



# **CURRENCY IDENTIFIER FOR VISUALLY IMPAIRED**

## **A PROJECT REPORT**

*Submitted by*

**ROHIT S**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

**KCG COLLEGE OF TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600 025**

**June 2022**

## **BONAFIDE CERTIFICATE**

Certified that this project report titled “**CURRENCY IDENTIFIER FOR VISUALLY IMPAIRED**” is the bonafide work of “**ROHIT S (311018205037)**” who carried out the project work under my supervision.

Dr. J Frank Vijay

**HEAD OF THE DEPARTMENT**

Professor

Dept. of Information Technology

KCG College of Technology

Karapakkam.

Dr. S Cloudin

**SUPERVISOR**

Assosicate Professor

Dept. of Information Technology

KCG College of Technology

Karapakkam.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **ABSTRACT**

Visually impaired people are struggling to identify the denominations of the currency. The transactions are basically out of trust, but the current survey says that many are getting cheated because of the cruel people in this society. Visually impaired people are confident enough to rely on the existing system. The Braille system is not flooded everywhere, even though people teach braille many who lack education or cannot afford it cannot learn it.

Nowadays everything has become automated with the implementation of IoT and machine learning so new systems for visually impaired people are integrated with mobile phones in order to overcome the existing crisis it bring the hardware prototype method.

The datasets of various currencies are taken and trained to the model if the shown currency matches with the currency in the database then the denomination is recognized and information is passed to the user. Over the past few years, machine learning has been playing a vital role in society.

Its contribution to the field of information technology and to society is rapidly increasing and turning everybody to the new era of smart living. As the data collected is increased rapidly smart technological system has become a need for the up growing technology.

## ACKNOWLEDGEMENT

we thank the Almighty GOD for the abundant blessings shoitred on us. it extend the deepest love and gratitude to the dear parents who built up the careers and backed us up in life.

we thank the management and the Principal **Dr. P Deiva Sundari** for the opportunities given to us for the career development.

we feel indebted to the Head of the Department **Dr. J Frank Vijay**, Professor, Department of Information Technology, KCG College of Technology, for all his enctheagement, which has sustained the labor and efforts.

we extend the sincere thanks to the project coordinator and supervisor **Dr. S Cloudin**, Associate Professor for his guidance and support.

we would like to thank all other faculty members of the Department of Information Technology for their help and advice throughout the life on this campus.

Finally, it are thankful to all the friends and all others who enctheaged us and helped us in doing this project.

ROHIT S

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	ACKNOWLEDGEMENT	iv
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	xii
1	INTRODUCTION	
	1.1 OVERVIEW	1
	1.2 CHALLENGES IN MACHINE LEARNING	5
	1.3 OBJECTIVE	8
	1.4 MOTIVATION	8
	1.5 ORGANIZATION OF REPORT	9
2	RELATED WORK	
	2.1 LITERATURE REVIEW	10
	2.2 EXISTING TECHNOLOGY	12
	2.3 INFERENCE OF LITERATURE REVIEW	13
	2.4 EXTRACTION FROM LITERATURE REVIEW	15
3	SYSTEM ANALYSIS	
	3.1 PROBLEM DEFINITION	17

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	3.2 PROPOSED SOLUTION	17
	3.3 SOFTWARE COMPONENTS	18
	3.4 USE CASES	20
<b>4</b>	<b>SYSTEM DESIGN</b>	
	4.1 MODULES	25
	4.1.1 Exam Proctor	25
	4.1.2 BPO Proctor	27
	4.2 EVENT OCCURRENCE	16
	4.3 CLOUD ARCHITECTURE	30
	4.4 ARCHITECTURE DIAGRAM OF AI PROCTOR	31
<b>5</b>	<b>IMPLEMENTATION</b>	
	5.1 EXAM PROCTOR MODULE	32
	5.1.1 Code explanation	36
	5.2 BPO PROCTOR MODULE	36
	5.2.1 Code explanation	39

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>6</b>	<b>SYSTEM TESTING</b>	<b>40</b>
<b>7</b>	<b>OUTPUT AND EXPLANATION</b>	<b>46</b>
<b>8</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>53</b>
	<b>APPENDIX 1 : SAMPLE CODE</b>	<b>54</b>
	<b>APPENDIX 2: PAPER PRESENTATION</b>	<b>58</b>
	<b>REFERENCES</b>	<b>59</b>

## **LIST OF TABLES**

<b>TABLE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
2.1	Inference of Literature Review	13
2.2	Extraction from Literature Review	15
3.1	List of Software Components	18
6.1	Registration	40
6.2	Login Module	41
6.3	Authentication	42



## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.1	Types of Machine Learning	2
1.2	Supervised Machine Learning	2
1.3	Unsupervised Machine Learning	3
1.4	Reinforcement Learning	4
1.5	Poor Quality of Data	5
1.6	Non Representative Training Data	6
1.7	Underfitting and Overfitting	7
3.1	Use Case Diagram	19
4.1	Module Diagram	24
4.2	Exam Proctor Event Occurrence	28

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
4.3	BPO Proctor Event Occurrence	29
4.4	Cloud Architecture	30

## **LIST OF ABBREVIATIONS**

AI	Artificial Intelligence
CNN	Convolution neural network
DL	Deep Learning
IDE	Integrated development environment
KBA	Knowledge Based Authentication
ML	Machine Learning

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 OVERVIEW**

This project is purely based upon the core concepts of IoT and machine learning. It is a currency reader for the visually impaired people out there who are getting tricked by many people. It is not possible for everybody to learn the braille system and implement them in real-world cases. Therefore, this currency reader helps the visually impaired people to know what currency they are receiving currently or while paying it to the vendors basically the transactions. The upcoming growing technology Internet of things and machine learning is being used which will benefit hundreds of people out there. The IoT module is primarily aimed at mapping out a complex information system with the combination of sensor data, and efficient data exchange through networking. Machine learning artificial intelligence plays a vital role in this field. On the other hand, collecting information and maintaining, running together with privacy and security provision in IoT is the main issue. But then everything has a solution as in this case blockchain can be implemented and information can be secured. The classification predictive model is the task of approximating the mapping function from input to discrete output variables.

According to the experiments by researchers, the proportion of visually impaired people is higher. Braille system does not work with everybody. The smartphone application development makes it possible to identify currency with an attractive UI and voice recognition. Over the past few years, machine learning is playing a vital role in society, its contribution to the field of information technology and to society is rapidly increasing and turning everybody to the new era of smart living. As the data collected is increased rapidly smart technological system has become a need for the up growing technology. The author proposed a mobile paper currency detection system that applied to Saudi Arabian papers. Recognizing the paper currencies method is based on some interesting features and correlations between two images. It uses Radial Basis Function Network for classification. The system has an accuracy of recognition of 95.37% for the Normal

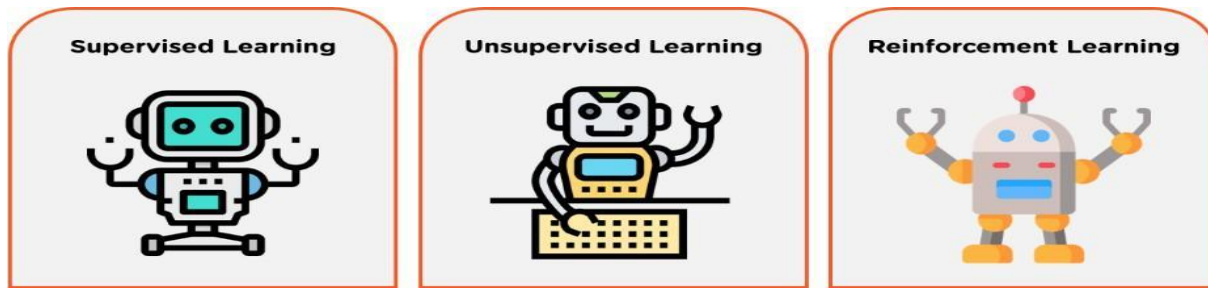
Non-Tilted Images, 91.65% for Noisy Non-Tilted Images, and 87.5% for Tilted Images. Sungwook et al. Proposed an efficient and fast Algorithm based on size information and correlation matching of multi templates. As different banknotes have different sizes so this information was regarded to be an important feature. This method was tested using 55 currencies of 30 different classes from five countries: EUR, KRW, RUB, CNY, and USD. The results of this method achieved 100% classification accuracy for normal banknotes and 99.8% classification accuracy for defiled banknotes. A non-parametric method is proposed for identifying paper currencies. The proposed method is based on the development of a nonparametric model for each class of paper currency. The model is obtained by averaging all available samples of one banknote. The tested banknote can be recognized by finding the values of the coefficients between the banknote and the nonparametric models and matching based on these values. For capturing the currency, the camera and currency should be aligned horizontally to get a good result. This method is applied to three kinds of Saudi Arabian banknotes and tested on a wide range of currencies and the accuracy reaches 100% of identification.

- **How machine learning works?**

The learning process starts with observations or data, such as instructions or examples, in order to detect patterns in data and make better future decisions based on the examples given into the system. Obviously, the goal is for computers to learn and modify themselves without the need for human involvement.

Machine learning techniques are either supervised or unsupervised. Refer figure 1.1, there are even other techniques apart from these two. Machine learning is now so common that you probably use it thousands of times per day without even realising it. Many academics believe it is the most effective technique to get closer to human-level AI.

Machine learning is now so common that you probably use it thousands of times every day without even realising it. Many academics believe it is the most effective method for advancing AI to the level of human intelligence.



As shown in figure 1.1, the following is the description of types of machine learning:

- **Supervised Learning**

Machine learning algorithms that are supervised "are taught using labelled instances, such as an input where the expected output is known." The learning method provides an inferred function to make predictions about the output values based on the analysis of a known training data set. The system may provide targets for any new input once it has been properly trained. The algorithm can then compare its output to the anticipated output and find faults, allowing the model to be adjusted as needed. It employs techniques such as regression, classification, prediction, and gradient boosting to forecast the value of the label on the unlabeled data using patterns.

It's frequently used in apps that forecast the future using historical data., For example, if a cat image is given to the trained machine learning model, then it predicts the probability of the cat using historical data used to train that model.

- **Unsupervised Learning**

Unsupervised learning techniques, on the other hand, are used when the data used for training is neither classed nor labeled. The algorithm must figure out what is being shown because the system is not given the "correct response." The idea is to look over the data and see if there is any structure there. On transactional data, this strategy works itll.

It can, for example, identify groups of clients with similar characteristics who can be targeted similarly in a marketing campaign. Nearest neighbor mapping, self-organizing maps, singular value decomposition, and k-means clustering are some of the most commonly utilized techniques. When a group of buses or truck characteristics is given as input to the trained unsupervised machine learning model, it will predict or

segregate those buses and truck instances separately from each other.

- **Semi-supervised Learning**

Because supervised and unsupervised machine learning use both unlabeled and labelled data for training, semi-supervised machine learning methods fall midway between unsupervised and supervised learning (although generally more of unlabeled than labeled).

The systems that improve this strategy can considerably improve learning accuracy. This strategy is typically employed when the incoming labelled data necessitates the training and learning of relevant/ competent restheces. Obtaining unlabeled data is also less expensive and time consuming.

