MORSE CODE DETECTION USING EYE BLINKS

A CREATIVE AND INNOVATIVE PROJECT REPORT

Submitted by

ISHWARYA RANI M (2019103527)

NAVVYA L (2019103548)

VISHNUPRIYA N (2019103599)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY:: CHENNAI 600 025

JUNE 2022

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **MORSE CODE DETECTION USING EYE BLINKS** is the bonafide work of **ISHWARYA RANI M (2019103537), NAVVYA L (2019103548) and VISHUNUPRIYA N (2019103599)** who carried out the project work under my supervision.

PLACE: CHENNAI **SIGNATURE**

DATE: 04.06.2022 **Mr. T.M. Thiyagu**

SUPERVISOR

Teaching Fellow

Department of Computer Science and Engineering

Anna University Chennai,

Chennai 600025

ABSTRACT

For ages, human beings have been communicating with one another through different modes of communication. One can communicate through speech, sign language and many other ways. Not all situations support the victim to convey help needed safely through hand signs or by speech. For private communication between two people, morse code has been developed which is highly efficient to exchange secrets. Morse code is a method used in telecommunication to encode text characters as standardized sequences of two different signal durations, called dots and dashes, or dits and dahs to represent alphabets. Morse code also helps in emergencies where a person cannot communicate through hand gestures. Different methods/modes are used in morse code, but our focus is on eye blinking. Signaling a dot in Morse code is as simple as blinking your eyes as quickly as you can. Signaling a dash in Morse code is as simple as blinking your eyes with a little 1-second pause. Every time a person/user blinks his/her eyes an output of dash and dot is generated which is Morse code. Our approach towards this area is to implement morse code using eye blinks in real-time assistance using a webcam to provide predicting power based on machine learning's tree algorithms. Also, this platform allows the end user to learn morse code by encoding a message. In this project, morse code is detected from a pattern of eye blinks using image processing.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	LIST OF FIGURES	vi
	LIST OF ABBREVIATION	vii
1.	**INTRODUCTION**	**1**
	1.1 Problem Statement	2
	1.2 Proposed Model	2
2.	**LITERATURE SURVEY**	**3**
3.	**METHODOLOGY**	**5**
	3.1 Overall Architecture Diagram	5
	3.2 Dataset	6
	3.2.1 Facial Landmarks	6
	3.2.2 Morse Code Dataset	6
4.	**MODULES**	**7**
	4.1 Locating Eyes	7
	4.1.1 Input	8
	4.1.2 Output	8
	4.1.3 Algorithm	8
	4.1.4 Implementation	8

	4.2	Determining Eye Aspect Ratio	10
		4.2.1 Input	10
		4.2.2 Output	10
		4.2.3 Algorithm	10
		4.2.4 Implementation	11
	4.3	Detecting Morse Code from Blinks	12
		4.3.1 Input	12
		4.3.2 Output	12
		4.3.3 Algorithm	13
		4.3.4 Implementation	13
	4.4	Encoding a Message into Morse Code	14
		4.4.1 Input	14
		4.4.2 Output	14
		4.4.3 Algorithm	14
		4.4.4 Implementation	15
5.	**OUTPUT SCREENSHOTS**		**16**
6.	**INNOVATIVE PLAN**		**18**
7.	**CONCLUSION AND FUTURE WORK**		**18**
8.	**REFERENCES**		**19**

LIST OF FIGURES

Fig No.	Figure Name	Page No.
3.1.1	Overall Architecture of the developed model.	5
3.2.1.1	68 feature points Model	6
3.2.2.1	Dictionary representing the morse code used for Encoding	6
3.2.2.2	Dictionary representing the morse code used for Decoding	7
4.1.4.1	Getting live stream from Webcam	8
4.1.4.2	Loading Face predictor landmarks dataset	9
4.1.4.3	Detecting face and rescaling	9
4.1.4.4	Detecting Eye coordinates	9
4.2.4.1	Loading coordinates and calculating EAR	11
4.2.4.2	Detecting if dot or dash using EAR	11
4.2.4.3	Dot and Dash are detected	12
4.3.4.1	Calling Converter Function	13
4.3.4.2	Converting morse code to text	13
4.3.4.3	Message is detected	14
4.4.4.1	Dictionary representing the morse code chart	15
4.4.4.2	Encoding the Message	15
4.4.4.3	English Message is encoded into morse code	16
5.1	Main Menu	16
5.2	Decoding message	17
5.3	Encoding message	17

LIST OF ABBREVIATIONS

LCD	Liquid Crystal Display	3
LIS	Locked In Syndrome	3
ML	Machine Learning	3
FGPA	Field Programmable Gate Array	4
IOT	Information Of Things	4
ALS	Amyotrophic Lateral Sclerosis	5
EAR	Eye Aspect Ratio	10

