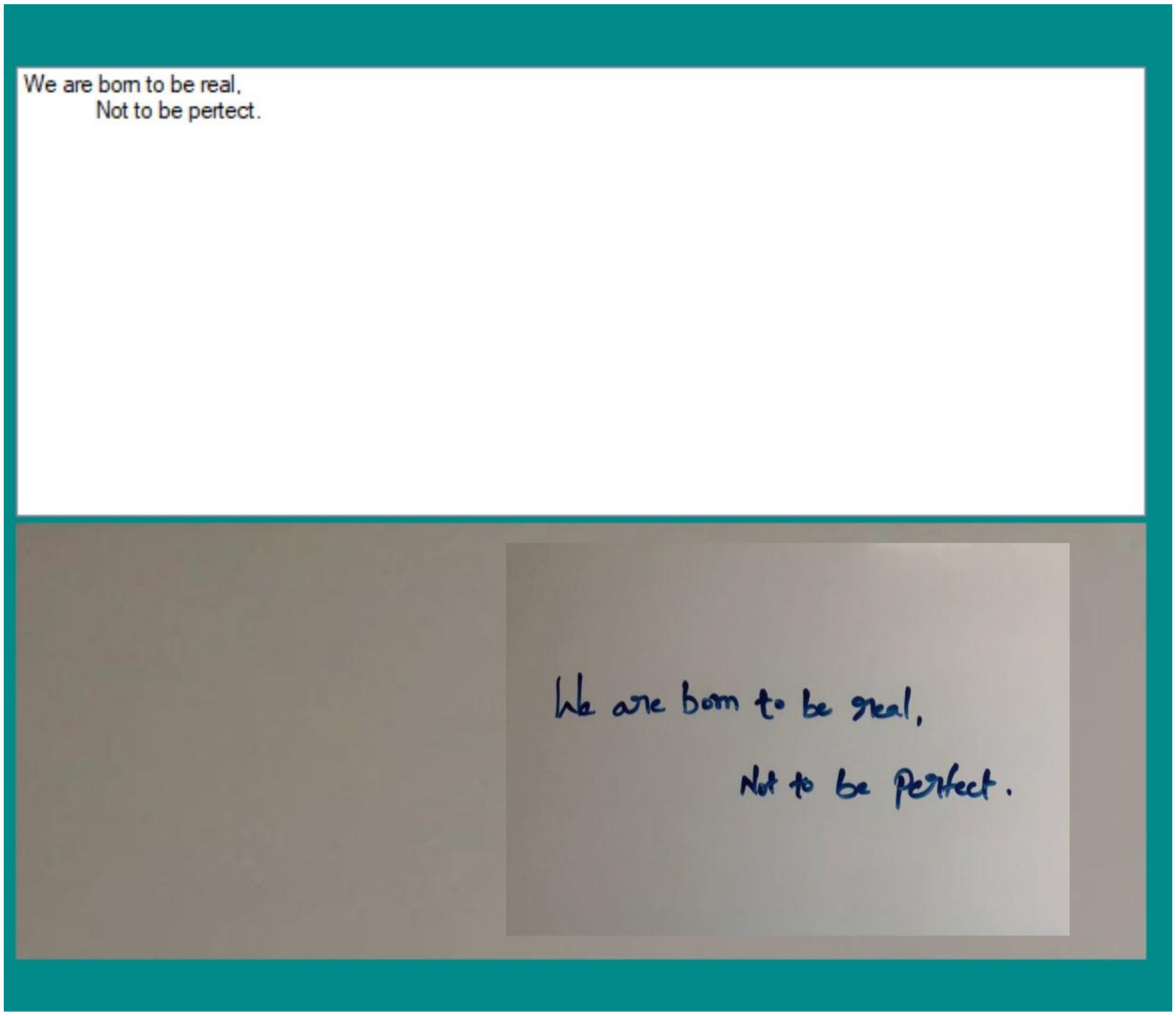
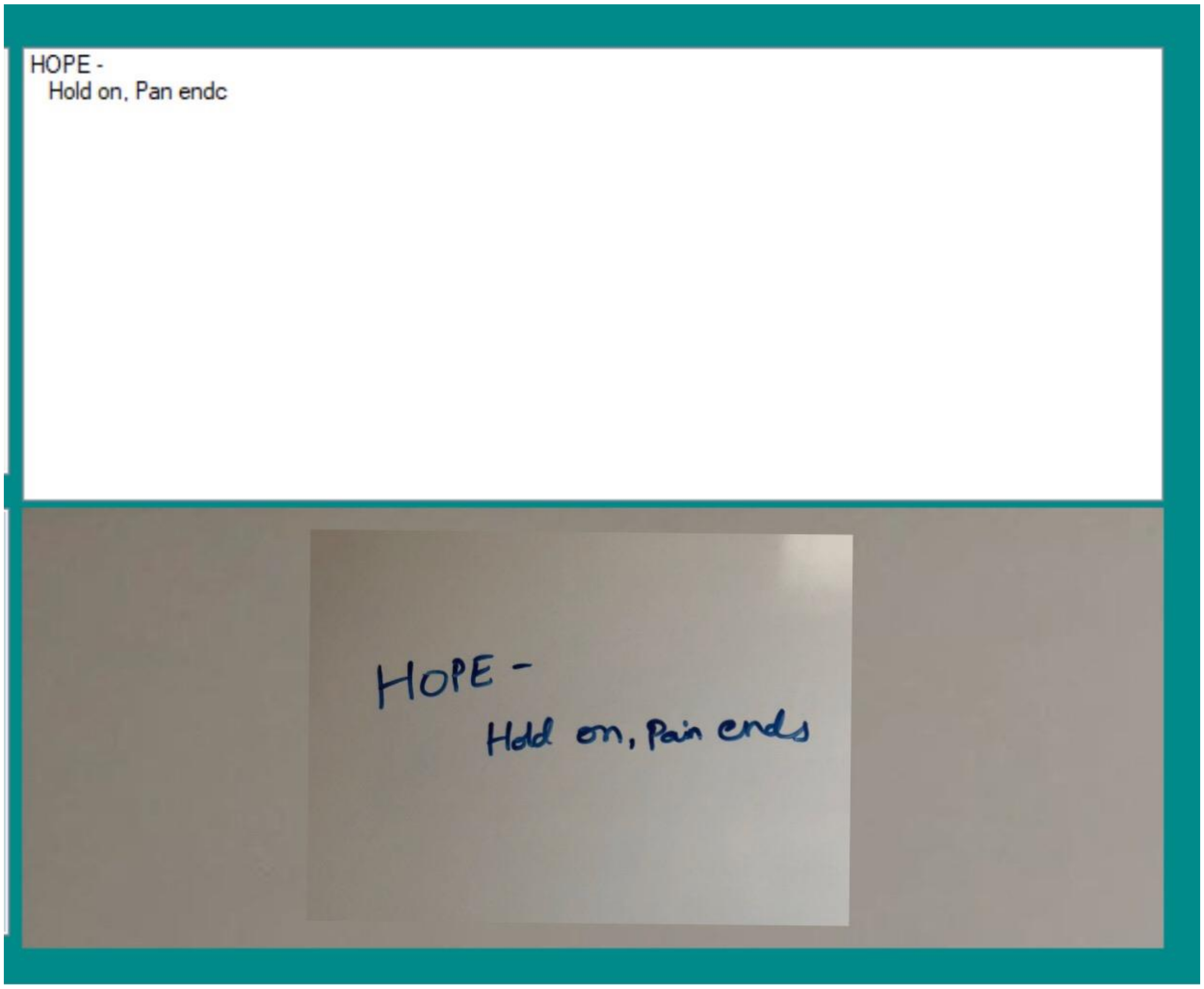

```
We are bom to be real,
         Not to be pertect.
```

