WHEELCHAIR CONTROLLER WITH EYE MONITORING

[1]MALLINATH, [2]DAMINI B R, [3]ZAIBA T A, [4]DIVYABHARATI, [5]Dr. R PRAKASH

DEPARTMENT OF ELELCTRICAL AND ELECTRONICS ENGINEERING
Acharya Institute of Technology

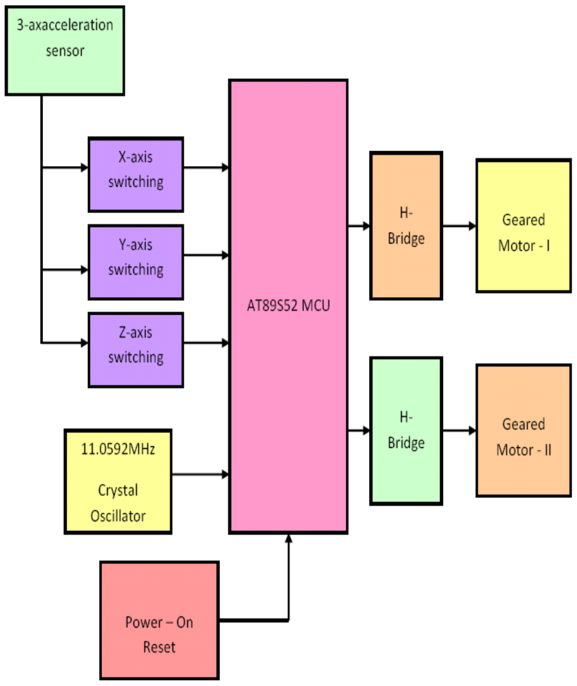
***Abstract*- One of the biggest curses on humanity is paralysis. In the worst scenario, just the eyes could be moved by the paralysed person. The voice- or head-based wheelchairs won't work well in that circumstance. Therefore, the optimal wheelchair for those people would be one that relies on eyeball movement. When compared to other automated wheelchairs, this would be more accurate. A technique for controlling wheelchairs using eyeball location is suggested. An algorithm develops an effective method to lower the cost and the computational complexity by providing numerous processing steps. Real-time eye detection and monitoring were the main objectives. The concept is to develop an eye-monitored system that moves the patient's wheelchair in accordance with eye movements. A patient can move in one direction by just looking in that direction while looking directly at the camera that is put on a headgear.**

***Keywords*- System that monitors the eyes and takes pictures of the eyes PWD (person with a disability), eye detection, and pupil detection;**

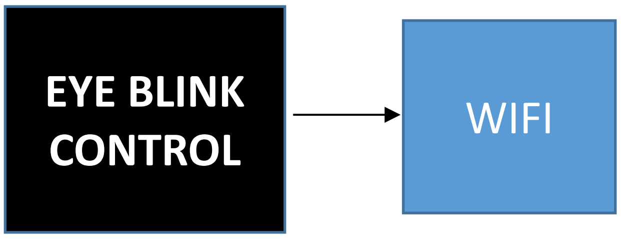
1. INTRODUCTION

With a growing population, there has been a sharp rise in the number of people who are susceptible to quadriplegia. For people with disabilities, several wheel chair systems have been developed. The systems for wheelchairs that are currently in use are mentioned below. The MEMS Accelerometer Sensor used by hand gesture-based wheelchair systems is mounted to the hand. The wheelchair control device is then operated via hand gestures. The voice of the user is used to operate the wheelchair through a voice operated wheelchair system. Accelerometer and Flex Sensor are used by Head and Finger based Automated Wheelchair System to operate the wheelchair. However, each of the aforementioned devices requires a significant amount of human work, and none of them are helpful for quadriplegics. When a person is quadriplegic, their level of paralysis is so severe that they are only able to move their eyes.