- **Reinforcement Learning**

Since both unsupervised and semi-supervised versions of machine learning models use the same approach to learning with unlabeled data to achieve similar goals, reinforcement learning is a natural extension of self-supervised learning. Reinforcement learning, on the other hand, adds a feedback loop to the equation. When a reinforcement learning solution completes a task effectively, it is rewarded with positive feedback, which improves the model's ability to connect the desired inputs and outputs. It can also receive negative feedback if it provides inaccurate ansitrs. In some ways, the approach functions similarly to training with a dog using a reward system. The trained reinforcement machine learning model will be given an input of shape square and the model predicted it as square. Hence it is rewarded with positive feedback or points and rewarded by negative feedback or points to retrain itself from predicting this shape incorrectly in the future.

## **1.2 CHALLENGES IN THE DOMAIN**

- Poor quality of data
- Time consumption is more
- Needs battery backup
- Nonrepresentative training data

### **1.1.1 Poor quality of data**

To properly train the data to the model, one must have the required data set but then some problems arise here where the government or the organization keeps changing the physical features of the denominations every five years once where large computational process comes into play. Secondly, if the quality of the data set is poor then the accuracy rate decreases rapidly.

### **1.1.2 Nonrepresentative training data/inconsistent**

This would be a major drawback as the government is imparting new notes every 2 years once so training the model will get complicated.

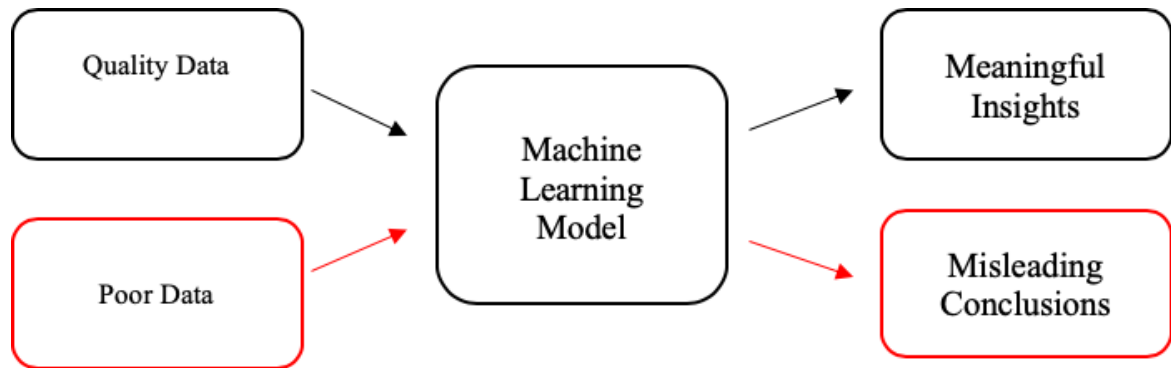
### **1.1.3 Time consumption is more**

As it requires parts to be assembled and algorithms to be implemented and trained, labor requirement is quite more and the rate at which the implementation is done takes quite a lot of time.

### **1.1.4 Needs battery backup**

As the hardware is poitred by low consumption elements battery will not be much of a problem but then charging it or might drain out quickly if it is used for a long time.





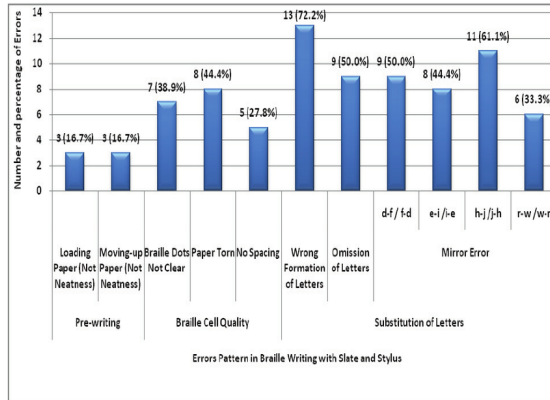
**Figure 1.2 Poor Quality of Data**

### **1.3 OBJECTIVE**

To develop a prototype for visually impaired people to identify the type of currency. This project is purely based upon the core concepts of IoT and machine learning. It is a currency reader for the visually impaired people out there who are getting tricked by many people. It is not possible for everybody to learn the braille system and implement them in real-world cases.

### **1.4 MOTIVATION**

Despite the quickly expanding utilization of Master cards and other electronic types of payment, money is still broadly utilized for ordinary exchanges because of its convenience. However, the visually impaired people may suffer from knowing each currency paper apart. Currency Recognition Systems (CRS) can be used to help blind and visually impaired people who suffer from monetary transactions. In this paper, a Currency Recognition System based on Oriented FAST and rotated BRIEF (ORB) algorithm is proposed.



How many blind people actually read braille?

The federation estimates that today only **one in 10** blind people can read Braille. That's down dramatically from the early 1900s. 13-Feb-2012



**Figure 1.3 Statistics from visually impaired site**

## 1.5 ORGANIZATION OF REPORT

This report is organized as follows:

Chapter 1 consists of an introduction to the domain (Machine Learning).

Chapter 2 consists of the related works including literature review and inference of the literature review.

Chapter 3 consists of the discussion on problem definition and proposed solution along with the software and hardware components.

Chapter 4 consists of the system design with the detailed explanation about the architecture diagram. Includes details of all modules along with the UML diagrams

Chapter 5 consists of the system implementation. Detailed discussion regarding the algorithms used in the project.

Chapter 6 consists of a detailed discussion regarding the testing components.

Chapter 7 consists of a detailed explanation regarding the output.

Chapter 8 discusses the conclusion and future work of the project. The fulfillment of the objective is also discussed in the conclusion.

Appendix 1 consists of sample python programming code to run currency identifier..

## **CHAPTER 2**

### **RELATED WORK**

#### **2.1 LITERATURE REVIEW:**

Trupti and Bawane [1] proposed a system that scans the whole paper for recognition. They considered the size as one of the three characteristics of paper currencies in addition to color and texture. Image histogram and the plenitude of different colors in a paper currency are calculated. The method discussed in this paper can be used for recognizing paper currencies from different countries. It also represents a currency recognition system using an ensemble neural network (ENN).

The individual neural networks in an ENN are skilled via negative correlation learning. The purpose of using negative correlation learning is to skill the individuals in an ensemble on different parts or portions of input patterns. They used ENN to identify new, old and noisy currencies. The Ensemble network is used for the categorization of different types of currency. It minimizes the chances of misclassification than a single network and ensemble network with independent training. They didn't make any experiments.

Yasir et al.[2] used Ftheier-Mellin transforms for invariant rotation, translation, and scale of the input image. The image is segmented and Markovian characteristics of each segment have been utilized to construct feature vectors. These vectors are then fed into an SVM classifier for paper currency recognition. This technique considered the whole paper currency was applied to both sides of the banknotes, and it could result in 98.7% accuracy in recognizing paper currencies. Due to using the texture characteristic, the system recognizes the banknotes in any direction.

Fuda et al. [3] recognize line graphs by tracing the connected components. The image is processed for the axes detection and is divided into the interior and exterior of the graph region. The connected components inside the axes are extracted for the line even in cases when multiple line charts intersect. In this, the connected component trace method can identify fthe different types of line graphs. When the lines are overlapping or in contagiousness to other lines, it is a subject that requires further research.

Huang and Tan [5] recognize the shapes from raster document images. Directional Single Connected Chain (DSCC) Algorithm [5] is used to detect the small straight lines on the edge image. Further, Curve fitting is used to form the lines, curves, and ellipses. Besides, the Curve fitting method is used by Huang et al. , in which they extract the graphical information in the documents by straight-line vectorization and extend this to other graphical entities with curve fitting. They use a data structure to keep information about the connected chains.

The curvature of the content in the data structure is evaluated and the chains are classified as lines or arcs. By this, they are able to extract the line, bar, and pie charts in images. This work contributes to the understanding of scientific charts.

Gaines[3] (2008) observed that recommendations based on practical experience of single users operating standard workstations had little to offer developers of complex systems integrating complex behavior of people and computers. To address this issue he presents a conceptual framework for person-computer interaction in complex systems based on an analysis of systems theory literature to derive design principles for person-computer interaction and a hierarchical model of person-computer systems.

He proposed the conceptual framework in the analysis of person-computer interaction in a complex system with six hierarchical layers: 1) Cultural layer: reflecting purpose and structure; 2) Intentionality layer: of anticipatory nature of an intelligent system that leads to the acquisition of knowledge; 3) Knowledge layer: that supports modeling and control activities of the anticipatory system; 4) Action layer: that transmits activities interfacing to the world; 5) Expression layer: that supports encoding of communications and actions and; 6) Physical layer: that addresses how encodings exist physically in the external world.

The Activity Diamond is a conceptual model developed by Per-Olof Hedvall for his Ph.D. thesis inspired by Cultural-Historical Activity Theory incorporating social and artefactual or natural contexts. Its application does not however appear to have been evaluated or validated either by independent expert review or experimentally.

There has, however, been no framework found that has helped technology developers to consider all of the possible interactions that occur at the same time and in the same place although there have been projects concerned with how to develop and use assistive technology to support some of these interactions. In order to ensure that the TEIF was a general framework, which could apply in many situations, a wide range of scenarios and technology solutions were considered during the development process.

Petrie et al.[6] (2002) investigated the use of universal interfaces for multimedia documents. They surveyed participants including blind and partially sighted plus experts using interviews and questionnaires and found that the users' requirements varied depending on their disabilities. For instance, one partially sighted reader may have particular problems in color blindness, or another may have no difficulties with color but require enlargement of text and graphics.

They developed a web-based tourist guide with formatted table contents for blind screen reader users and alt-text attributes for images. For partially sighted readers using screen magnification of text and images they allowed background and foreground colors to be easily adjusted with Scalable Vector Graphic (SVG) versions of images and maps, which allow the output to be zoomed by up to 4 times without quality degradation.

One of the most important problems facing visual impaired people is money recognition, especially for paper currency. In this paper, it presents a simple system currency recognition system applied on Egyptian banknotes. The proposed system is based on simple image processing utilities that ensure performing the process as fast and robust as possible. The basic techniques utilized in the proposed system include image foreground segmentation, histogram enhancement, region of interest (ROI) extraction, and finally template matching based on the cross-correlation between the captured image and the data set. The experimental results demonstrate that the proposed method can recognize Egyptian paper money with high quality reaching 89% in a short time.

## 2.2 EXISTING TECHNOLOGY

According to the Indian authorities, the big proportion of visually impaired people became higher. About one hundred and sixty-five people are visually impaired in line with lakh individuals. Blind people are eighty-two percent of them, and low vision was 18 percent. The latest smartphone development makes it possible to identify currency with an attractive one. But everybody cannot afford such end equipment, therefore it create hardware integrating all the activities that are required to identify the currency note.

The IoT module is primarily aimed at mapping out a complex information system with the combination of sensor data, and efficient data exchange through networking. Machine learning artificial intelligence plays a vital role in this field. On the other hand, collecting information and maintaining, running together with privacy and security provision in IoT is the main issue. But then everything has a solution as in this case blockchain can be implemented and information can be secured[1]. The classification predictive model is the task of approximating the mapping function from input to discrete output variables.