Fig 5.19 Complex Text Conversion

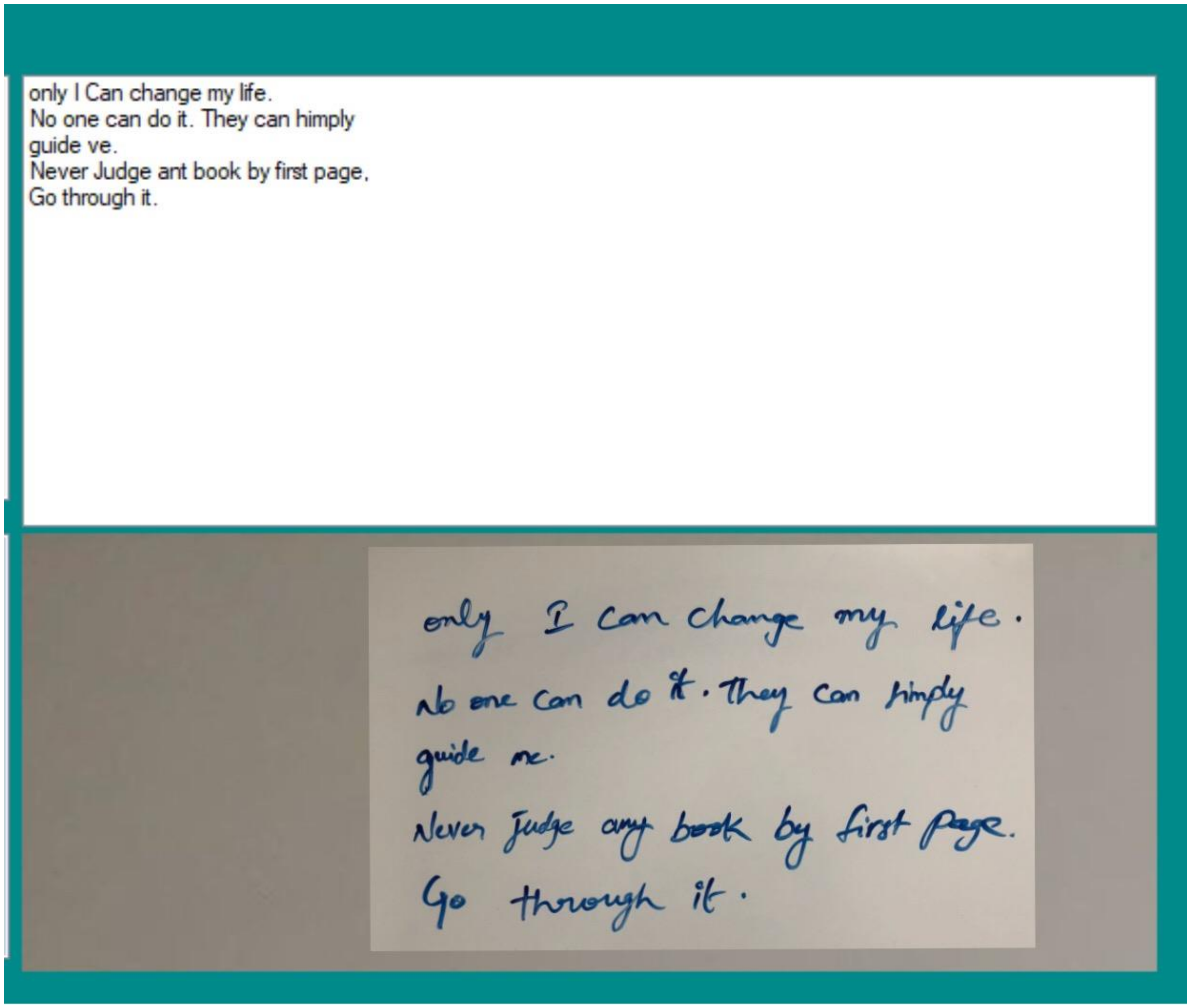
Figure 5.19 shows the image to text conversion. Captured image is correctly getting converted as the output except word “ perfect” other than that system can correctly read all the letters and gave the correct report.



```
HOPE -
  Hold on, Pan endc
```

Fig 5.20 Complex Text Conversion

Figure 5.20 show the image to text conversion. Captured image is correctly getting converted as the output except word “ ends” other than that system can correctly read all the letters and gave the correct report.

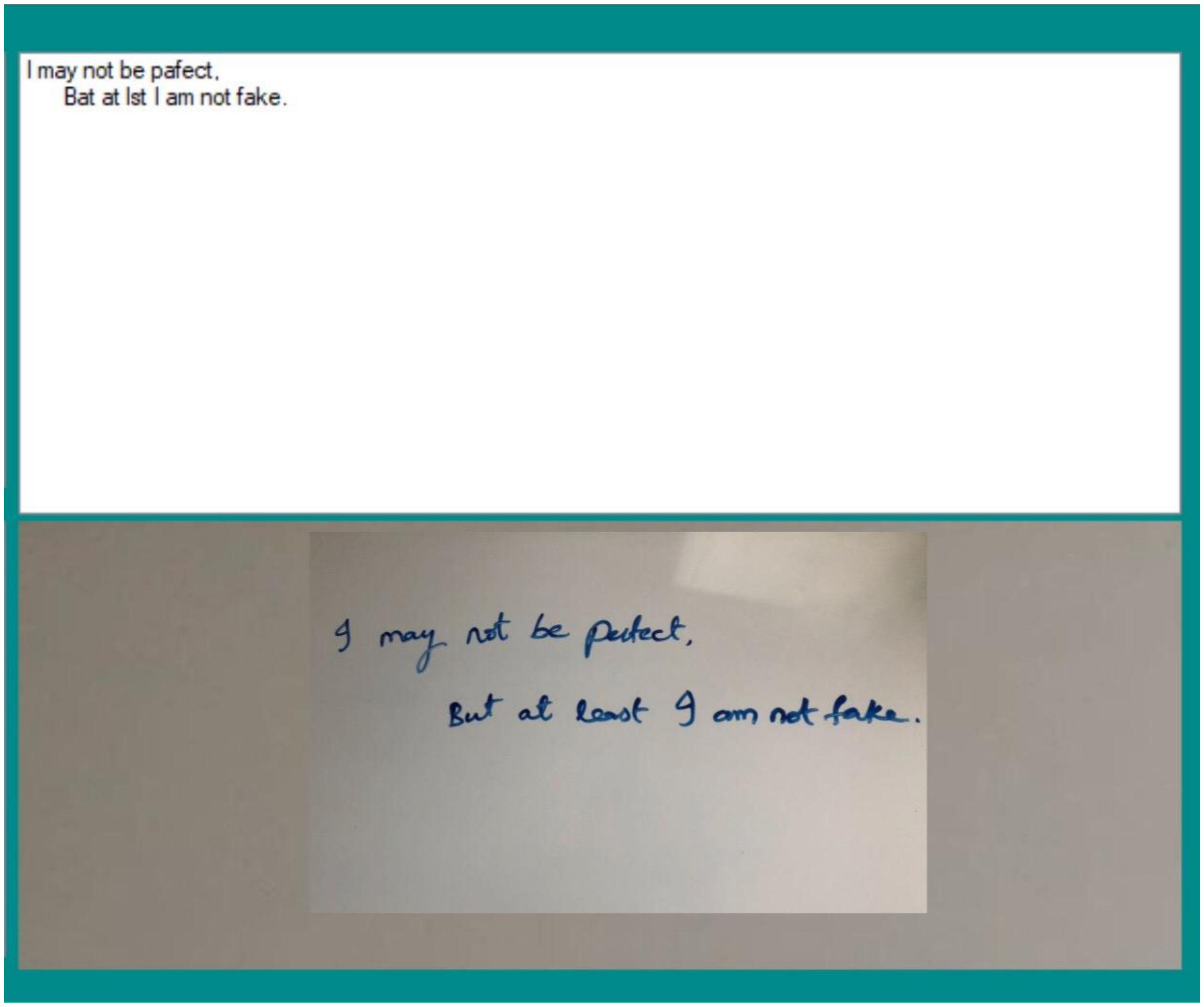


```
only I Can change my life.
No one can do it. They can himply
guide ve.
Never Judge ant book by first page,
Go through it.
```

only I can change my life.
No one can do it. They can simply
guide me.
Never Judge any book by first page.
Go through it.

Fig 5.21 Complex Text Conversion

Figure 5.21 show the image to text conversion. Captured image is correctly getting converted as the output except words “ simply” , “me” and “and” other than that system can correctly read all the letters and gave the correct report.



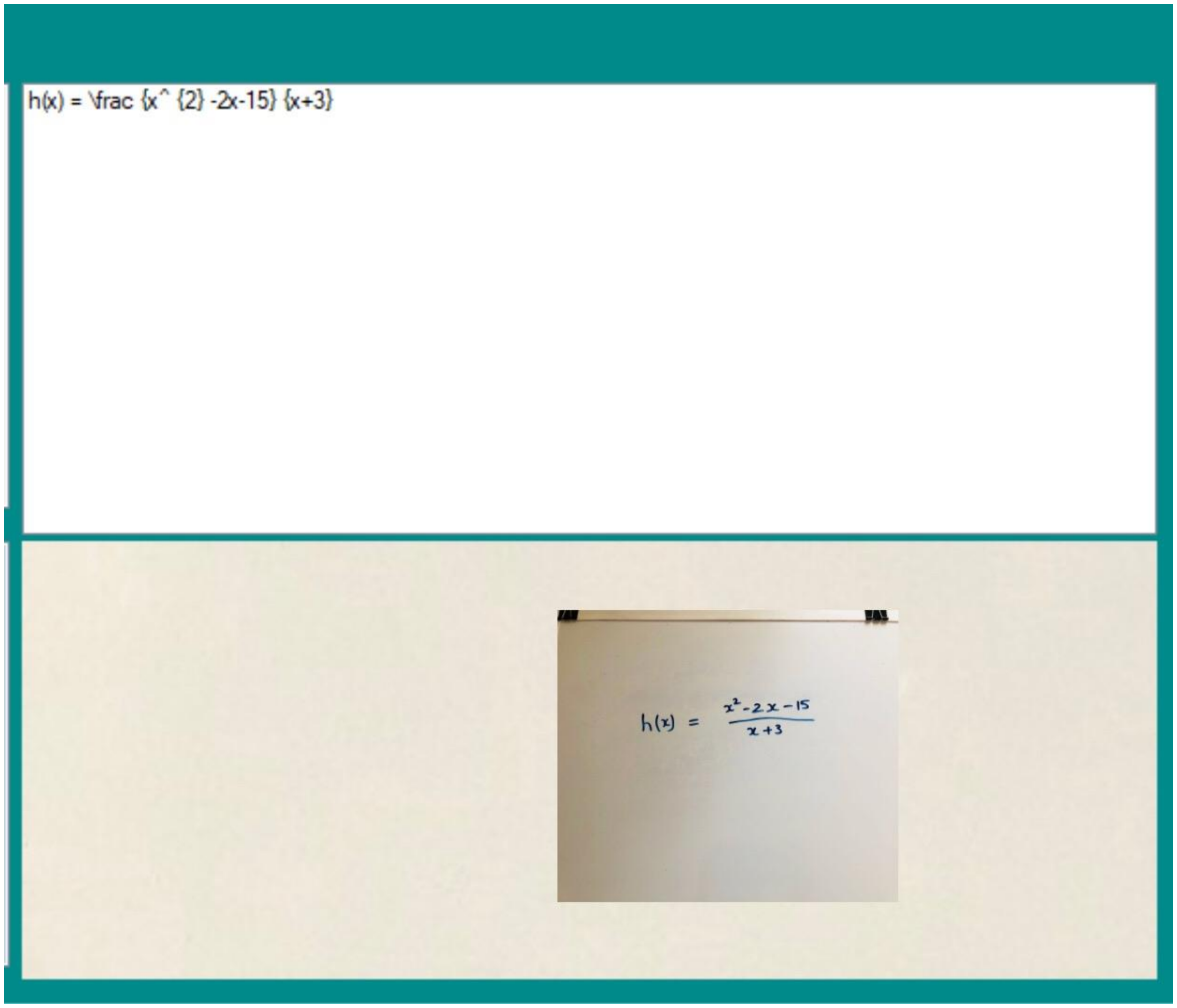
```
I may not be pafect,
      Bat at lst I am not fake.
```