1. INTRODUCTION

Human communications are ordinarily through voices and expressions. Other than common dialects sign dialect comprising of facial expressions and body signals is another fundamental communication way for individuals beneath specific conditions. Not all situations support the victims or disabled people to convey help needed safely through hand signs or by speech. The communication between paralyzed patients and the individuals going to and caring for them were usually mouth incited joysticks, incited breathe puffing straws, tongue development examination, switch mounted close user's head, etc. These frameworks are expensive to actualize, increment push on the patients, and require talented labour to set up and keep up the framework for appropriate working. Barely any gadgets that have been created that can address this issue is not in a patient-friendly and cost-effective way. This work points to plan a basic and cost-effective computer program for patients or victims enduring from problems in communication with the alternative of using eye movements/blinks for communication.

The Morse code was created within the early 19th century when the individuals did not have any thought of developing circuits to send voice messages from one put to another. This strategy demonstrated that capability in English was the only prerequisite to communicate with the rest of the world. Afterward on, this code was acknowledged universally and a common Worldwide Morse code has been created and utilized. Morse code was prevalently known as the dialect of dabs and dashes. A long time afterward, this dialect came up with extemporized forms like transmitting content data as an arrangement of flipping tones, changing brightness levels, or ticks that can be straightforwardly decoded. It has been the foremost easiest and reasonable

strategy of transmitting and getting messages. Over the long time this strategy was basically utilized in radio communication but nowadays this technique has numerous applications in flying, naval force, and assistive strategy like making an individual with incapacities to communicate. Here using eye-blinks, the user can effortlessly express himself/herself.

1.1. PROBLEM STATEMENT

Communication is an important process through which a person can express his/her feelings and thoughts to the other person. Human beings have been communicating with one another through different modes of communication such as through speech, sign language and many other ways. Not all situations support the victim to convey help needed safely through hand signs or by speech. For private communication between two people, morse code has been developed which is highly efficient to exchange secrets through which people can communicate just using eye blinks. This method of communication is also useful for people with disabilities. Decoding and communication through morse code can be difficult for people who are new to morse code as well as for people who are familiar with morse code.

1.2. PROPOSED MODEL

The focus of this project is to develop a platform to implement morse code using eye blinks in real-time assistance using a webcam to provide predicting power based on machine learning's tree algorithms. This platform also allows the end user to learn morse code by encoding a message. In this project, morse code is detected using image processing.

2. LITERATURE SURVEY

Prof. Shalvi Tyagi, Anshul Gupta, Anamika Maurya, Arshita Garg, Kaustubh Vats, Amit Kumar. The video input is captured via the camera module, transferred to the Arduino UNO Board and displayed as text or voice signals on an LCD screen. It is developed with keen interest in helping the Locked-in Syndrome (LIS) paralysis patients and has a sustainable and effective solution. However, low lights affect the accuracy in prediction.

G Sumanth Naga Deepak, B Rohit, Ch Akhil, D Sai Surya Chandra Bharath and Kolla Bhanu Prakash, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Andhra Pradesh. OpenCV is used to open the webcam, .dlib is used to detect the face and eye region, FLASK connects ML model and the front-end part and as a proxy server to store ML model, fetch the output and display. Decision Tree and Random Forest algorithms are used, which provide higher accuracy compared to the existing systems. However, the time taken to execute is quite high, the process is tiresome and complicated and the occurrences of extra dots and dashes causes hindrance in prediction.

Kristen Grauman, Margrit Betke, James Gips, Gary R. Bradski. This author uses gradient flow fields, color-based techniques for detection of eye sclera, horizontal gradient maps using infrared lighting. 98% of the blinks are detected and has the ability to classify the eye blinks as involuntary(short) or voluntary(long). However, if a user begins to quint eyes for an extended period of time, his or her open eye template becomes out of date, and the system may give faulty outputs until the tracker is lost for some reason and re-initializes itself.

Pushya D, Shreyas N, Shreya P, Sandeep Varma N, Dept. of Information Science, B.M.S. College of Engineering Bengaluru, India. Haar Cascade

algorithm (face detection in each image) and Local Binary Pattern (face recognizer) are used along with histograms to depict the implementation of face recognition. It has an accurate face recognition algorithm, requires no additional hardware, no physical footprints are left and the processing time is fast. However, the system will fail to recognize the registered user when the lighting is poor and users may also find it difficult to remember complex Morse code compared to traditional alphabets and numbers.