## 2.1 INFERENCE OF LITERATURE REVIEW

**Table 2.1 Inference of Literature Review**

<b>Title</b>	<b>Contribution</b>
Currency Recognition System for Visually Impaired: Egyptian Banknote as a Study Case	a method of artificial vision to recognize Mexican banknotes. Images captured are supposed to be taken under no illumination changes i.e. the input images of notes are illumination invariant. Here features like color and texture of the banknotes are extracted. On the basis of RGB space to extract color and the Local Binary Patterns to extract texture, respectively these features of banknotes are classified.

<p>Indian currency recognition for visually disable people</p>	<p>The fake currency can be detected with the extraction of existing features of banknotes. These features vary in accordance with the currency of the corresponding country. Here features considered are micro-printing, optically variable ink (OVI), water-marking, security thread and ultraviolet lines, etc. Sample currency note has to go through optical character recognition.</p>
<p>A Voice Based Assistant Using Google Dialogflow</p>	<p>The Speech Recognition Model is one of the most important part of a Virtual Assistant. Considering the various Neural Networks that are required for building up of a speech recognition system, it was necessary to survey the models that provided the insight by determining the accuracy and other factors of each Model. It was observed that High Accuracy and less Validation Accuracy was achieved for Convolutional Neural Network (CNN) model as compared to Basic Neural Network. Thus, proving that CNN is a better choice for speech recognition systems.</p>



**2.1 EXTRACTION FROM LITERATURE REVIEW**

**Table 2.2 Extraction from Literature Review**

Title	Extraction
Currency Recognition System for Visually Impaired: Egyptian Banknote as a Study Case	Currency Recognition Systems (CRS) can be used to help blind and visually impaired people who suffer from monetary transactions. In this paper, a Currency Recognition System based on Oriented FAST and rotated BRIEF (ORB) algorithm is proposed.

<p>Indian currency recognition for visually disable people</p>	<p>Currency recognition or bank-note recognition is a process of identifying the denominational value of a currency. It is a simple and straightforward task for the normal human beings, but if it consider the visually challenged people currency recognition is a challenging task. Visually handicapped people have a difficult time distinguishing betiten different cash denominations. Even though unique symbols are embossed on different currencies in India, the task is still too difficult and time-consuming for the blind.</p>
<p>A Voice Based Assistant Using Google Dialogflow</p>	<p>brings a deep need for automatic currency recognition systems. So, the paper studies about the systems in order to help the visually challenged or impaired people; so that they can differentiate betiten various types of Indian currencies through implementation of image processing techniques. The study aims to investigate different techniques for recognising Indian rupee banknotes. The proposed work extracts different and distinctive properties of Indian currency notes, few of them are the central number, RBI logo, colthe band, and special symbols or marks for visually impaired, and applies algorithms designed for the detection of each and every specific feature. From the work the visually impaired people will be capable of recognizing different types of Indian Currencies while their monetary transactions, so that they lead their life independently both socially and financially.</p>



## **CHAPTER 3 SYSTEM**

### **ANALYSIS**

#### **3.1 PROBLEM DEFINITION**

- According to the Indian authorities, the big proportion of visually impaired people became higher. About one hundred and sixty-five people are visually impaired in line with lakh individuals. Blind people are eighty-two percent of them, and low vision was 18 percent.
- The latest smartphone development makes it possible to identify currency with an attractive one. But everybody cannot afford such end equipment, therefore it create hardware integrating all the activities that are required to identify the currency note.

#### **3.2 PROPOSED SOLUTION**

Firstly, the device will be in the form of a pen where the bottom opening will be fitted with a micro camera and the top opening will consist of an output loudspeaker, camera will be connected to an RPi where the implementation part takes place. As soon as the camera senses a currency note image it will trigger the tinyML model where the model is stored in the cloud and the model will be trained using the dataset. After finding out the type of currency the data sends the currency number as text.

Using the text -to- speech module it convert the above text to a speech machine and implement it in the Arduino Nano 33 BLE Sense microcontroller. The RaspberryPi will be used to contact the Google Services on the Cloud to perform a more complex task once triggered by the Arduino. The project will be split into two parts:

Part 1: Emulating the machine learning model and speech Google Assistant on an RPi.

Part 2: Implementing a KWS on the Arduino Nano for the output and trigger.

The training test data for this can be taken from Kaggle and the model can be trained.

it will use the below-listed modules :

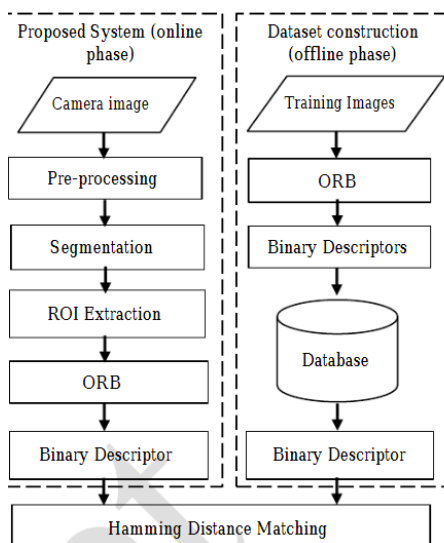
**gTTS** – Google Text To Speech, for converting the given text to speech

-> **pyaudio** – for voice engine in python

The dataset contains these fthe input characteristics:

- The variance of the image transformed into wavelets
- The asymmetry of the image transformed into wavelets
- Kurtosis of the image transformed into wavelets
- Image entropy

It needs to balance the data, the easiest way to do this is to drop a number of instances of the target function. This is called random undersampling. it will use Logistic Regression for Fake Currency Detection and classification. Open cv module will be used in order to detect the input image and the process happens then the google assistant speech module is triggered and it says the type of currency.



**Figure 3.1 Proposed system**

### 3.1 SOFTWARE AND HARDWARE COMPONENTS

Sl no.	Software Components
1	Anaconda(Python Distribution)
2	TensorFlow
3	Speech Recognition
4	OpenCV-python
5	Firebase cloud
6	Visual Studio Code
7	The RaspberryPi

- **Anaconda (Python Distribution)**

The project is planned to develop using Anaconda's Python 3 distribution as the main programming language which provides a runtime to run high-level machine learning models for malpractice prediction. Anaconda provides the high-level optimized scientific computing python libraries for development with its highly reliable package manager to install different python libraries needed which is supported by major operating systems like Linux, MacOS, Windows.

- **TensorFlow**

TensorFlow library is used to preprocess and make the images to fit according to the machine learning model's input parameter before providing that image to the model for malpractice prediction for camera detection using YoloV3 model. It is used to load the saved machine learning models for other predictions for face spoofing detection and for some array and image manipulation. In mobile phone and third party intervention detection, it is used to customize the output layer of the YoloV3 model for final prediction.

- **SpeechRecognition**

Google's SpeechRecognition python API is used to translate the audio recorded

from the candidate during the examination to a text file containing words spoken by that candidate using Recognizer class for audio recognition and AudioFile class to read the wav audio file to find out the common appearing words in the question paper.

- **OpenCV-python**

OpenCV-python library is used to capture and feed the frames of the live video of the candidate who is using the proctor to the machine learning models for malpractice prediction after doing some image preprocessing like resizing the image for all prediction before providing to all model, color channels transformation and calculating histograms for face spoofing detection, finding contours and moments for eyeball tracking and for other image manipulations.

- **Firebasecloud**

Firebase is a mobile app platform with integrated, unified client libraries in various mobile programming languages. Firebase's different backend-as-a-service (BaaS) features help you develop high-quality apps, grow the user base,

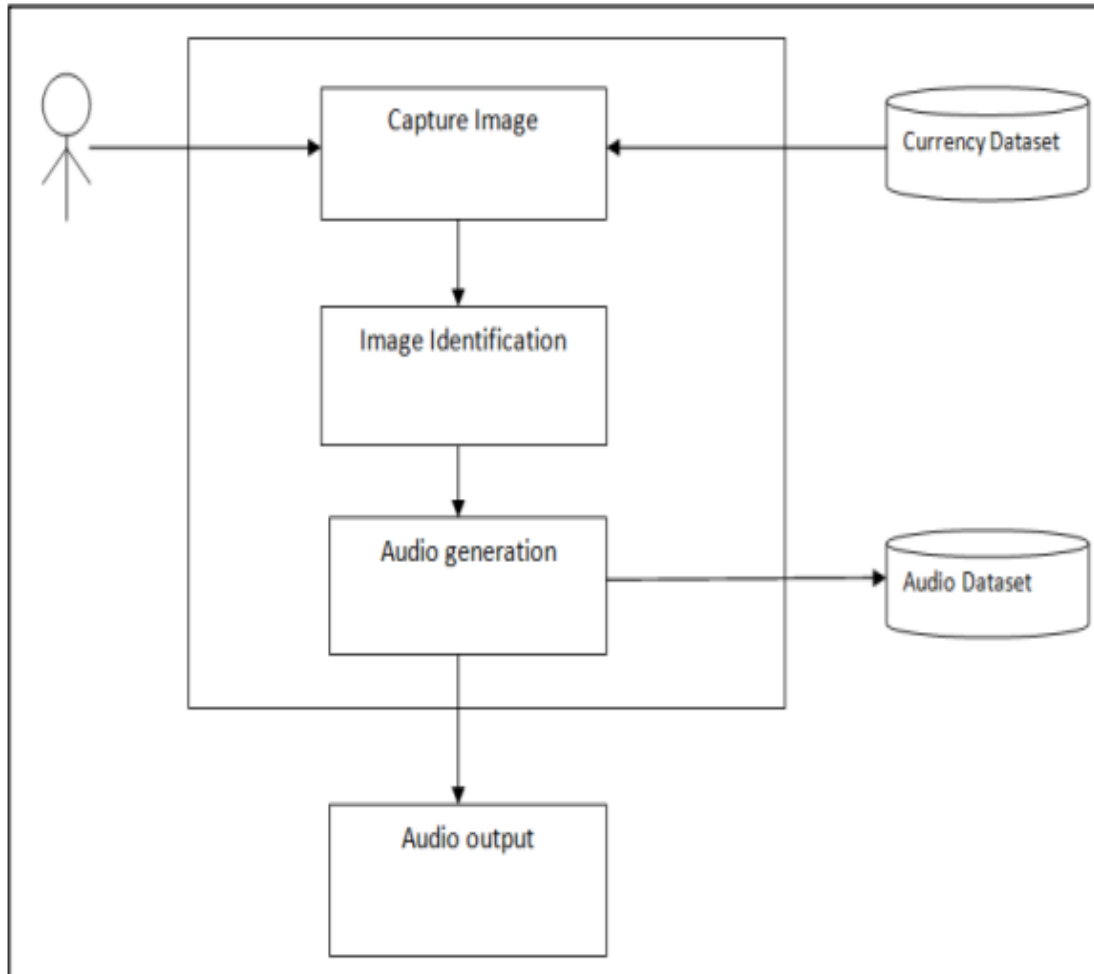
- **Visual studio code**

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

- **Raspberry**

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

### 3.2 USE CASES



**Figure 3.1 depicts the user case diagram showing the interaction with various actors and use cases.**

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other flow diagrams (activity, sequence, collaboration, and Statechart) also have the same purpose. It will look into some specific purpose, which will distinguish it from the other flow diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

When the initial task is complete, use case diagrams are modelled to present the outside view.