Fig 5.22 Complex Text Conversion

Figure 5.22show the image to text conversion. Captured image is correctly getting converted as the output except words “perfect” and “ but” other than that system can correctly read all the letters and gave the correct report.

6.3.2 Mathematical Equation Images to Text Conversion

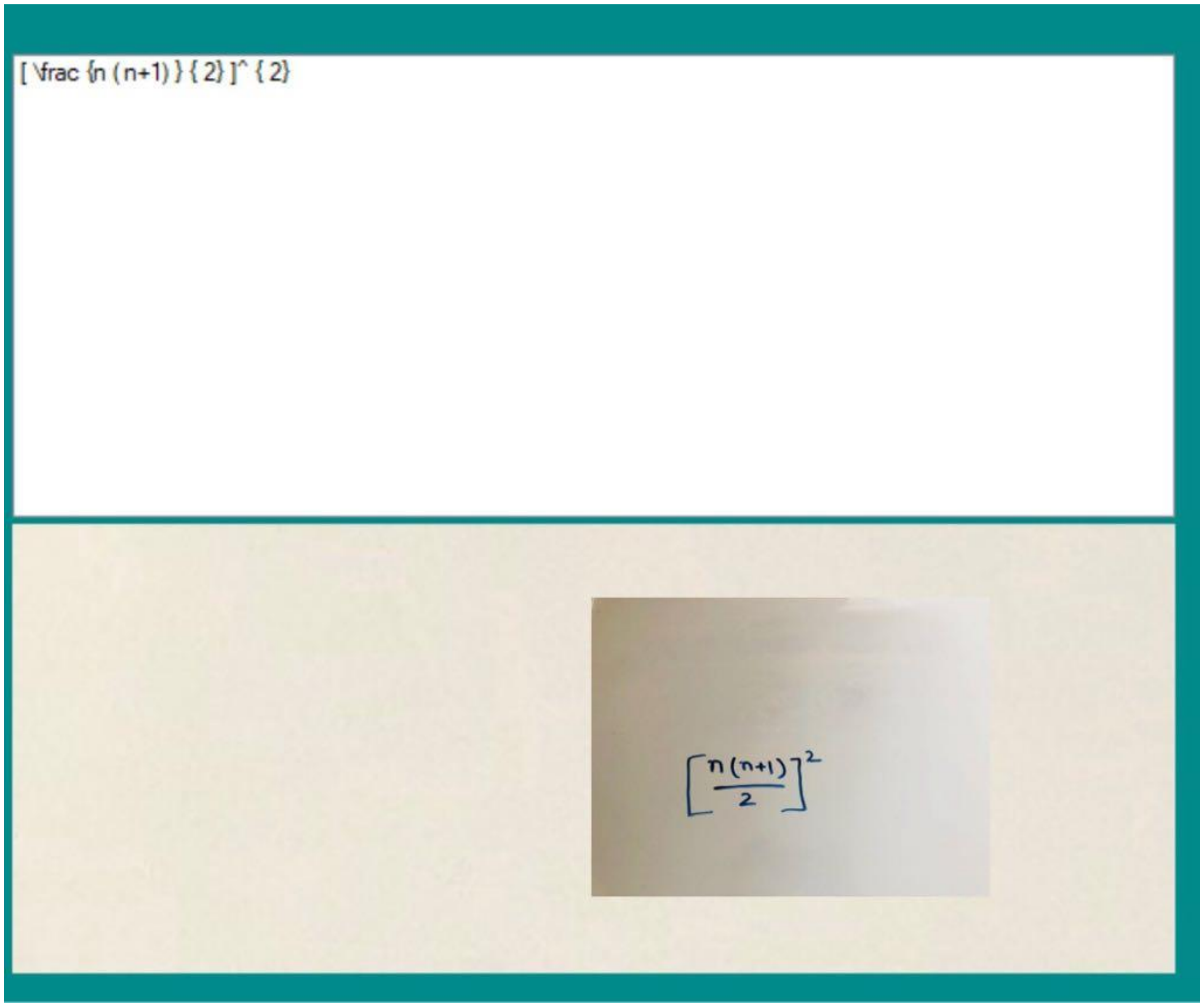
This section gives result for image to text conversion for Mathematical Equations .



```
h(x) = \frac {x^ {2} -2x-15} {x+3}
```

Fig 5.23 Mathematical Equation Image to Text Conversion.