Sushmitha M, Namrata Kolkar, Suman G S, Keerti Kulkarni, Dept of Electronics and Communication Eng, BNM Institute of Technology, Bangalore, India. It uses OpenCV, dlib and imutils to detect eye blinks through face pattern recognition. It has no after effect on the user, cost efficient and easily adaptable. However, this system only enables mouse clicks, but does not provide means to express their complete thoughts.

Sudhir Rao Rupanagudi, Vikas N S, Vivek C Bharadwaj, Manju and Dhruva N World Serve Education Bangalore, India. Sowmya K. S. Associate Professor Don Bosco Institute of Technology Bangalore, India. The camera is connected to a Xilinx Spartan 3e FPGA (Field Programmable Gate Array) with conversion to red chrominance for face detection and blink recognition. It was built to help people suffering from Motor Neuron Disease with 92% accuracy. The only failure case was detected when the set up was used in low light.

Juhi Singh a , Ritik Miglani, Chirag Goel a Amity School of Engineering and Technology, Noida, Delhi-201313, India. It uses OpenCV, web sockets and Flutter along with Viola Jones algorithm. It is made flexile, detection correctly decodes the morse code and security is established while Viola Jones algorithm helps in achieving high detection rates. But voice capabilities are not included.

Nayera Tarek, Mariam Abo Mandour, Nada El-Madah, Reem Ali, Sara Yahia, Bassant Mohamed, Dina Mostafa, Sara El-Metwally. The system implements a three layered IOT architecture, the perception layer, communication layer, and application layer. It is a cost-efficient wearable device. Frequently uses phrases by the users can be extended to the list to include additional phrases depending on their custom needs and abilities. Device suitable for a standard communication. ALS (Amyotrophic Lateral Sclerosis) patients will need more training time than reported in the testing experiment and the age, educational level, patient's awareness of the technology, and their acceptance will play a key role on the success of Morse Glasses usage.

3. METHODOLOGY

3.1. OVERALL ARCHITECTURE DIAGRAM

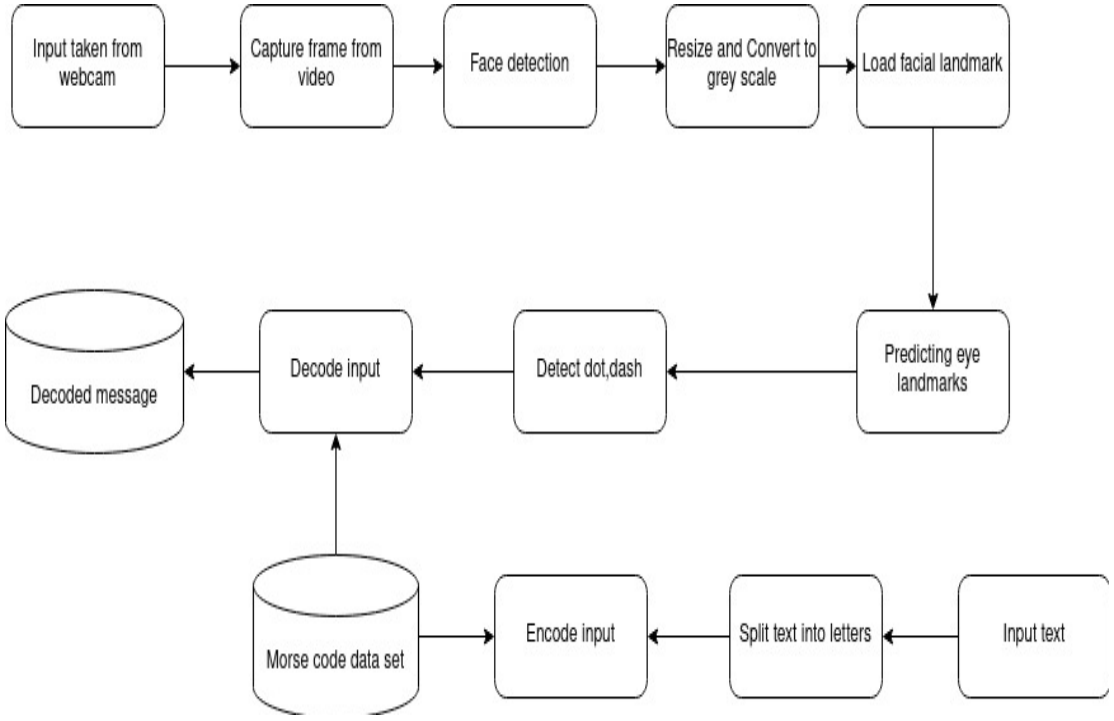

Fig. 3.1.1. Overall Architecture of the developed model.