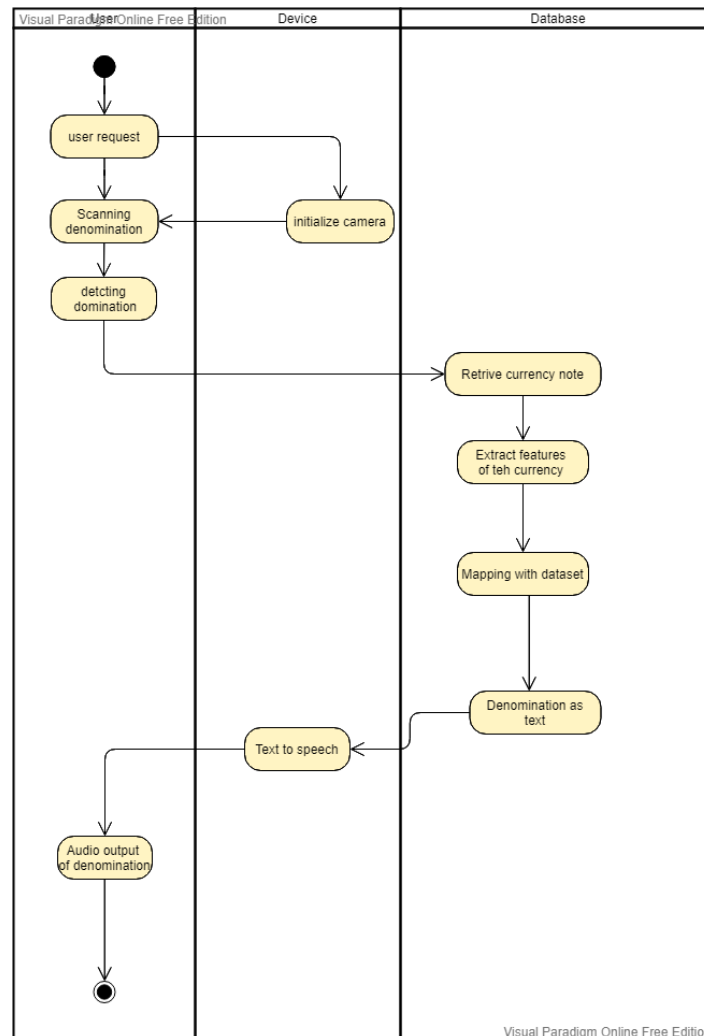
In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.



- Show the interaction among the requirements are actors.

### 3.3 ACTIVITY DIAGRAM



**Figure 3.3 Activity diagram**

- Initial State or Start Point

A small filled circle folloitd by an arrow represents the initial action state or the start point for any activity diagram. For the activity diagram using swimlanes, make sure the start point is placed in the top left corner of the first column.

- Activity or Action State

An action state represents the non-interruptible action of objects. You can draw an action state in SmartDraw using a rectangle with rounded corners.

- Action Flow

Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.

- Object Flow

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to action indicates that the active state uses the object.

## CHAPTER 4

### SYSTEM DESIGN

#### 4.1 MODULES

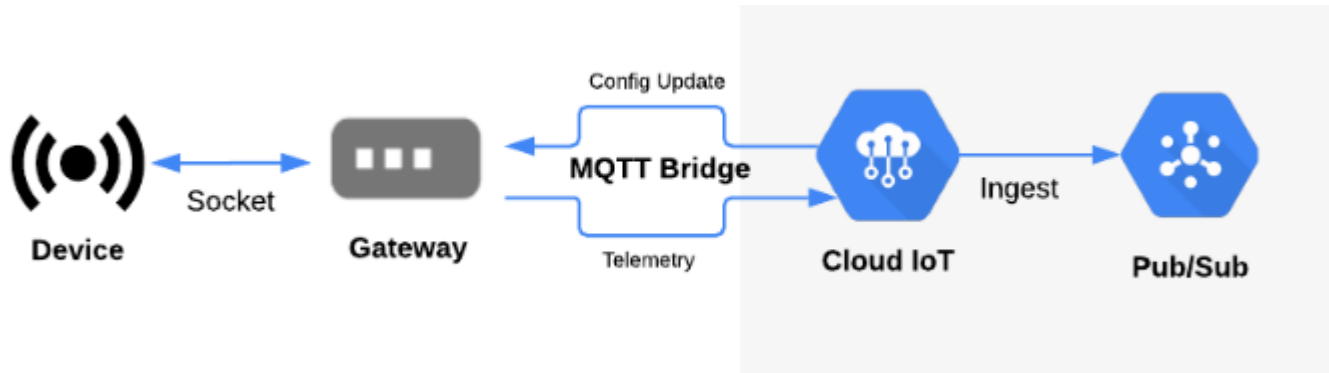


Figure 4.1 Module Diagram

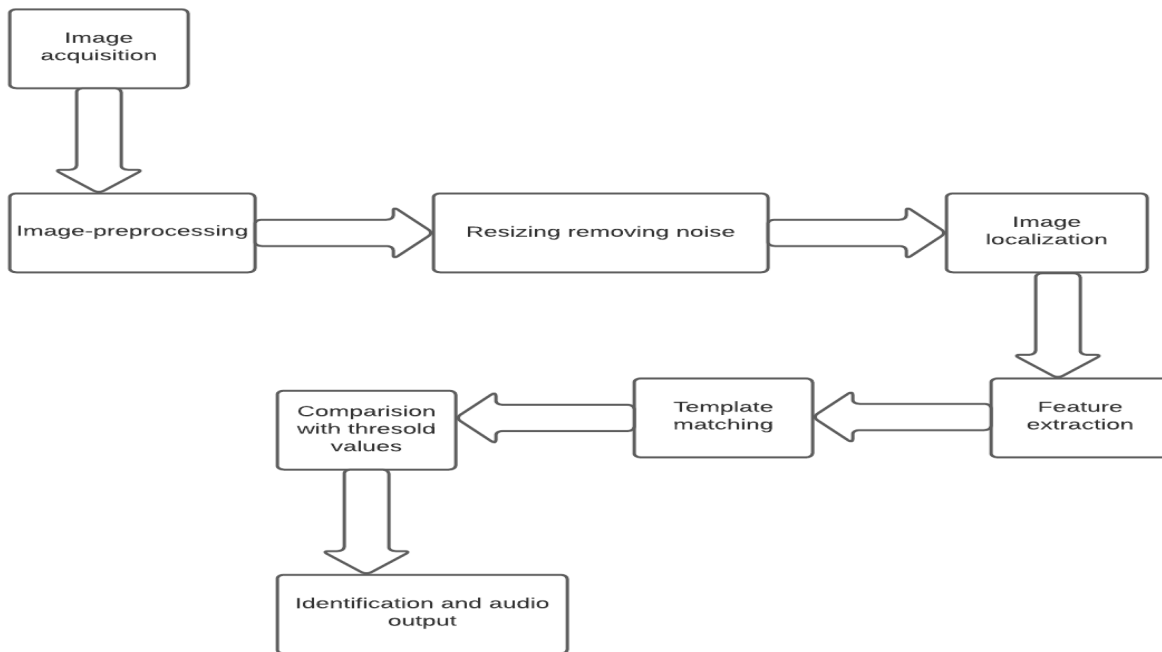


Figure 4.2 Detailed Module Diagram

Figure 4.1 and 4.2 depicts the module that has been implemented to build the project. The details are given in detail below:

In the preprocessing methodology, the model is trained with different images of currency notes and these will include the different currency denominations. Then the process of extraction comes into play by resizing the image that is present. Later the image is enhanced to make them brighter and darker points up to the contrasting level. This process will lead to feature extraction which produces the required feature points. The descriptors from the extraction will compare the sample imputed image and query image to find the probability of the highest match, the final result

will be retrieved in the form of text format and will be converted to speech module.

- The above-mentioned diagram explains the basic flow of image processing and enhancements. These steps are used in both BruteForceMatcher and the ConvolutionNeuralNetwork. The CNN algorithm is mainly used in the classification of images which will be effective for very large datasets.
- The SIFT and SURF detector and descriptor that was used for a very long time and even though it is comparatively old it has proved its success rate in a number of applications, including object recognition, image classifying and stitching, visual mapping development. It brings up large computational problems. Efficient replacement to the above-mentioned methods which has quite a large similarity in matching performance and is less affected by the general constraints and is highly capable of being deployed for real-time performance known as ORB.
- The image classification algorithm used in this project is CNN (convolution Neural Network) the scanned image undergoes various steps for the classification to occur. Firstly, the input image dimensions are flattened to one dimension(width pixels x height pixels) normalize the image pixel values by dividing them with standard values (255). Next, a model architecture has to be built with dense layers and the model has to be trained and should make predicted accordingly. One major advantage of using CNNs over NNs is that you do not need to flatten the input images to 1D as they are capable of working with image data in 2D. This helps in retaining the “spatial” properties of images.

The steps of a currency recognition system based on image processing are as follows –

- Image Acquisition
- Pre-processing
- Edge Detection
- Image Segmentation
- Feature Extraction
- Comparison

and finally, the output is displayed as pop up or can also be given as an audio output for visually impaired. The currency recognition always depends on the characteristics of the note belonging to certain country and the process of extracting different properties or features of currency note directly affects the ability of currency recognition.

Various algorithms based on image processing are been proposed over time to extract the features. These features consists of Security thread, Length/colthe of note, RBI logo, identification mark and other security features.

Feature extraction plays main role in recognition of currency notes. Hence, different algorithms are used for feature extraction. In this paper it will study fthe algorithms used in feature extraction

- FAST
- ORB
- SIFT
- SURF
- Image Acquisition  
It involves capturing the image of a currency note in the acceptable kind. The image is captured by a itb camera in RGB format.
- Image Acquisition  
In this step, the image is captured using a camera that is mounted on the head of the stick, and then the Gaussian Smoothing technique is applied to blur the image in order to remove the noise from the image.
- FAST  
FAST Algorithm is used for Image denoising which plays a very important role in the process of image processing. Features from accelerated segment test (FAST) could be a corner detection method. The purpose of the FAST algorithm was to develop an interest point detector to be used in real time like SLAM on a mobile robot, which have limited computational restheces.  
Algorithm: –  
Step 1: Choose a pixel a within the Currency image. Consider that the intensity of this pixel is IP. This is often the pixel which is to be identified as an interesting point or not.  
Step 2: Set a threshold intensity value T1 (1/5 of the pixel During test).  
Step 3: Choose a circle of 16 pixels around the pixel a.  
Step 4: If the value T is greater than or equal to IP, the adjacent pixels have to be above or below the interest point.  
Step 5: To create the algorithm fast, first compare the intensity of pixels 1, 5, 9 and 13 of the circles with IP. As evident from the figure above, at least three of these fthe pixels should satisfy the threshold criterion so that the interest point will exist.

Step 6: If three of the pixels are not above or below  $IP + T$ , then P is not a point. In that case reject the pixel p as a possible interest point. Else if three of the pixels are above or below  $IP + T$ , check for all 16 pixels and check whether 12 adjacent pixels fall in the criterion.

Step 7: For remaining pixels within the Currency image repeat the above steps.