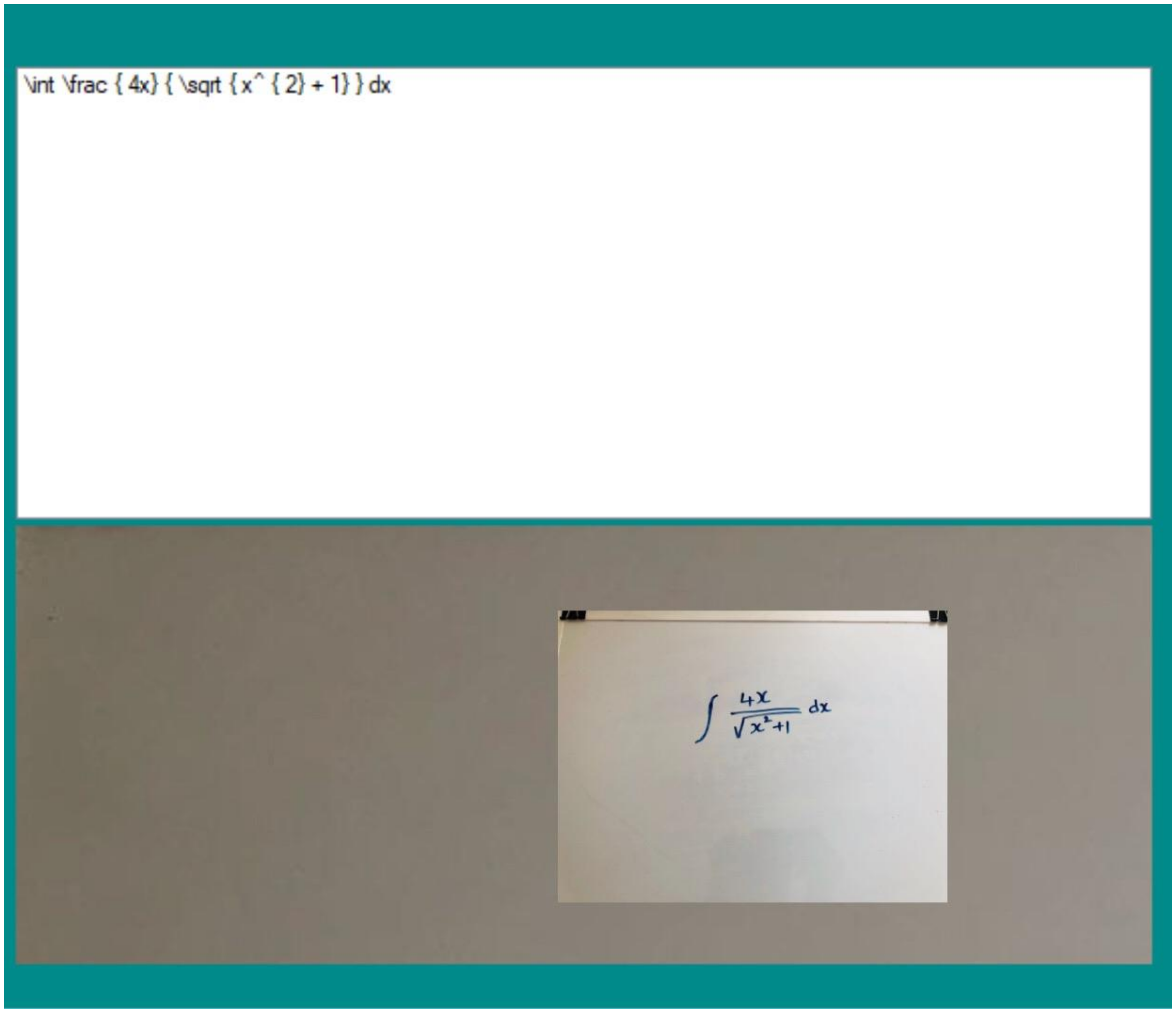
Figure 5.23 shows the image to text conversion for mathematical equations. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
[ \frac {n ( n+1) } { 2} ]^ { 2}
```

Fig 5.24 Mathematical Equation Image to Text Conversion.

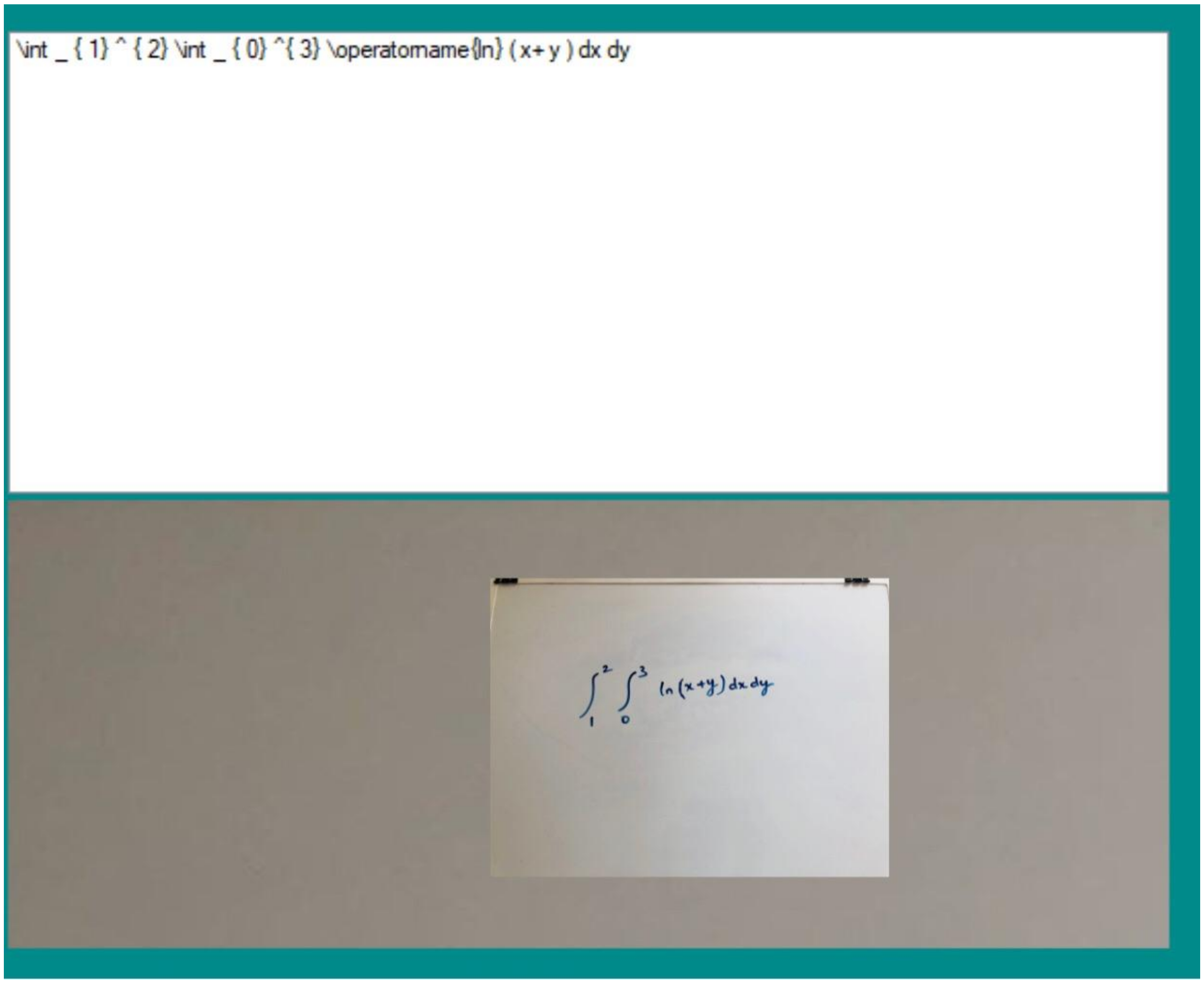
Figure 5.24 shows the image to text conversion for mathematical equations. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
\int \frac { 4x} { \sqrt { x^ { 2} + 1} } dx
```

Fig 5.25 Mathematical Equation Image to Text Conversion.