3.2. DATASET

3.2.1 Facial Landmarks

The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.

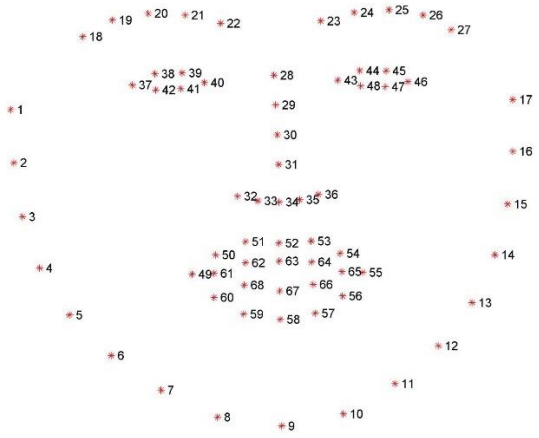

Fig. 3.2.1.1 68 feature points Model

3.2.2 Morse Code dataset

Morse code represents letters of the alphabet, numerals, and punctuation marks by an arrangement of dots, dashes, and spaces. This dataset is used for encoding and decoding.

```
MORSE_CODE_DICT = { 'A':'.-', 'B':'-...',
                    'C':'-.-.', 'D':'-..', 'E':'.',
                    'F':'..-.', 'G':'--.', 'H':'....',
                    'I':'..', 'J':'.---', 'K':'-.-',
                    'L':'.-..', 'M':'--', 'N':'-.',
                    'O':'---', 'P':'.--.', 'Q':'--.-',
                    'R':'.-.', 'S':'...', 'T':'-',
                    'U':'..-', 'V':'...-', 'W':'.--',
                    'X':'-..-', 'Y':'-.--', 'Z':'--..',
                    '1':'.----', '2':'..---', '3':'...--',
                    '4':'....-', '5':'.....', '6':'-....',
                    '7':'--...', '8':'---..', '9':'----.',
                    '0':'-----', ', ':'--..--', '.':'.-.-.-',
                    '?':'..--..', '/':'-..-.', '-':'-....-',
                    '(':'-.--.', ')':'-.--.-'}
```

Fig. 3.2.2.1 Dictionary representing the morse code used for Encoding

```
def dataset(c):
    morseDict={
    '.-': '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','.-.-': ' '
    }
```

Fig. 3.2.2.2 Dictionary representing the morse code used for Decoding

4. MODULES

4.1 LOCATING EYES

- Frame are captured from live video from web-cam or a recorded video.
- Captured frame is then resized and converted to grey scale image.
- Face detection technique is applied to detect faces on each frame.
- The eyes are predicted on the detected faces.
- The landmarks and converted into arrays.
- On these arrays the eye coordinates will be passed

4.1.1 Input

Live video captured from the web came is given as the input

4.1.2 Output

Array of the eye coordinates

4.1.3 Algorithm

```
# start the video stream thread
vs = FileVideoStream(args["video"]).start()
# grab the frame from the threaded video file stream,
frame = vs.read()
#resize it, and convert it to grayscale
frame = imutils.resize(frame, width=450)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# detect faces in the grayscale frame
rects = detector(gray, 0)
# determine the facial landmarks for the face region
```

```
shape = predictor(gray, rect)
# convert the facial landmark (x, y)-coordinates to a NumPy array
shape = face_utils.shape_to_np(shape)
# extract the left and right eye coordinates
leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]
```

4.1.4 Implementation

```
1   #-----Use VideoCapture in OpenCV-----
2   import cv2
3   import dlib
4   import math
5   from functions import *
6   from morse_converter import converter
7   BLINK_RATIO_THRESHOLD = 5.4
8
9   #-----livestream from the webcam-----
10  cap = cv2.VideoCapture(0)
```

Fig. 4.1.4.1 Getting live stream from Webcam

```
15  #-----name of the display window in OpenCV-----
16  cv2.namedWindow('DECODE')
17
18  #-----Face detection with dlib-----
19  detector = dlib.get_frontal_face_detector()
20
21  #-----Detecting Eyes using landmarks in dlib-----
22  predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
23  #-----these landmarks are based on the image above-----
24  left_eye_landmarks  = [36, 37, 38, 39, 40, 41]
25  right_eye_landmarks = [42, 43, 44, 45, 46, 47]
```

Fig. 4.1.4.2 Loading Face predictor landmarks dataset

```
flag=0
s=''
duration=''

while True:
    #-----capturing frame-----
    retval, frame = cap.read()

    #-----exit the application if frame not found-----
    if not retval:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    #-----converting image to grayscale-----
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #-----Face detection with dlib-----
    #detecting faces in the frame
    faces,_,_ = detector.run(image = frame, upsample_num_times = 0,
                             adjust_threshold = 0.0)
```