- ORB

ORB Algorithm is to get the features of images because it may be a in no time algorithm regarding the time. In additionally, ORB is strong and efficient for getting binary descriptors. It basically depends on Oriented Fast and Rotated Brief Algorithm. The proposed application is implemented using the libraries of OpenCV run on the android platform. OpenCV libraries are having high speed in displaying result.

Algorithm: –

Step 1: Pre-processing

In this step, image processing operations are performed to arrange the currency image for the segmentation process.

Step 2: Segmentation

In this step, convert the currency image into a binary image that consists of two colors black and white.

Step 3: ROI Extracting

To extract the currency from the image, the two-pass connected component Algorithm is employed.

Step 4: Feature Extraction

In this step, the (ORB) Algorithm is employed to perform the subsequent operations for best performance:

- A quick algorithm is employed to detect corners and interesting points in a currency Image.
- Harris corner detector is employed to assign a score for each interest point supported the variation of intensities round the corner point.
- Compute the vector direction and also assign it because the interest point orientation using the interest point and centroid.
- Step 5: ORB Description

After getting those interesting points of the image within the detector stage, it'd like to extract the feature description for these interest points. For this purpose, a quick algorithm is employed to make the feature descriptors with relation to a neighborhood shape sort of a rectangle or circle.

#### Step 6: Matching

In this matching phase the best number of matches with the database informs that it matches with the currency. And also provides audio messages for blind people.

$$\text{Accuracy (\%)} = (\text{Success tests} / (\text{Success tests} + \text{Failure Test})) * 100$$

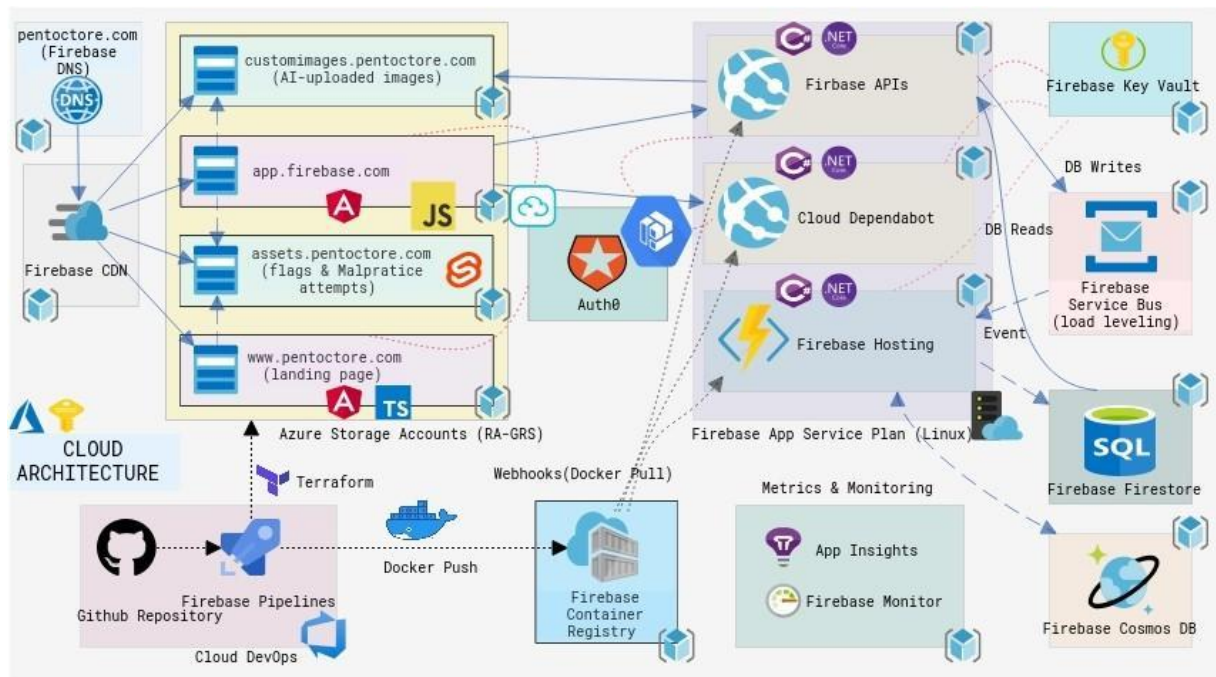
$$\text{Accuracy (\%)} = (\text{Success tests} / (\text{Success tests} + \text{Failure Test})) * 100$$

The accuracy of the proposed system is measured using the subsequent equation as below:

- SIFT

- It's a method for recognising steady, conspicuous feature points.
- It also gives a feature set for each such point that characterise a small image region around the point. These characteristics are unaffected by rotation or scale

## 4.2 CLOUD ARCHITECTURE



**Figure 4.2 Cloud Architecture**

In the above given figure 4.2, the AI Multi Modelling system is deployed in Host based Cloud Services with the help of Fire base and GCP containers. This is integrated with a Firebase CDN comprising 3 divisions to upload images. The Architecture here is a pipeline with docker to push the necessary Terrforce project integration needed for the algos to work. As soon as the App insights booleans to 1(true) the API'S AND Depandabot Allocate Service bus Load Levelling with the firebase fire store and DB To Mandate the Allocated the Malpractices on every single user entity.



### 4.3 ARCHITECTURE DIAGRAM

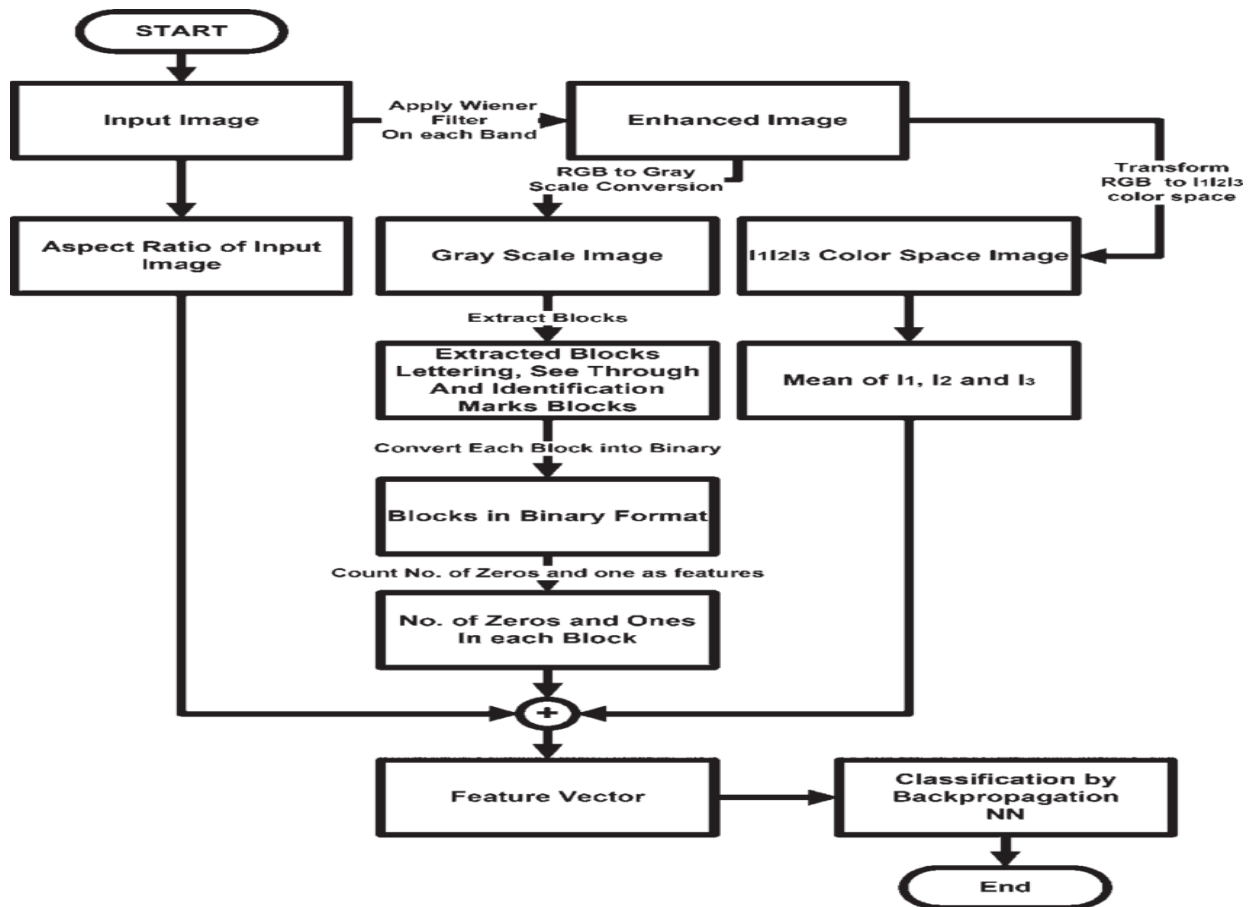


Figure 4.3 Architecture diagram

Any real-world system is used by different users. The users can be developers, testers, business people, analysts, and many more. Hence, before designing a system, the architecture is made with different perspectives in mind. The most important part is to visualize the system from the perspective of different users. The better it understands the better it can build the system.

UML plays an important role in defining different perspectives of a system. These perspectives are –

- Design
- Implementation
- Process
- Deployment

The center is the Use Case view which connects all these fthe. A Use Case represents the functionality of the system. Hence, other perspectives are connected with the use case.

The design of a system consists of classes, interfaces, and collaboration. UML provides a class diagram, an object diagram to support this.

Implementation defines the components assembled together to make a complete physical system. UML component diagram is used to support the implementation perspective.

Process defines the flow of the system. Hence, the same elements as used in Design are also used to support this perspective.

Deployment represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective.

## **CHAPTER 5**

### **IMPLEMENTATION**

#### **5.1 CREATING FIREBASE AND INPUT DETECTION WITH MATLAB**

- Implementing a python code to scan the displayed image and to store the preset of currencies to map with the output.
- With the help of the camera, the displayed currency is scanned.
- The IDLE shell is run, in-order to map the detected image with the preloaded input
- Once the dimensions of the detected input matches with the preloaded data, the detected denomination is displayed on the IDLE shell.

#### **5.2 TEXT-TO-SPEECH CONVERSION AND HARDWARE PROTOTYPE CONSTRUCTION**

- The device will be in the form of a pen where the bottom opening will be fitted with a micro camera and the top opening will consist of an output loudspeaker, camera will be connected to an RPi where the implementation part takes place. As soon as the camera senses a currency note image it will trigger the tinyML model where the model is stored in the cloud and the model will be trained using the dataset. After finding out the type of currency the data sends the currency number as text.
- After finding out the type of currency the data sends the currency number as text. Using the text-to- speech module it convert the above text to a speech machine and implement it in the RPi microcontroller. The RPi will be used to contact the Google Services on the Cloud to perform a more complex task once.