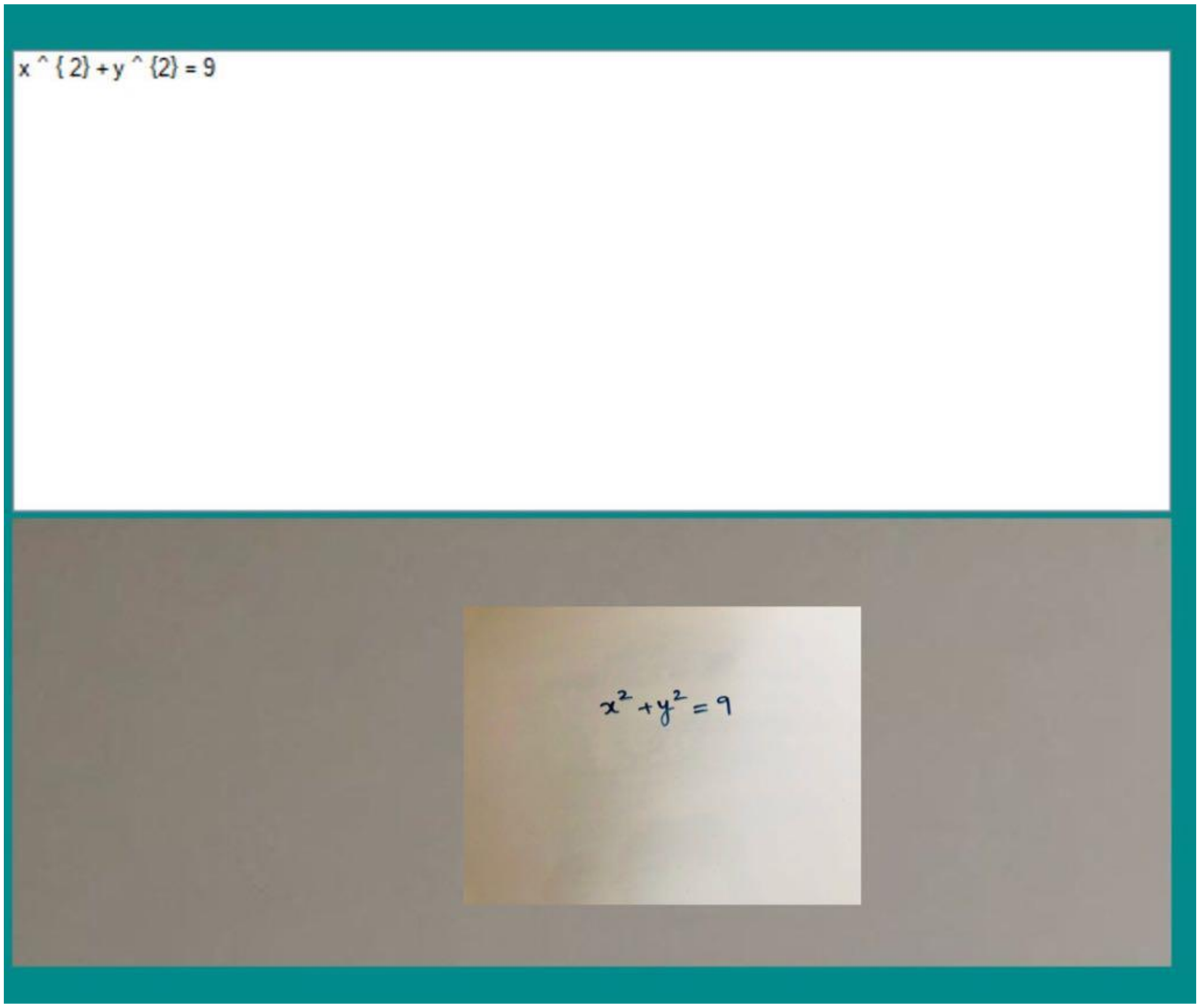
Figure 5.25 shows the image to text conversion for mathematical equations. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
\int _ { 1} ^ { 2} \int _ { 0} ^{ 3} \operatorname{ln} ( x+ y ) dx dy
```

Fig 6.26 Mathematical Equation Image to Text Conversion.

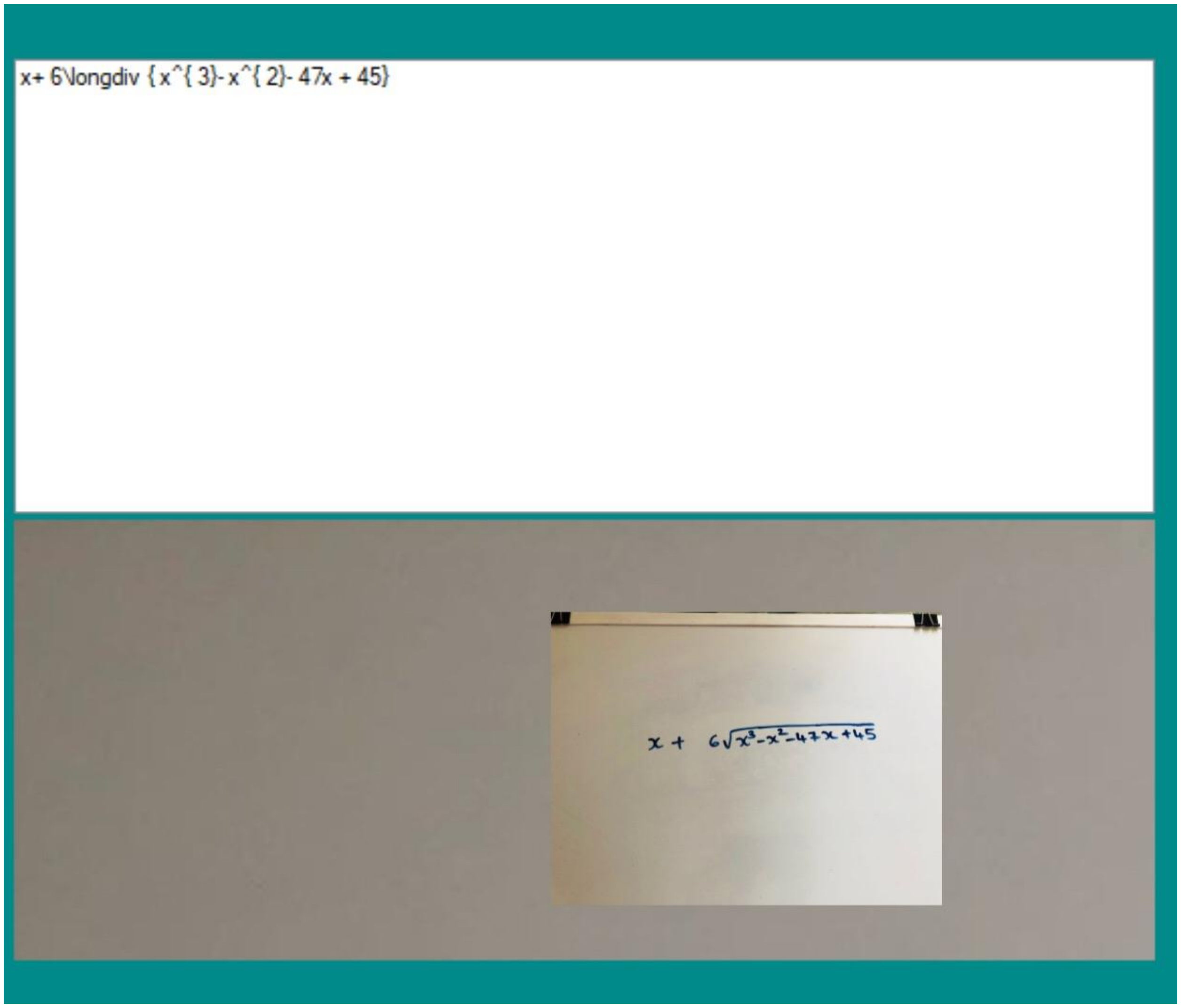
Figure 6.26 shows the image to text conversion for mathematical equations. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



x ^ { 2} + y ^ {2} = 9

Fig 6.27 Mathematical Equation Image to Text Conversion.

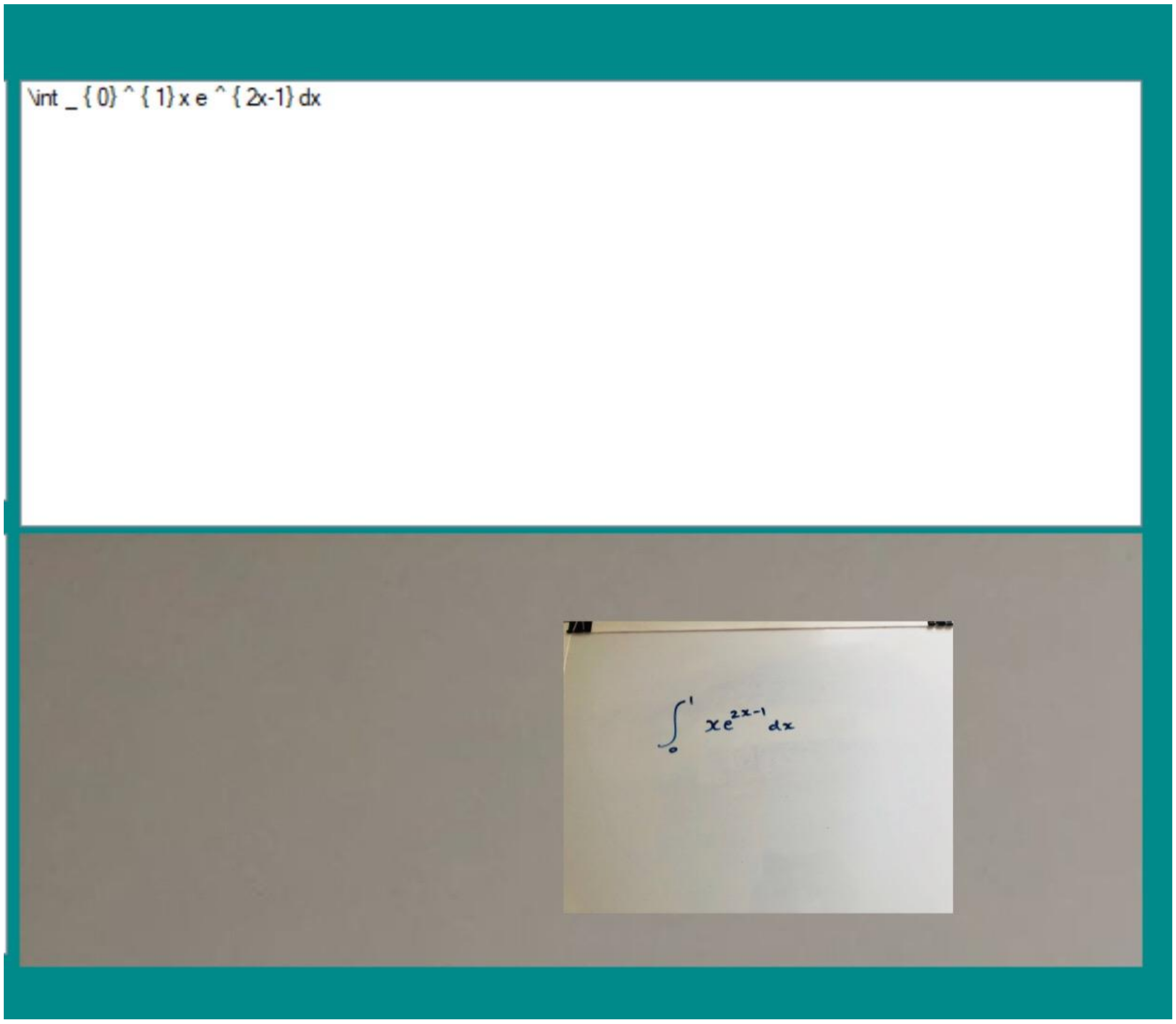
Figure 6.27 shows the image to text conversion for mathematical equations. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
x+ 6\longdiv {x^{ 3}- x^{ 2}- 47x + 45}
```

Fig 6.28 Mathematical Equation Image to Text Conversion.