Fig. 4.1.4.3 Detecting face and rescaling

```
    #-----Detecting Eyes using landmarks in dlib-----
    for face in faces:
        landmarks = predictor(frame, face)
        #-----Calculating blink ratio for one eye-----
        left_eye_ratio  = get_blink_ratio(left_eye_landmarks, landmarks)
        right_eye_ratio = get_blink_ratio(right_eye_landmarks, landmarks)
        blink_ratio     = (left_eye_ratio + right_eye_ratio) / 2

```

Fig. 4.1.4.4 Detecting Eye coordinates

4.2 DETERMINING EYE ASPECT RATIO

- A metric called Eye Aspect Ratio will be computed to decide when an eye is closed or when an eye is opened.
- EAR is computed to calculate the flat and vertical remove of an eye.
- We will decide whether a wink has happened or not using the calculation of decreasing and increasing the EAR value.
- EAR = || p2 – p6 || + || p3 - p5 || / 2 || p1 – p4 ||

4.2.1 Input

Array of the eye coordinates

4.2.2 Output

Blink pattern

4.2.3 Algorithm

```
# function for calculating eye aspect ratio
def eye_aspect_ratio(self,eye):
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])
    self.ear = (A + B) / (2.0 * C)
    return self.ear
# use the coordinates to compute the eye aspect ratio for both eyes
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)

# average the eye aspect ratio together for both eyes
EAR = (leftEAR + rightEAR) / 2.0
```

4.2.4 Implementation

```python
import math

def midpoint(point1 ,point2):
    return int((point1.x + point2.x)/2), int((point1.y + point2.y)/2)

def euclidean_distance(point1 , point2):
    return math.sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)


def get_blink_ratio(eye_points, facial_landmarks):

    #loading all the required points
    corner_left  = (facial_landmarks.part(eye_points[0]).x,
                    facial_landmarks.part(eye_points[0]).y)
    corner_right = (facial_landmarks.part(eye_points[3]).x,
                    facial_landmarks.part(eye_points[3]).y)

    center_top    = midpoint(facial_landmarks.part(eye_points[1]),
                             facial_landmarks.part(eye_points[2]))
    center_bottom = midpoint(facial_landmarks.part(eye_points[5]),
                             facial_landmarks.part(eye_points[4]))

    #calculating distance
    horizontal_length = euclidean_distance(corner_left,corner_right)
    vertical_length = euclidean_distance(center_top,center_bottom)

    ratio = horizontal_length / vertical_length

    return ratio
```

Fig. 4.2.4.1 Loading coordinates and calculating EAR

```python
        if blink_ratio > BLINK_RATIO_THRESHOLD:
            #-----Blink detected!-----
            fl=f'{flag}'
            cv2.putText(frame,fl,(10,50), cv2.FONT_HERSHEY_SIMPLEX,
                        2,(255,255,255),2,cv2.LINE_AA)
            flag+=1
        else:
            #-----detect if dot or dash-----
            if flag>15 and flag<=30:
                s+='.'
            if flag>30 and flag<=50:
                s+='-'
            if flag>5 and flag<=15:
                s+=' '
            flag=0
```

Fig. 4.2.4.2 Detecting if dot or dash using EAR

Fig. 4.2.4.3 Dot and Dash are detected

4.3 DETECTING MORSE CODE FROM BLINKS

- Following the implementation of blink detection using EAR, morse code is produced from these blink patterns.
- When the user closes his eyes for 1 second, a '.'(dot) is detected
- When the user closes his eyes for 2 seconds, a '-'(dash) is detected.
- This dot and dash are saved before being translated to English using the morse code training and decoding data set.

4.3.1 Input

Blink pattern detected from module 2

4.3.2 Output

Decoded message

4.3.3 Algorithm

#Check to see if the eye aspect ratio is below the blink threshold
if ear < EYE_AR_THRESH:
    if eye_closed_duration==1:
        #dot is detected
    if eye_closed_duration==2:
        #dash is detected
#Map detected dots and dash to their corresponding alphabets
Store the dots and dash
Decode the dots and dash to the corresponding alphabets

4.3.4 Implementation

```
74          cv2.putText(frame,s,(10,100), cv2.FONT_HERSHEY_SIMPLEX,
75                      2,(255,255,255),2,cv2.LINE_AA)
76          cv2.putText(frame,converter(s),(10,150), cv2.FONT_HERSHEY_SIMPLEX,
77                      2,(255,255,255),2,cv2.LINE_AA)
78
```