### 5.2.1 Code explanation

```

from gpt2 import GPT
from autoencoder import system as gpt

import os
from playground import playground

import tensorflow

from gpt2 import gpt

#GPT2 code depends with the default input image as shown below
max_len = 0
max_pt = 0
max_lr = 0

with tf.Session(graph=gpt)
# with an autoencoder to GPT

#GPT2_img = read_img('GPT2encoder_img_0.jpg')
#GPT2_img = read_img('GPT2encoder_img_1.jpg')
#GPT2_img = read_img('GPT2encoder_img_2.jpg')
#GPT2_img = read_img('GPT2encoder_img_3.jpg')
#GPT2_img = read_img('GPT2encoder_img_4.jpg')
#GPT2_img = read_img('GPT2encoder_img_5.jpg')
#GPT2_img = read_img('GPT2encoder_img_6.jpg')

# decoding result as image
original = decode_img(img, n)
reimg = original + original

# weights and descriptors

```

Here, the above code explains the input that is given as a image file and it process and compares with model that is the trained model.

```

(train, dev) = mh.detectAndCompute(train_img, None)

training_set = ['Files/en_jpg', 'Files/en_jpg', 'Files/en_jpg', 'Files/en_jpg']

for i in range(4, len(training_set)):
    # train image
    train_img = cv.imread(training_set[i])

    (train, dev2) = mh.detectAndCompute(train_img, None)

    # create feature matrix
    hf = cv.hconcat([
        all_matches = hf.imshow(hdist, dev2, hf)

    good = []
    # give an arbitrary number = 0.700
    # if good is present in list of good matches
    for (w, y) in all_matches:
        if multistance < 0.700 * multistance:
            good.append(w)

    if len(good) > max_val:
        max_val = len(good)
        max_y = y
        max_h = h

    print(i, ' ', len(training_set[i]), ' ', len(good))

```

Here, in the above image we extract the feature of the notes and convert it into grayscale for better comparison and sends it to the model and use image classification to detect the currency

12

[illegible]

```
C:\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib
r-opencv-master\detect.py'
0 files/20.jpg 10
1 files/50.jpg 10
2 files/100.jpg 4
1 files/500.jpg 9
files/20.jpg
good matches 10
Detected denomination: Rs. 20
```

13

It's a machine-learning algorithm for machines to understand the features of the image with foresight and remember the features to guess whether the name of the new image is fed to the machine.

1. A target image is a value to be predicted by the model
2. Give a set of inputs and their expected outputs
3. After the training, it will have a model that will then map new data to one of the categories trained on.
4. Once the model is ready, it can be fed with a set of inputs to which it will provide a predicted output.

In the preprocessing methodology, the model is trained with different images of currency notes and these will include the different currency denominations. Then the process of extraction comes into play by resizing the image that is present.

The training test data for this can be taken from Kaggle and the model can be trained.

The dataset contains these five input characteristics:

1. The variance of the image transformed into wavelets
2. The asymmetry of the image transformed into wavelets
3. Kurtosis of the image transformed into wavelets
4. Image entropy

## **5.3 MODULES AND IMPLEMENTATION**

Module 1: Dataset Training

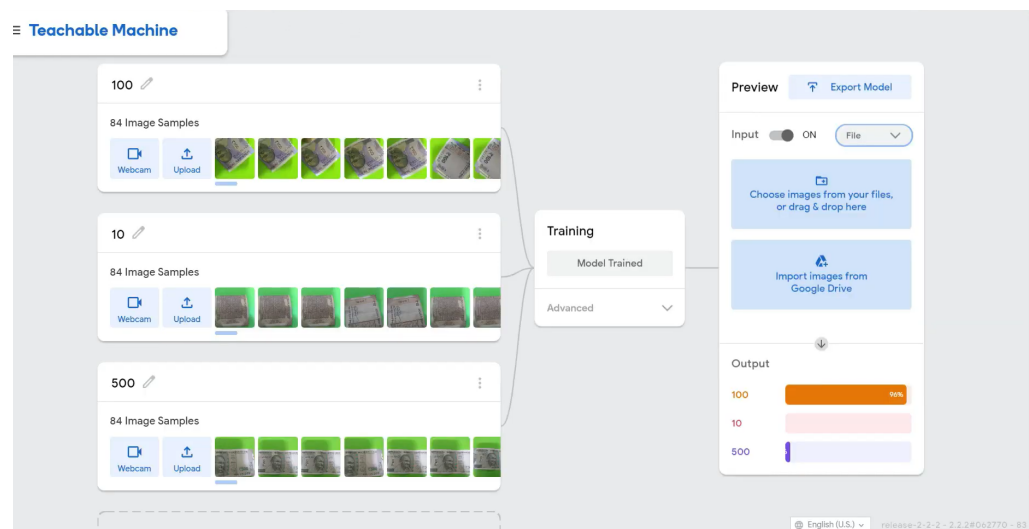
Module 2: Creating firebase and input detection with MATLAB

Module 3: Text-to-Speech Conversion and Hardware Prototype Construction

Creating firebase and input detection with MATLAB

1. Implementing a python code to scan the displayed image and to store the preset of currencies to map with the output.
2. With the help of the camera, the displayed currency is scanned.
3. The IDLE shell is run, in-order to map the detected image with the preloaded input

4. Once the dimensions of the detected input match with the preloaded data, the detected denomination is displayed on the IDLE shell.
  - a. Text-to-Speech Conversion and
  - b. Hardware Prototype Construction
5. The device will be in the form of a pen where the bottom opening will be fitted with a micro camera and the top opening will consist of an output loudspeaker, camera will be connected to an RPi where the implementation part takes place. As soon as the camera senses a currency note image it will trigger the tinyML model where the model is stored in the cloud and the model will be trained using the dataset. After finding out the type of currency the data sends the currency number as text.
6. After finding out the type of currency the data sends the currency number as text. Using the text-to-speech module it convert the above text to a speech machine and implement it in the RPi microcontroller. The RPi will be used to contact the Google Services on the Cloud to perform a more complex task once.



(FIG: 5.1 sthece: dataset training from [kaggle.com/training](https://kaggle.com/training))

**CHAPTER 6**  
**SYSTEM TESTING**

**Table 6.1 Registration**

Module Name:	Registration
Test Title:	To detect currency notes
Description	Test the detection
Test Designed by	Rohit s
Test Designed date	20/5/2022
Test Executed by	Rohit s
Test Executed date	20/05/22

Step	Test Case	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Show the currency	Rs 20. Rs 50, Rs 100	User should Detect	Detected successfully	Pass
2	Audio file	Rohit			

## CHAPTER 7

### OUTPUT AND EXPLANATION

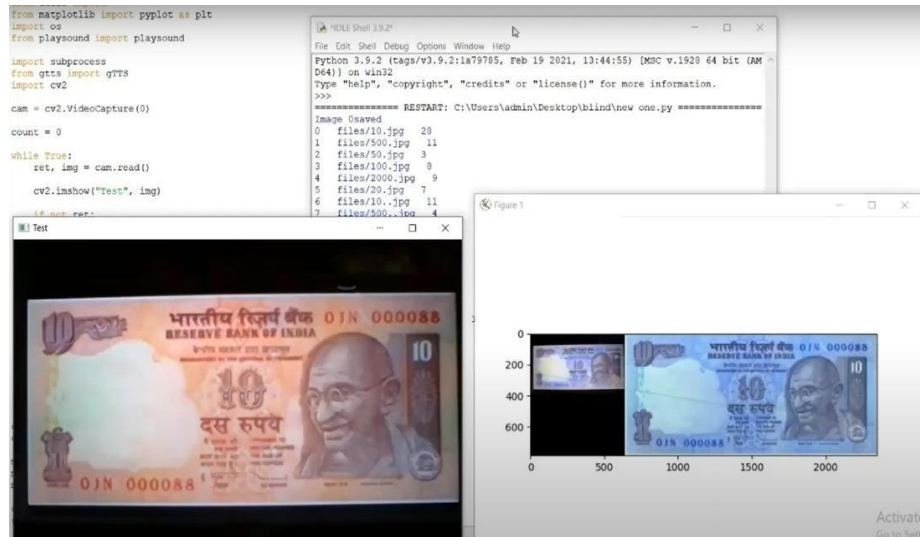


Fig 7.1 output

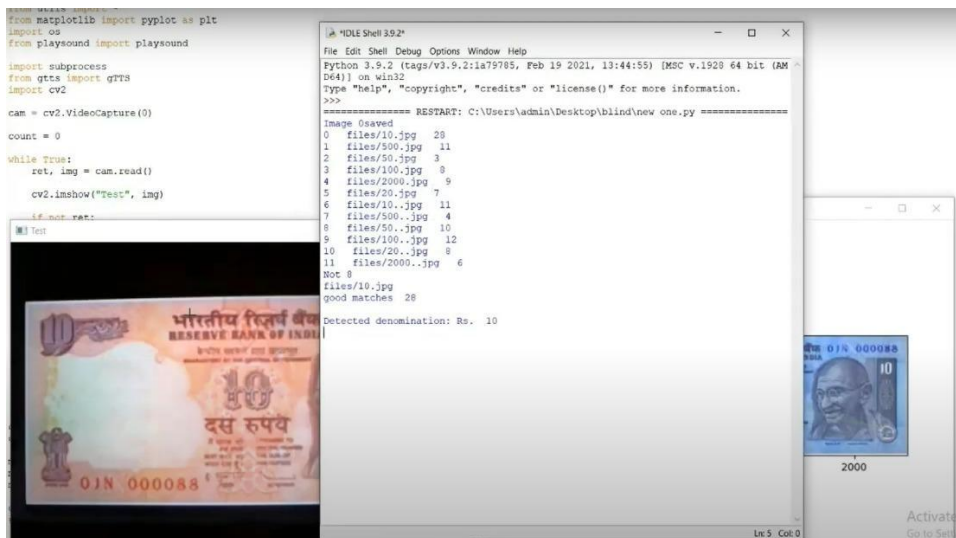


Fig 7.2 output



Therefore, as of now a preview of the model is yet to made and real-time currency detection model is trained and the output has been a successful one. The model is able to distinguish betiten the type of currencies. Works yet to be done is to integrate it with IoT and the cloud.

As it can see from the above stats many people in the current scenario do not utilize the current system effectively and the blame is not on them since many might not be educated to use braille or to check for a symbol identifier on a currency, basically each currency has a unique symbol to identify the denominations for visually impaired people but then in today's scenario nobody is able to identify it as they are not itll educated. Therefore, bringing this technology into the market will help many visually people out there who are struggling/ getting cheated by the cruel people out there.

## **CHAPTER 8**

### **CONCLUSION AND FUTURE ENHANCEMENTS**

#### **8.1 CONCLUSION**

The model detects the denomination of currency with the help of an image processing and classification algorithm and a handy hardware prototype and it can be implemented in real-time applications. Much future enhancement can be brought to the existing module such as handy personal assistance. This will require the inclusion of a few more pre-processing techniques and some modifications to the existing prototype to incorporate assistant features. Besides, the current system has poor accuracy for identifying coins other than notes due to the reflective nature of the material. Suitable lighting stheces can prevent this problem in the future. Therefore, the proposed prototype can rectify the main issue of visually impaired people and will help them to avoid getting cheated on.

#### **8.2 FUTURE ENHANCEMENTS**

The proposed system will be able to solve most of the crises that are currently occurring, this prototype is integrated with Machine Learning and the Internet of Things which are mostly used in the current era. Visually impaired people will be able to identify the denominations perfectly with the proposed system and can avoid getting cheated. Many more enhancements can be made to the proposed system when it is brought into use, and the better quality of data can be trained to the model such that the accuracy of prediction will be high. Secondly, with the overuse of the proposed system battery may get degraded soon so in order to enhance new lithium batteries can be updated. Thirdly, small assistance can be integrated into the system such that it'll accompany the person and help their peers to track them if they are lost. Many more enhancements can be made depending upon the user's interest. The product can also be brought out with assistants for visually impaired people and schedule their medical details in the cloud such that assistants can help the people with their medical intake by notifying them through the speaker.