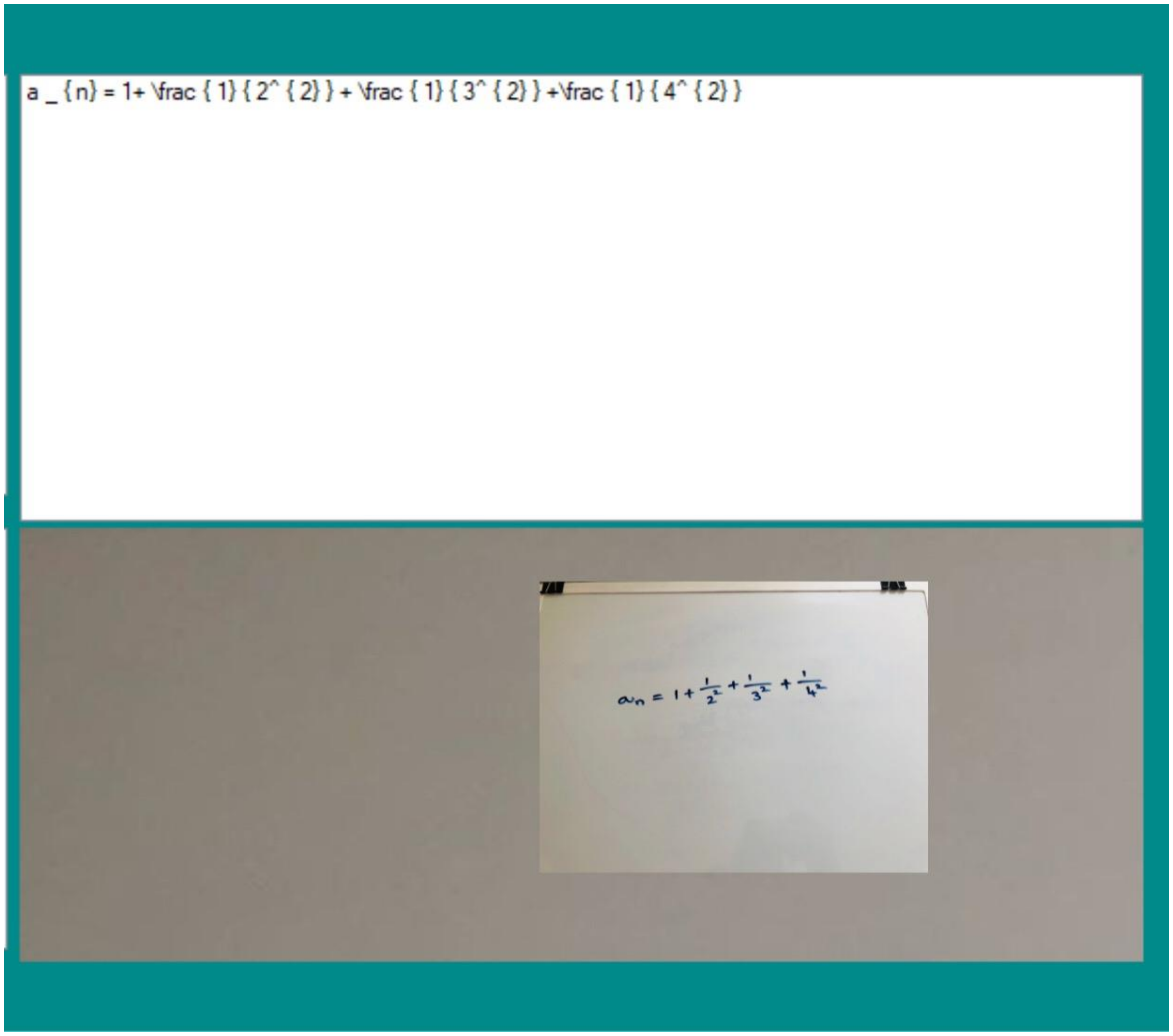
Figure 6.28 shows the image to text conversion for mathematical equations. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
\int _ { 0} ^ { 1} x e ^ { 2x-1} dx
```

Fig 6.29 Mathematical Equation Image to Text Conversion.

Figure 6.29 shows the image to text conversion for mathematical equations. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.