Fig. 4.3.4.1 Calling Converter Function

```
6       MORSE_CODE_DICT = { 'A':'.-', 'B':'-...',
7                           'C':'-.-.', 'D':'-..', 'E':'.',
8                           'F':'..-.', 'G':'--.', 'H':'....',
9                           'I':'..', 'J':'.---', 'K':'-.-',
10                          'L':'.-..', 'M':'--', 'N':'-.',
11                          'O':'---', 'P':'.--.', 'Q':'--.-',
12                          'R':'.-.', 'S':'...', 'T':'-',
13                          'U':'..-', 'V':'...-', 'W':'.--',
14                          'X':'-..-', 'Y':'-.--', 'Z':'--..',
15                          '1':'.----', '2':'..---', '3':'...--',
16                          '4':'....-', '5':'.....', '6':'-....',
17                          '7':'--...', '8':'---..', '9':'----.',
18                          '0':'-----', ', ':'--..--', '.':'.-.-.-',
19                          '?':'..--..', '/':'-..-.', '-':'-....-',
20                          '(':'-.--.', ')':'-.--.-'}
32      if morseDict.get(c)!='Check':
33          return (morseDict.get(c))
34      else:
35          return ''
36
37  def converter(s):
38      code=s.split()
39      message=''
40      for c in code:
41          message+=dataset(c)
42          message+=' '
43      return message
```

Fig 4.3.4.2 Converting morse code to text

Fig. 4.3.4.3 Message is detected

4.4 ENCODING A MESSAGE INTO MORSE CODE

- Input is taken from user.
- The input sentence is spilt into word and then to letters.
- The letters are then encoded using the morse code data set

4.4.1 Input

English Message

4.4.2 Output

Encoded morse code

4.4.3 Algorithm

```
#function to convert alphabet to morse code
def convertTextToMorse(s):

    #part of dataset of morse code
    morseDict={
            ' ' : 'Check'
            'a' : '._'
            'b' : '_...'
            'c' : '_._.'
```

'd' : '_..'
'e' : '.'
'f' : '.._.'
'g' : '_ _.'
'h' : '....'
}
if morseDict.get(s)!='Check':
return (morseDict.get(s))

4.4.4 Implementation

```
def encrypt(message):
    # Dictionary representing the morse code chart
    MORSE_CODE_DICT = { 'A':'.-', 'B':'-...',
                        'C':'-.-.', 'D':'-..', 'E':'.',
                        'F':'..-.', 'G':'--.', 'H':'....',
                        'I':'..', 'J':'.---', 'K':'-.-',
                        'L':'.-..', 'M':'--', 'N':'-.',
                        'O':'---', 'P':'.--.', 'Q':'--.-',
                        'R':'.-.', 'S':'...', 'T':'-',
                        'U':'..-', 'V':'...-', 'W':'.--',
                        'X':'-..-', 'Y':'-.--', 'Z':'--..',
                        '1':'.----', '2':'..---', '3':'...--',
                        '4':'....-', '5':'.....', '6':'-....',
                        '7':'--...', '8':'---..', '9':'----.',
                        '0':'-----', ', ':'--..--', '.':'.-.-.-',
                        '?':'..--..', '/':'-..-.', '-':'-....-',
                        '(':'-.--.', ')':'-.--.-'}
```

Fig. 4.4.4.1 Dictionary representing the morse code chart

```
    cipher = ''
    for letter in message:
        if letter != ' ':
            # Looks up the dictionary and adds the
            # corresponding morse code
            # along with a space to separate
            # morse codes for different characters
            cipher += MORSE_CODE_DICT[letter] + ' '
        else:
            # 1 space indicates different characters
            # and 2 indicates different words
            cipher += '  '
    return cipher
```

Fig. 4.4.4.2 Encoding the Message

```
C:\Users\HP\myPython\CIP PROJECT>python converter2.py
Enter Message to encrypt:
hello how are you
.... . .-.. .-.. ---   .... --- .--   .- .-. .   -.-- --- ..-

C:\Users\HP\myPython\CIP PROJECT>python converter2.py
Enter Message to encrypt:
i am fine
..   .- --   ..-. .. -. .
```