## APPENDIX 1

### SAMPLE CODE

```
from utils import *
from matplotlib import pyplot as plt
import os
from playsound import playsound

import subprocess
from gtts import gTTS

#this code depicts with the default input image as shwon below
max_val = 8
max_pt = -1
max_kp = 0

orb = cv2.ORB_create()
# orb is an alternative to SIFT

#test_img = read_img('files/test_100_2.jpg')
#test_img = read_img('files/50.jpg')
#test_img = read_img('files/test_500_1.jpg')
#test_img = read_img('files/100.jpg')
test_img = read_img('files/test_20_4.jpg')

# resizing must be dynamic
original = resize_img(test_img, 0.4)
display('original', original)

# keypoints and descriptors
# (kp1, des1) = orb.detectAndCompute(test_img, None)
(kp1, des1) = orb.detectAndCompute(test_img, None)

training_set = ['files/20.jpg', 'files/50.jpg', 'files/100.jpg', 'files/500.jpg']
```

```

for i in range(0, len(training_set)):
    # train image
    train_img = cv2.imread(training_set[i])

    (kp2, des2) = orb.detectAndCompute(train_img, None)

    # brute force matcher
    bf = cv2.BFMatcher()
    all_matches = bf.knnMatch(des1, des2, k=2)

    good = []
    # give an arbitrary number -> 0.789
    # if good -> append to list of good matches
    for (m, n) in all_matches:
        if m.distance < 0.789 * n.distance:
            good.append([m])

    if len(good) > max_val:
        max_val = len(good)
        max_pt = i
        max_kp = kp2

    print(i, ' ', training_set[i], ' ', len(good))

if max_val != 8:
    print(training_set[max_pt])
    print('good matches ', max_val)

    train_img = cv2.imread(training_set[max_pt])
    img3 = cv2.drawMatchesKnn(test_img, kp1, train_img, max_kp, good, 4)

    note = str(training_set[max_pt])[6:-4]
    print("\nDetected denomination: Rs. ", note)

```

```
audio_file = 'audio/{ }.mp3'.format(note)
#audio_file = "value.mp3"
#tts = gTTS(text=speech_out, lang="en")
#tts.save(audio_file)
#return_code = subprocess.call(["afplay", audio_file])
playsound(audio_file)
(plt.imshow(img3), plt.show())
```

else:

```
print('No Matches')
```

```
#!/usr/bin/env python
```

```
# @Date: 2017-03-22
```

```
# UE14CS348 Digital Image Processing Mini Project
```

```
# Indian paper currency detection
```

```
# utils.py
```

```
# contains utility functions
```

```
import cv2
```

```
import math
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from pprint import pprint
```

```
# read image as is
```

```
def read_img(file_name):
```

```
    img = cv2.imread(file_name)
```

```
    return img
```

```
# resize image with fixed aspect ratio
```

```
def resize_img(image, scale):
```

```
res = cv2.resize(image, None, fx=scale, fy=scale, interpolation = cv2.INTER_AREA)
return res
```

```
# convert image to grayscale
```

```
def img_to_gray(image):
    img_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    return img_gray
```

```
# gaussian blurred grayscale
```

```
def img_to_gaussian_gray(image):
    img_gray = cv2.GaussianBlur(img_to_gray(image), (5, 5), 0)
    return img_gray
```

```
# convert image to negative
```

```
def img_to_neg(image):
    img_neg = 255 - image
    return img_neg
```

```
# binarize (threshold)
```

```
# retval not used currently
```

```
def binary_thresh(image, threshold):
    retval, img_thresh = cv2.threshold(image, threshold, 255, cv2.THRESH_BINARY)
    return img_thresh
```

```
# NO IDEA HOW THIS WORKS
```

```
def adaptive_thresh(image):
    img_thresh = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY, 11, 8)
    # cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst]) →
    dsta
```

```
return img_thresh
```

```
# sobel edge operator
```

```
def sobel_edge(image, align):
```

```
    img_horiz = cv2.Sobel(image, cv2.CV_8U, 0, 1)
```

```
    img_vert = cv2.Sobel(image, cv2.CV_8U, 1, 0)
```

```
    if align == 'h':
```

```
        return img_horiz
```

```
    elif align == 'v':
```

```
        return img_vert
```

```
    else:
```

```
        print('use h or v')
```

```
# sobel edge x + y
```

```
def sobel_edge2(image):
```

```
    # ksize = size of extended sobel kernel
```

```
    grad_x = cv2.Sobel(image, cv2.CV_16S, 1, 0, ksize=3, borderType = cv2.BORDER_DEFAULT)
```

```
    grad_y = cv2.Sobel(image, cv2.CV_16S, 0, 1, ksize=3, borderType = cv2.BORDER_DEFAULT)
```

```
    abs_grad_x = cv2.convertScaleAbs(grad_x)
```

```
    abs_grad_y = cv2.convertScaleAbs(grad_y)
```

```
    dst = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)
```

```
    return dst
```

```
# canny edge operator
```

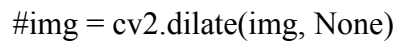
```
def canny_edge(image, block_size, ksize):
```

```
    # block_size => Neighborhood size
```

```
    # ksize => Aperture parameter for the Sobel operator
```

```
    # 350, 350 => for smaller 500
```

```
    # 720, 350 => Devnagari 500, Reserve bank of India
```

```
img = cv2.Canny(image, block_size, ksize)
# dilate to fill up the numbers

return img
```

```
# laplacian edge
def laplacian_edge(image):
    # good for text
    img = cv2.Laplacian(image, cv2.CV_8U)
    return img
```

```
# detect countthes
def find_contthes(image):
    (_, contthes, _) = cv2.findContours(image, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    contthes = sorted(contthes, key = cv2.contourArea, reverse = True)[:5]
    return contthes
```

```
# median blur
def median_blur(image):
    blurred_img = cv2.medianBlur(image, 3)
    return blurred_img
```

```
# dilate image to close lines
def dilate_img(image):
    img = cv2.dilate(image, np.ones((5,5), np.uint8))
    return img
```

```
# erode image
def close(image):
```



```

img = cv2.Canny(image, 75, 300)
img = cv2.dilate(img, None)
img = cv2.erode(img, None)
return img

```

```

def harris_edge(image):
    img_gray = np.float32(image)

    corners = cv2.goodFeaturesToTrack(img_gray, 4, 0.03, 200, None, None,
2,useHarrisDetector=True, k=0.04)
    corners = np.int0(corners)

    for corner in corners:
        x, y = corner.ravel()
        cv2.circle(image, (x, y), 3, 255, -1)
    return image

```

# calculate histogram

```

def histogram(image):
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])
    # cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])
    plt.plot(hist)
    plt.show()

```

# fast ftheier transform

```

def ftheier(image):
    f = np.fft.fft2(image)
    fshift = np.fft.fftshift(f)
    magnitude_spectrum = 20 * np.log(np.abs(fshift))

    plt.subplot(121), plt.imshow(image, cmap='gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])

```

```
plt.subplot(122), plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('FFT'), plt.xticks([]), plt.yticks([])

plt.show()
```

```
# calculate scale and fit into display
```

```
def display(window_name, image):
```

```
    screen_res = 1440, 900      # MacBook Air
```

```
    scale_width = screen_res[0] / image.shape[1]
```

```
    scale_height = screen_res[1] / image.shape[0]
```

```
    scale = min(scale_width, scale_height)
```

```
    window_width = int(image.shape[1] * scale)
```

```
    window_height = int(image.shape[0] * scale)
```

```
# rescale the resolution of the window
```

```
cv2.namedWindow(window_name, cv2.WINDOW_NORMAL)
```

```
cv2.resizeWindow(window_name, window_width, window_height)
```

```
# display image
```

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

```
# wait for any key to quit the program
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

```
#!/usr/bin/env python
```

```
# @Date: 2017-03-22
```

```
# UE14CS348 Digital Image Processing Mini Project
```

```
# Indian paper currency detection
```

```
# utils.py
```

```
# contains utility functions
```

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
from pprint import pprint

# read image as is
def read_img(file_name):
    img = cv2.imread(file_name)
    return img

# resize image with fixed aspect ratio
def resize_img(image, scale):
    res = cv2.resize(image, None, fx=scale, fy=scale, interpolation = cv2.INTER_AREA)
    return res

# convert image to grayscale
def img_to_gray(image):
    img_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    return img_gray

# gaussian blurred grayscale
def img_to_gaussian_gray(image):
    img_gray = cv2.GaussianBlur(img_to_gray(image), (5, 5), 0)
    return img_gray

# convert image to negative
def img_to_neg(image):
    img_neg = 255 - image
```

```
return img_neg
```

```
# binarize (threshold)
```

```
# retval not used currently
```

```
def binary_thresh(image, threshold):
```

```
    retval, img_thresh = cv2.threshold(image, threshold, 255, cv2.THRESH_BINARY)
```

```
    return img_thresh
```

```
# NO IDEA HOW THIS WORKS
```

```
def adaptive_thresh(image):
```

```
    img_thresh = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,  
cv2.THRESH_BINARY, 11, 8)
```

```
    # cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst]) →  
dsta
```

```
    return img_thresh
```

```
# sobel edge operator
```

```
def sobel_edge(image, align):
```

```
    img_horiz = cv2.Sobel(image, cv2.CV_8U, 0, 1)
```

```
    img_vert = cv2.Sobel(image, cv2.CV_8U, 1, 0)
```

```
    if align == 'h':
```

```
        return img_horiz
```

```
    elif align == 'v':
```

```
        return img_vert
```

```
    else:
```

```
        print('use h or v')
```

```
# sobel edge x + y
```

```
def sobel_edge2(image):
```

```
    # ksize = size of extended sobel kernel
```

```
    grad_x = cv2.Sobel(image, cv2.CV_16S, 1, 0, ksize=3, borderType = cv2.BORDER_DEFAULT)
```

```
grad_y = cv2.Sobel(image, cv2.CV_16S, 0, 1, ksize=3, borderType = cv2.BORDER_DEFAULT)
```

```
abs_grad_x = cv2.convertScaleAbs(grad_x)
```

```
abs_grad_y = cv2.convertScaleAbs(grad_y)
```

```
dst = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)
```

```
return dst
```

```
# canny edge operator
```

```
def canny_edge(image, block_size, ksize):
```

```
    # block_size => Neighborhood size
```

```
    # ksize => Aperture parameter for the Sobel operator
```

```
    # 350, 350 => for smaller 500
```

```
    # 720, 350 => Devnagari 500, Reserve bank of India
```

```
    img = cv2.Canny(image, block_size, ksize)
```

```
    # dilate to fill up the numbers
```

```
    #img = cv2.dilate(img, None)
```

```
    return img
```

```
# laplacian edge
```

```
def laplacian_edge(image):
```

```
    # good for text
```

```
    img = cv2.Laplacian(image, cv2.CV_8U)
```

```
    return img
```

```
# detect countthes
```

```
def find_countthes(image):
```

```
    (_, countthes, _) = cv2.findContours(image, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