```
a _ {n} = 1+ \frac { 1} { 2^ { 2} } + \frac { 1} { 3^ { 2} } +\frac { 1} { 4^ { 2} }
```

Fig 6.30 Mathematical Equation Image to Text Conversion.

Figure 6.30 shows the image to text conversion for mathematical equations. Captured image is correctly getting converted as the output. System can correctly read all the letters and gave the correct report.

Chapter 6

EVALUATION

The Evolution of entire system is performed using time factor (how much time does system take to finish the task) , cost factor (how much cost is needed to build a system) and accuracy of entire system

6.1: Evaluation of system by considering TIME factor

We have calculated time taken by system to convert image into text after the image is captured by Raspberry Pi. Also, We have calculated time taken by entire system staring form gesture recognition to sending data file to student’s application . We have taken 100 images for simple text , 100 images for complex text and 50 images for mathematical equation. We have calculated average time for text conversion by using following formula.

Time taken by system to convert one image into text = $t_{1.}$

Hence for 100 images time will be t_{100}

So, Average time T = $t1+t2+t3+t4\ldots\ldots\ldots t_{100}$ / 100

So, we have calculated time for each image to be converted into text and then we have found the average time which is shown in Table 6.1.

For entire system, we have taken 100 image combination of simple text, complex text and mathematical equation. For each image we have calculated time starting with gesture recognition to providing data files to student’s application and after that we have found average time for 100 images.

	Process	**Average Time**
1	Text conversion	20.61 sec
2	Over all system	1 min 20 sec

Table 6.1 Require time for execution

In results we have clearly mentioned which is simple text , which is complex text. Simple text has clear , readable and neat handwriting. Complex text has attached letters and not clear handwritings.

	Input images	**Number of input images**	**Average Time**
1	Simple Text	100	19.20 sec
2	Complex Text	100	20.68 sec
3	Mathematical Equations	50	21.95 sec
	Total	**250**	**20.61 sec**

Table 6.2 Average time for text conversion

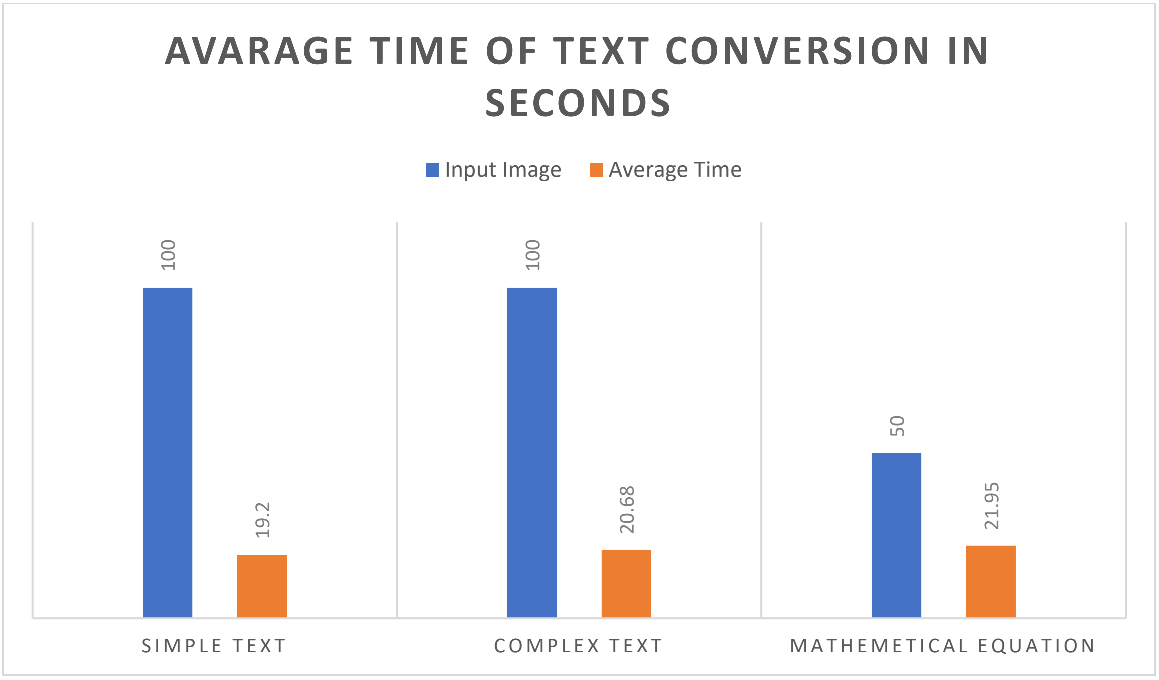


Figure 6.1 (a) Average Time for Text Conversion (with number of inputs)

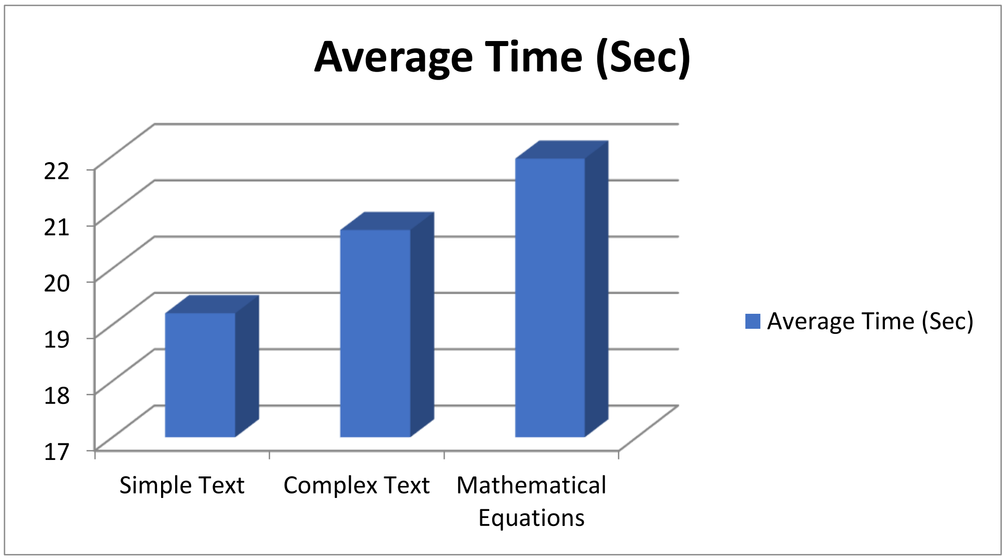


Fig 6.1 (b) Average Time for Text Conversion

Average recognition time is shown here. In figure 6.1 (a) we show the average time with number of inputs. In that our x-axis contain the information about the number which kind of data is there and y-axis gives the information about number of input and the average time for that inputs. Same is mention in figure 6.2. But we only show the information about average time.

Simple text has clear , neat and readable handwritings and complex text has attached letters and not readable handwritings hence, the time taken for simple text is lesser than the time taken by complex text. For mathematical equations the output is provided in latex format hence the time taken by system to convert mathematical equation is high.

6.2 Evaluation of system by COST factor.

Rather than using all the complicated things for capturing image which are so costly, we are using raspberry PI 3B to capture image. We display overall cost require in our system below and then mention the requirement of other systems for this process.

	Device	**Price**
1	Raspberry pi Model 3B	\$35
2	5-volt 2Amp power Adaptor	\$10
3	Raspberry pi Mega Pixel Camera module	\$15
4	Minimum 16GB MicroSD card	\$7-15
	Total Cost	**\$67-95**

Table 6.3 Cost of System

While other systems require:

- Software Licensing Fees (One-Time Charge)
- Annual Software Maintenance Fees (annual license fee for software maintenance/periodic functional releases, etc.)
- Planning: Solution Requirements, Reviewing Business & IT environments, User Profiles, Testing, Training and Support Plans
- Installation, Configuration and Enrollment (Training Speakers)
- Training Material Creation & Delivery (Train-the –Trainer Model) (Assumes training & skills transfer for 5 people): End User, System Administrators, Client Support Help Desk or On-Line Support Line)
- Testing & Customer acceptance
- Post-installation support (for 2-year period up to 80 hours)

6.3 Evaluation of system by ACCURACY factor.

Two main modules are there in our system.

1. Gesture recognition
2. Text conversion

We have to find the accuracy of both this module. Following are measurements which are used to obtain accuracy for gesture recognition.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Here,

- ➢ TP is True Positive: test result is one that detects the condition when the condition is present.

- TN is True Negative: test result is one that does not detect the condition when the condition is absent.
- FP is False Positive: test result is one that detects the condition when the condition is absent.
- FN is False Negative: test result is one that does not detect the condition when the condition is present.

		Condition	
		Present	**Absent**
Test	**Positive**	True Positive	False Positive
	Negative	False Negative	True Negative

Table 6.4 Accuracy Parameter

1. Gesture Recognition Accuracy.

For gesture recognition we take all the combination which is possible for 5 fingers. For one finger we have 5 combinations, for 2 fingers we have 10 combinations, for 3 fingers we have 10 combinations, for 4 fingers we have 5 combinations and for 5 we have only one possibility. According to this we have 31 different gestures to recognize.

	Parameter	**Value**
1	Input images	32
2	TP	32
3	TN	00
4	FP	00
5	FN	00

Table 6.5 Results for gesture recognition

Table 6.5 shows the accuracy results for all the true positive, true negative , false positive, false negative parameters.

According to this result the final accuracy is

Accuracy ={(32+ 00)/ (32 + 00 + 00 + 00)*100}

Accuracy = 100%

2. Text Conversion Accuracy

For text conversion we are taking all the possibilities for A-Z and a-z alphabet, for numbers and for mathematical equations system's overall accuracy is 92% which is average calculation of simple text conversion's accuracy, complex text conversion accuracy and mathematical equations conversion accuracy (see table 6.6).

	Parameter	**Simple Text**	**Complex Text**	**Mathematical Equations**
1	**Input images**	100	100	50
2	**TP**	100	94	42
3	**TN**	00	00	00
4	**FP**	00	00	04
5	**FN**	00	06	04
6	**Accuracy**	**100%**	**94%**	**84%**

Table 6.6 Results for Text Conversion

Table 6.6 shows the image to text conversion Accuracy for simple text, complex text , mathematical equations.

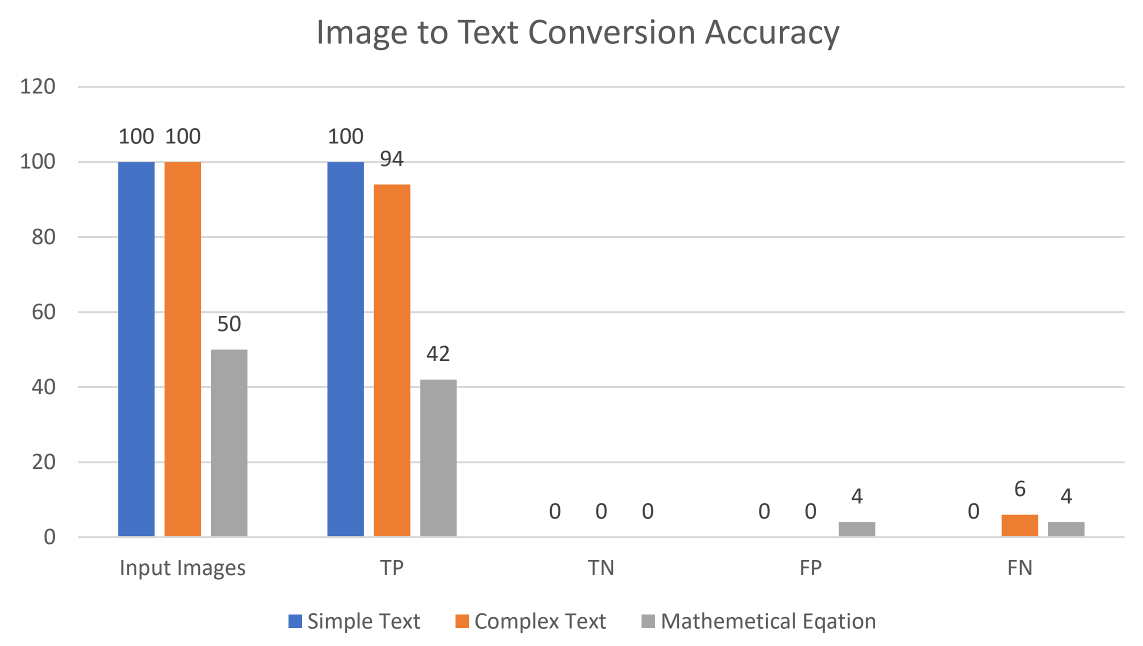


Fig 6.2 Results for Text Conversion

Here figure 6.2 shows the results for text conversion. X-axis contains the information about which kind of input we are taken and y-axis contains information about number of inputs, TP, TN, FP and FN vales for all the three-different kind of inputs.

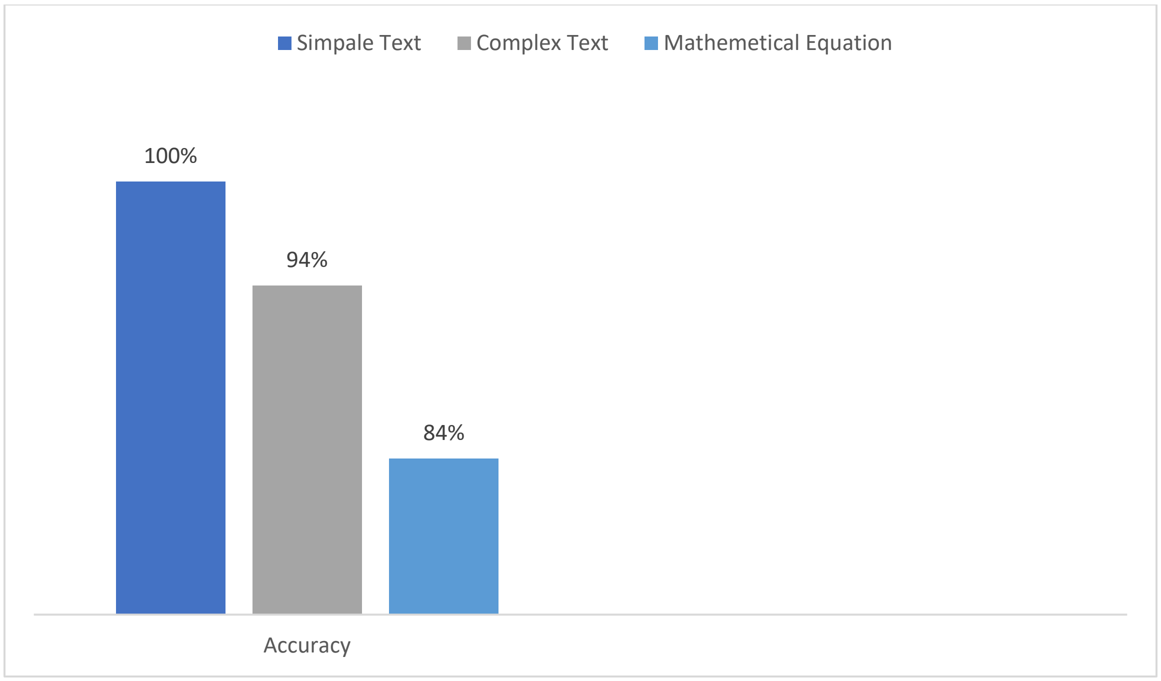


Fig 6.3 Accuracy for Text Conversion

Figure 6.3 shows the information about accuracy of text recognition. Here x-axis contains the information about the kind of inputs and y-axis gives the accuracy for that inputs. The Simple text are easy to convert as it has clear hand writing hence the system's accuracy is 100% , the complex text has attached letters so system gives false output to convert complex text. Hence, system's accuracy is 94% for complex text conversion. In mathematical equations the equations needed to be converted into latex format hence the accuracy is 84%.

CHAPTER 7

CONCLUSION & FUTURE WORK

In conclusion we would like to say that we have built a system to convert white board handwritten text images into editable format by following a unique approach from existing systems. We have tried to remove deficiency of existing system and made more accurate and efficient system. We have used hand gesture recognition to capture image of white board. To implement hand gesture recognition we have used our unique algorithm that gives 100% accuracy of hand gesture recognition. For image to text conversion we have built our own convolutional neural network that gives conversion accuracy of 92%. Also , we have built a system with very less cost of $50. Hence, we have built a cheapest and accurate system to provide lecture images into editable format for students.

<u>FUTURE WORK</u>