Fig. 4.4.4.3 English Message is encoded into morse code

5. OUTPUT SCREENSHOTS

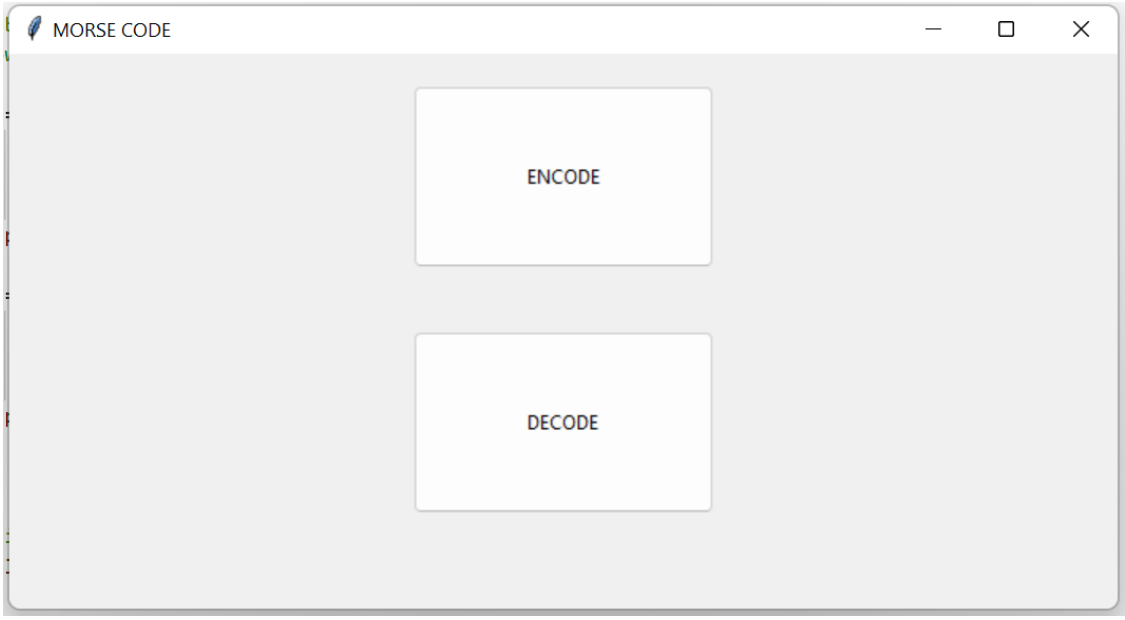

Fig. 5.1 Main Menu

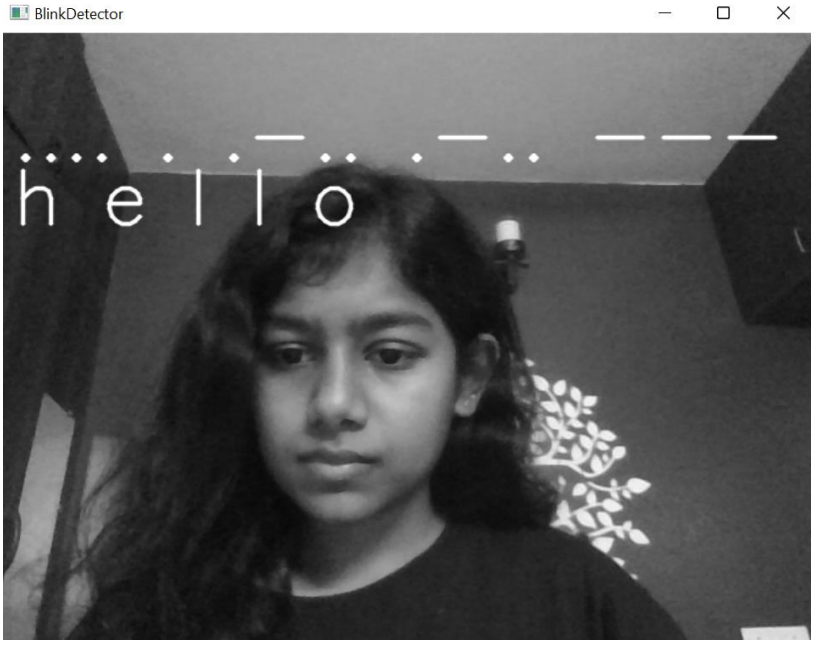

Fig. 5.2 Decoding message

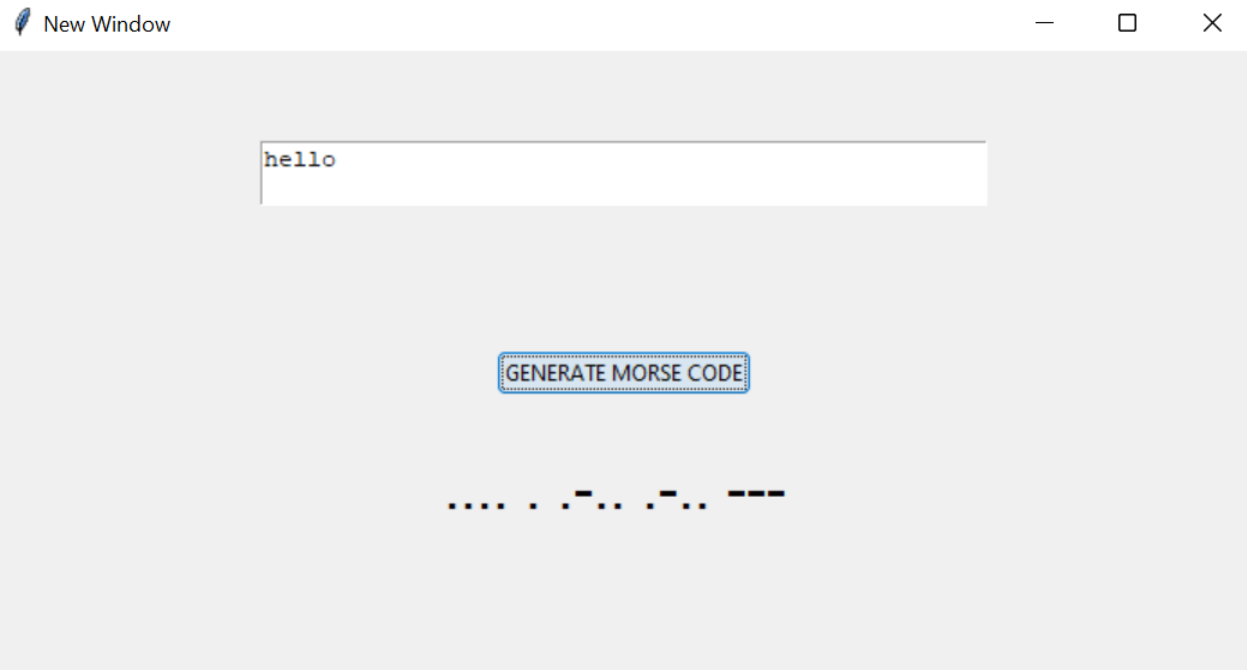

Fig. 5.3 Encoding message

6. INNOVATIVE PLAN

The current systems and developed projects either have implemented the morse code detection from eye blinks for the purpose of helping LIS (Locked-in Syndrome) or ALS (Amyotrophic Lateral Sclerosis) patients or for the purpose of secret communication. We have brought the use of such a communication for the purpose emergency situations where the victim is incapable of conveying through words along with the implementation to support patients who have difficulty in conveying messages to caretakers through speech and signs.

Another feature is, this system can classify blinks as voluntary (long) and involuntary (short) blinks since the camera module would be running in the background.

Voice capabilities are included, in case of patient's communication with care takers or family members making it easier as any normal person communicates. This feature proves to be a major advantage to people who are disabled for life and treatment for such disabilities are not possible.

We have overcome the problem that most of the earlier systems had in common that the system fails to recognize the registered user when the lighting and picture quality is poor.

Also, a user may not know or forget morse code, in which case, we have provided a text to morse code convertor to ease the learning of the user.

7. CONCLUSION AND FUTURE WORKS