```
    countthes = sorted(countthes, key = cv2.contourArea, reverse = True)[:5]
```

```
    return countthes
```

```
# median blur
```

```
def median_blur(image):  
    blurred_img = cv2.medianBlur(image, 3)  
    return blurred_img
```

```
# dilate image to close lines
```

```
def dilate_img(image):  
    img = cv2.dilate(image, np.ones((5,5), np.uint8))  
    return img
```

```
# erode image
```

```
def close(image):  
    img = cv2.Canny(image, 75, 300)  
    img = cv2.dilate(img, None)  
    img = cv2.erode(img, None)  
    return img
```

```
def harris_edge(image):
```

```
    img_gray = np.float32(image)
```

```
    corners = cv2.goodFeaturesToTrack(img_gray, 4, 0.03, 200, None, None,  
2,useHarrisDetector=True, k=0.04)  
    corners = np.int0(corners)
```

```
    for corner in corners:
```

```
        x, y = corner.ravel()
```

```
        cv2.circle(image, (x, y), 3, 255, -1)
```

```
    return image
```

```
# calculate histogram
```

```
def histogram(image):
```

```
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])
```

```
    # cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])
```

```
    plt.plot(hist)
```

```
    plt.show()
```

```
# fast ftheier transform
```

```
def ftheier(image):
```

```
    f = np.fft.fft2(image)
```

```
    fshift = np.fft.fftshift(f)
```

```
    magnitude_spectrum = 20 * np.log(np.abs(fshift))
```

```
    plt.subplot(121), plt.imshow(image, cmap='gray')
```

```
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
```

```
    plt.subplot(122), plt.imshow(magnitude_spectrum, cmap='gray')
```

```
    plt.title('FFT'), plt.xticks([]), plt.yticks([])
```

```
    plt.show()
```

```
# calculate scale and fit into display
```

```
def display(window_name, image):
```

```
    screen_res = 1440, 900      # MacBook Air
```

```
    scale_width = screen_res[0] / image.shape[1]
```

```
    scale_height = screen_res[1] / image.shape[0]
```

```
    scale = min(scale_width, scale_height)
```

```
    window_width = int(image.shape[1] * scale)
```

```
    window_height = int(image.shape[0] * scale)
```

```
    # rescale the resolution of the window
```

```
    cv2.namedWindow(window_name, cv2.WINDOW_NORMAL)
```

```

cv2.resizeWindow(window_name, window_width, window_height)

# display image
cv2.imshow(window_name, image)

# wait for any key to quit the program
cv2.waitKey(0)
cv2.destroyAllWindows()

#!/usr/bin/python
# @Date: 2017-03-22

# test file
# TODO:
#     Figure out the point transform
#     Figure out testing data warping
#     Use itbcam as input
#     Figure out how to use contours
#         Currently detects inner rect -> detect outermost rectangle
#     Try using video stream from android phone

from utils import *
from matplotlib import pyplot as plt

import subprocess
from gtts import gTTS

# image = read_img('files/500_1.jpg')
# orig = image
# orig = resize_img(orig, 0.5)
# img = resize_img(img, 0.6)

# img = img_to_gray(img)
# img = canny_edge(img, 720, 350)
# img = canny_edge(img, 270, 390)

```



```

# img = laplacian_edge(img)

# img = find_contthes(img)
# img = img_to_neg(img)
# img = binary_thresh(img, 85)
# img = close(img)
# img = adaptive_thresh(img)
# img = sobel_edge(img, 'v')
# img = sobel_edge2(img)
# img = median_blur(img)
# img = binary_thresh(img, 106)
# img = dilate_img(img)
# img = binary_thresh(img, 120)

# img = foo_convolution(img)
# histogram(img)
# ftheier(img)
# img = harris_edge(img)

# display('image',img)

# show the original image and the edge detected image
# cv2.imshow("Image", image)
# cv2.imshow("Edged", edged)
# cv2.waitKey(0)
# cv2.destroyAllWindows()

# find the contthes in the edged image, keeping only the
# largest ones, and initialize the screen contthe

# must define here

'''

```

```
kernel = np.ones((5,5), np.uint8)
```

```
img_erosion = cv2.erode(img, kernel, iterations=1)
```

```
img_dilation = cv2.dilate(img, kernel, iterations=1)
```

```
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
```

```
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

```
display('image', closing)
```

```
'''
```

```
'''
```

```
r = 500.0/ image.shape[1]
```

```
dim = (500, int(image.shape[0] * r))
```

```
image = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
```

```
#image = resize_img(image, 0.6)
```

```
ratio = image.shape[0] / 500.0
```

```
#display('image',image)
```

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
gray = cv2.GaussianBlur(gray, (5, 5), 0)
```

```
edged = cv2.Canny(gray, 75, 200)
```

```
'''
```

```
'''
```

```
show the original image and the edge detected image
```

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

```
cv2.imshow("Edged", edged)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

```
find largest contthes
```

```
'''
```

```
'''
```

```
(_,cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

```

cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]
#print "cnts: ", cnts
screenCnt = 0

n = .02
flag = True
while(n<.9 and flag==True):      #remove while loop if wrong contour is being detected
    print(n)
    for c in cnts:
        # approximate the contour
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c,n*peri, True)
        print("Approx: ", len(approx))

        # if the approximated contour has 4 points, then it
        # can assume that it have found the screen
        if len(approx) == 4:
            screenCnt = approx
            flag=False
            break

    n+=.01

warped = image
'''

'''
print('Screen count:', screenCnt)

cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 2)
cv2.imshow("Outline", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
'''

'''

```

```

warped = fthe_point_transform(orig, screenCnt.reshape(4, 2))
#warped = orig[ screenCnt[0][0][1]:screenCnt[1][0][1],screenCnt[0][0][0]:screenCnt[2][0][0]]
cv2.imshow('Warped',warped)
cv2.waitKey(0)
cv2.destroyAllWindows()
'''

'''

r = 500.0/warped.shape[1]
dim = (500, 240)
warped = cv2.resize(warped, dim, interpolation = cv2.INTER_AREA)

#cv2.imshow("Original", image)
cv2.imshow("Orignal", orig)
cv2.imshow("Scanned", warped)
cv2.waitKey(0)
cv2.destroyAllWindows()
'''

max_val = 8
max_pt = -1
max_kp = 0

orb = cv2.ORB_create()
# orb is an alternative to SIFT

#test_img = read_img('files/test_100_2.jpg')
#test_img = read_img('files/test_50_2.jpg')
test_img = read_img('files/test_20_2.jpg')
#test_img = read_img('files/test_100_3.jpg')
# test_img = read_img('files/test_20_4.jpg')

# resizing must be dynamic
original = resize_img(test_img, 0.4)
display('original', original)

```

```

# keypoints and descriptors
# (kp1, des1) = orb.detectAndCompute(test_img, None)
(kp1, des1) = orb.detectAndCompute(test_img, None)

training_set = ['files/20.jpg', 'files/50.jpg', 'files/100.jpg', 'files/500.jpg']

for i in range(0, len(training_set)):
    # train image
    train_img = cv2.imread(training_set[i])

    (kp2, des2) = orb.detectAndCompute(train_img, None)

    # brute force matcher
    bf = cv2.BFMatcher()
    all_matches = bf.knnMatch(des1, des2, k=2)

    good = []
    # give an arbitrary number -> 0.789
    # if good -> append to list of good matches
    for (m, n) in all_matches:
        if m.distance < 0.789 * n.distance:
            good.append([m])

    if len(good) > max_val:
        max_val = len(good)
        max_pt = i
        max_kp = kp2

    print(i, ' ', training_set[i], ' ', len(good))

if max_val != 8:
    print(training_set[max_pt])
    print('good matches ', max_val)

```

```
train_img = cv2.imread(training_set[max_pt])  
img3 = cv2.drawMatchesKnn(test_img, kp1, train_img, max_kp, good, 4)
```

```
note = str(training_set[max_pt])[6:-4]  
print("\nDetected denomination: Rs. ', note)
```

```
audio_file = 'audio/' + note + '.mp3'
```

```
# audio_file = "value.mp3"  
# tts = gTTS(text=speech_out, lang="en")  
tts.save(audio_file)
```

## APPENDIX 2

### PAPER PRESENTATION

Conference name : VIRTUAL INTERNATIONAL CONFERENCE ON FUTURISTIC COMMUNICATION AND NETWORK TECHNOLOGIES 2021 (VICFCNT-2021).

conference link: <https://chennai.vit.ac.in/files/vicfcnt/home.html>





**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)  
**CHENNAI**



VIRTUAL INTERNATIONAL CONFERENCE ON  
FUTURISTIC COMMUNICATION AND NETWORK TECHNOLOGIES  
(VICFCNT-2021)  
December 10-11, 2021

**Certificate of Appreciation**

This is to certify that

**Prof/Dr/Mr/Ms.** ..... *Rohit S* .....

**representing**

KCS college of technology - Chennai

**presented a paper titled**

Currency identifier for visually impaired people

**Dr. Sivasubramanian A**  
Convener & Dean-SENSE

**Dr. V. S. Kanchana Bhaaskaran**  
Pro Vice Chancellor

VIT – Recognised as Institution of Eminence (IoE) by Government of India  
VIT - A place to learn; A chance to grow



## REFERENCES

1. Trupti, Bawane (2021),” Currency Recognition System for Visually Impaired: Egyptian Banknote”, Springer., pp.1-25.
2. Yasir et al. (2008),” Currency Recognition System for Visually Impaired:”, Springer., pp.. 4220-4224.
3. Fuda et al,NRC (2021),” INDIAN CURRENCY RECOGNITION FOR BLIND PEOPLE”, Springer., pp.220-225.
4. Huang and Tan,ShittaYadav, (2020),” Eye Tracking and Head Movement Detection: A State-of-Art Survey," IEEE Jthenal of Translational Engineering in Health and Medicine, vol. 1, pp. 2100212 -2100212.
5. Bhaskar, N& Kumar, PM 2020, ‘Optimal processing of nearest-neighbor user queries in crowdsthecing based on the whale optimization algorithm’, Soft Computing, DOI:<https://doi.org/10.1007/s00500-020-04722-0>, ISSN 1432-7643.
6. Bhaskar, N, Mohan Kumar, P&ArokiaRenjit, J (2020), ‘Evolutionary Fuzzybased gravitational search algorithm for query optimization in crowdsthecing system to minimize cost and latency’, Computational Intelligence, DOI:<https://doi.org/10.1111/coin.12382>,Online ISSN:1467-8640.
7. Cloudin S., Mohan Kumar P., ”Adaptive mobility-based intelligent decision-making system for driver behavior prediction with motion nanosensor in VANET”, International Jthenal of Heavy Vehicle Systems,2018, Doi:10.1007/s40430-018-1286-2.
8. Frank Vijay J.”Cloud data analysis using a genetic algorithm-based job scheduling process”,2019 Expert systems 10.1111/exsy.12436.
9. Divvala S.K, D. Hoiem, J. H. Hays, A. A. Efros and M. Hebert (2009), "An empirical study of context in object detection," IEEE Conference on Computer Vision and Pattern Recognition., pp. 1271-1278..