We have to make this system more advanced by converting mathematical equation images into text using machine learning algorithms. We want to implement voice to text conversion with hand gesture recognition for lecture video to text conversion.

REFERENCES

[1] Nidhibahen Patel, Selena He ,“A survey on hand gesture recognition techniques, methods and tools”, in International Journal of Research in Advent and Technology, Volume 6 , Issue 6 , June 2018.

[2] Gangaputra, Sachin. "Handwritten digit database". Retrieved 17 August 2013.

[3] Qiao, Yu (2007). "THE MNIST DATABASE of handwritten digits". Retrieved 18 August 2013.

[4] Platt, John C. (1999). "Using analytic QP and sparseness to speed training of support vector machines" (PDF). Advances in Neural Information Processing Systems: 557–563. Retrieved 18 August 2013.

[5] LeCun, Yann; Corinna Cortes; Christopher J.C. Burges. "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges". Retrieved 17 August 2013.

[6] Kussul, Ernst; Tatiana Baidyk (2004). "Improved method of handwritten digit recognition tested on MNIST database". Image and Vision Computing. 22 (12): 971–981. doi:10.1016/j.imavis.2004.03.008.

[7] Zhang, Bin; Sargur N. Srihari (2004). "Fast k -Nearest Neighbor Classification Using Cluster-Based Trees" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 26 (4): 525–528. doi:10.1109/TPAMI.2004.1265868. PMID 15382657. Retrieved 18 August 2013.

[8] "Support vector machines speed pattern recognition - Vision Systems Design". Vision Systems Design. Retrieved 17 August 2013.

[9] https://wikipedia.org/wiki/Otsu%27s_method

[10] Ravina Mithe, Supriya Indalkar and Nilam Divekar, “Optical character recognition”, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-2, Issue-1, pp 72-75 March 2001.

[11] Réjean Plamondon and Sargur N. Srihari, “On-line and off-line handwriting recognition: a comprehensive survey”, IEEE Transactions On Pattern Analysis And Machine Intelligence, VOL. 22, NO. 1, pp 63- 84, JANUARY 2000.

[12] Fred W. M. Stentiford, “Automatic feature design for optical character recognition using an evolutionary search procedure”, IEEE Transactions On Pattern Analysis And Machine Intelligence, VOL. PAMI7, NO. 3, pp 349-355, MAY 1985.

[13] Mulindwa, Desire Burume, Shengzhi Du, and Jacobus A. Jordaan. "An intelligent character recognition system for automatic mark capturing." Image and Signal Processing (CISP), 2014 7th International Congress on. IEEE, 2014.

[14] HGRYanan Xu, Dong-Won Park and GouCholPok, " Hand Gesture Recognition Based on Convex Defect Detection", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp. 7075-7079.

[15] M. Elleuch, R. Maalej, and M. Kherallah, "A New Design Based - SVM of the CNN Classifier Architecture with Dropout for Offline Arabic Handwritten Recognition," Procedia - Procedia Computer Science, vol. 80, pp. 1712–1723, 2016.

[16] B. Kwolek, "Face Detection Using Convolutional Neural Networks and Gabor Filters," *Artificial Neural Networks - ICANN*, pp. 551–556, 2005.

[17] Zhenzhen, Guan, et al. "Intelligent recognition for surface roughness based on microscopic image texture characters." Measurement, Information and Control (MIC), 2012 International Conference on. Vol. 1. IEEE, 2012.

[18] L. Chen, S. Wang, W. Fan, J. Sun, and N. Satoshi, "Reconstruction combined training for convolutional neural networks on character recognition," 2015 13th International Conference on Document Analysis and Recognition (ICDAR). pp.431–435, 2015.

[19] M. M. Santoni, D. I. Sensuse, A. M. Arymurthy, and M. I. Fanany, "Cattle Race Classification Using Gray Level Co-occurrence Matrix Convolutional Neural Networks," *Procedia Comput. Sci.*, vol. 59, no. October, pp. 493–502, 2015.

[20] Hussain, Rafaqat, et al. "Recognition based segmentation of connected characters in text based CAPTCHAs." Communication Software and Networks (ICCSN), 2016 8th IEEE International Conference on. IEEE, 2016.

[21] Mehta, Honey, Sanjay Singla, and Aarti Mahajan. "Optical character recognition (OCR) system for Roman script & English language using Artificial Neural Network (ANN) classifier." Research Advances in Integrated Navigation Systems (RAINS), International Conference on. IEEE, 2016.

[22] Singh, Dipti, et al. "An application of SVM in character recognition with chain code." Communication, Control and Intelligent Systems (CCIS), 2015. IEEE, 2015.

[23] Harpreet Kauri and Jyoti Rani, "A Review: Study of Various Techniques of Hand Gesture Recognition", IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), 2016.

[24] Mokhtar M. Hasan, and Pramod K. Mishra, “Hand Gesture Modeling and Recognition using Geometric Features: A Review”, Canadian Journal on Image Processing and Computer Vision, 2012, Volume 3, Issue 1.

[25] Srinivas Ganapathyraju, “Hand Gesture Recognition Using Convexity Hull Defects to Control an Industrial Robot”, 2013 3rd International Conference on Instrumentation Control and Automation (ICA) , Bali, Indonesia, August 28-30, 2013.

[26] Chun-Yao Wang, Ying-Chin Lin, Han Yuan Tan, Jing-Yun Zeng,” Understanding Mathematical Expressions from Camera Image” , in The 33rd Workshop on Combinatorial Mathematics and Computation Theory.

[27] J. S. Sonkusare, N. B. Chopade, R. Sor, and S. L. Tade, "A Review on Hand Gesture Recognition System, " 2015 Int. Conf Comput. Commun. Control Autorn., pp. 790-794, 2015.

[28] D. Q. J. Dqg, V. lurp, L. Pdun, O. Kdqg, H. W. Lv, K. Kinect, H. G. Recognition, Z. Fdq, E. H. Fodvvlilhg, E. S. Fodvvlilhuv, and V. Dv, "using Kinect Depth Camera, " pp. 5-8,2015.

[29] G. R. S. Murthy & R. S. Jadon, 2009. A Review if Vision Based Hand Gestures Recognition, International Journal of Information Technology and Knowledge Management, vol. 2(2), pp. 405-410.

[30] Laura Dipietro, Angelo M. Sabatini, and Paolo Dario, 2008. Survey of Glove-Based Systems and their applications, IEEE Transactions on systems, Man and Cybernetics, Part C: Applications and reviews, vol. 38(4), pp. 461-482, doi: 10.1109/TSMCC.2008.923862

[31] https://www.youtube.com/watch?v=v-XcmsYlzjA

[31]Tiecheng Liu and Chekuri Choudary,” CONTENT EXTRACTION AND SUMMARIZATION OF INSTRUCTIONAL VIDEOS”, in ICIP 2006, pp 149-152.

[32] HGR , SoukainaChraaMesbahi, “Hand gesture recognition based on convexity approach and background subtraction”, 2018 International Conference on Intelligent Systems and Computer Vision (ISCV).

[33] Seiji Okuni, Shinji Tsuruoka, Glenn P. Rayat, Hiroharu Kawanaka, Tsuyoshi Shinogi, “Video Scene Segmentation Using the State Recognition of Blackboard for Blended Learning”, in IEEE 2007 International Conference on Convergence Information Technology, pp 2437-2442.

[34] Ali Shariq Imran, Faouzi Alaya Cheikh, “LECTURE CONTENT CLASSIFICATION TOOL” ,in Proceedings of the 5th International Symposium on Communications, Control and Signal Processing, ISCCSP 2012, Rome, Italy, 2-4 May 2012.

[35] Zhengyou Zhang and Li-wei He, “NOTETAKING WITH A CAMERA: WHITEBOARD SCANNING AND IMAGE ENHANCEMENT”, in ICAPPS 2004, pp. III-533 to III-536.

[36] F. H. Yeh, G. C. Lee, I. J. Chen and C. H. Liao, “Robust Handwriting Extraction and Lecture Video Summarization”, in 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp 357-360.

[37] Markus Wienecke, Gernot A. Fink and Gerhard Sagerer, “Towards Automatic Video-based Whiteboard Reading”, in IEEE 2003.

[38] Marcus LIWICKI and Horst BUNKE,” Handwriting Recognition of Whiteboard Notes”.

[39] Luigi Lamberti& Francesco Camastra, 2011. Real-Time Hand Gesture Recognition Using a Color Glove, Springer 16th international conference on Image analysis and processing: Part I (ICIAP'11), pp. 365-373.

[40] S. Mitra, T. Acharya. “Gesture Recognition: A Survey”, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, pp. 311-324, 2007.

[41] K. Ryba, T. Mcivor, M. Shakir, and D. Paez, “Liberated Learning: Analysis of University Students’ Perceptions and Experiences with Continuous Automated Speech Recognition,” E-J. Instructional Science and Technology, vol. 9, no. 1, Mar. 2006

[42] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, “A unified framework for gesture recognition and spatiotemporal gesture segmentation,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 31, no. 9, pp. 1685–1699, 2009.

[43] Kajale, Renuka, Soubhik Das, and ParitoshMedhekar. "Supervised machine learning in intelligent character recognition of handwritten and printed nameplate." In Advances in Computing, Communication and Control (ICAC3), 2017 International Conference on, pp. 1-5. IEEE, 2017.