Detection of morse code has been implemented using eye blinks in real-time assistance using a webcam to provide predicting power of the conveyed message. This platform also allows the end user to learn morse code by encoding a message. Morse code is detected from a pattern of eye blinks using image processing. This would help any victim of assault to give emergency signals during emergency situations without alerting the assailant. Patients with disabilities, who cannot convey their requirements properly through speech or sign, may use this technology to convey their needs through morse code, th 18 converted into text and voice signals.

Further this project can be integrated with emergency services and platforms. This may help people in emergency to convey an emergency message just using eye blinks and help can be sent. This project can be further integrated with mobile camera for more easy and handy communication.

REFERENCES

[1] A Novel Method for Eye Tracking and Blink Detection in video frames Leo Pauly, Deepa Sankar Division of Electronics and Communication Engineering School of Engineering Cochin University of Science and Technology

[2] Eye Blink Detection Using Local Binary Patterns Krystyna Malik, Bogdan Smolka Silesian University of Technology, Department of Automatic Control Akademicka 16 Str, 44-100 Gliwice, Poland

[3] Real-Time Eye Blink Detection using Facial Landmarks Tereza Soukupova and Jan Cech Center for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University in Prague

[4] Kranthi Kumar, V. Sai Srikar, Y. Swapnika, V. Sai Sravani, N. Aditya,"A Novel Approach for Morse Code Detection from Eye Blinks and Decoding using OpenCV".

[5] Timo Ahonen, Abdenour Hadid and Matti Pietikäinen, "Face Recognition with Local Binary Patterns", European Conference on Computer Vision