2. BLOCK DAIGRAM



2.1 TRANSMITTER



3. METHODOLOGY

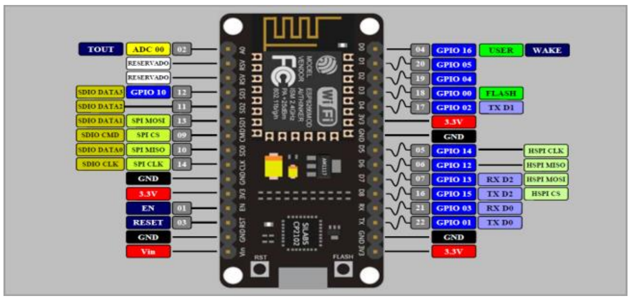
3.1) In reaction to an eye movement that a head-mounted camera picks up, a wheelchair is moved.
3.2) The head mount camera is connected to a laptop, and a running MATLAB script analyses the image and transmits commands to the microcontroller to move the wheelchair's wheels.
3.3) The system was quite advantageous to these folks. However, in addition to the wheelchair system, you also had to carry your laptop all the time. People came up with the idea of using Raspberry Pi to control the entire wheelchair system to reduce the bulk and cost of the MATLAB-based Eye Movement based Electronic Wheelchair System.People started utilising Raspberry Pi-based Wheelchair Systems since they are portable and have their own operating system.
3.4) The fundamental issue with the existing Raspberry Pi-based wheelchair system is latency (delay in responding).
3.5) As a result, we have created a system that minimises latency and employs efficient image processing algorithms using OpenCV.
3.6) The two algorithms, along with sight, are used to move the wheelchair.
3.7) Python is used to programme the Raspberry Pi.
When power is provided to the Raspberry Pi by power backups, such as through power banks, the same method is continuously done using shell script.

4. HARDWARE REQUIREMENT

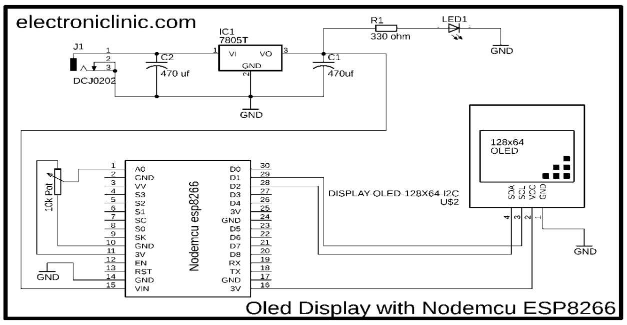
4.1) Computer with 80 HDD/Raspberry Pi and 4 GB of RAM 4.
1. Intel Core i3-1215U, 12th generation of processors (10 MB cache, 6 cores, 8 threads, up to 4.40 GHz Turbo)
2. Windows 11 Home Single Language, English as the operating system
3. Intel® UHD Graphics Video Card
4. 16-inch FHD+ 1920x1200 non-touch display with a wide viewing angle and 250 nits of brightness.
5. Memory * DDR4, 3200 MHz, 8 GB, 1 x 8 GB
6. 512 GB M.2 PCIe NVMe SSD hard drive

4.2) NODE MCU ESP8266

The microcontroller known as the ESP8266 was created by Espressif Systems. The ESP8266 is a self-contained WiFi networking system that can run standalone programmes and serves as a bridge between WiFi and current micro controllers. This module includes a built-in USB connector as well as a wide range of pin-outs. Like Arduino, NodeMCUdevkit may be easily connected to your laptop through a tiny USB cable and flashed. Additionally, it is right away breadboard friendly.
Specification:
1. • 3.3 volts.
2. Soft-AP for Wi-Fi Direct (P2P).
3. • Current usage ranges from 10uA to 170mA.
4. Flash memory attachable with a maximum of 16MB (512K normal).
5. • A TCP/IP protocol stack that is integrated.
Tensilica L106 32-bit processor.
7. • 80–160 MHz processor speed.
8. • RAM: 32K + 80K.
9. • GPIOs: 17 (multiplexed with other functions).
10. One input with a resolution of 1024 steps for analogue to digital.
11. The output power in 802.11b mode is +19.5 dBm.
12. Support for 802.11 b/g/n.
13. Maximum number of active TCP connections

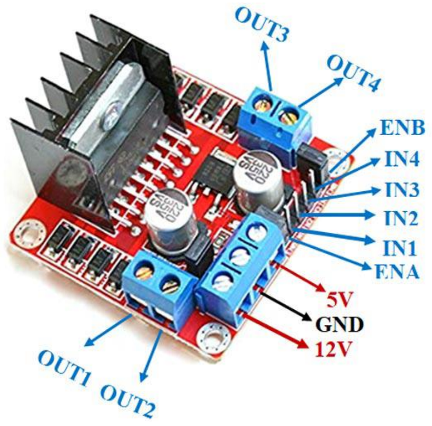


SCHEMATIC DAIGRAM

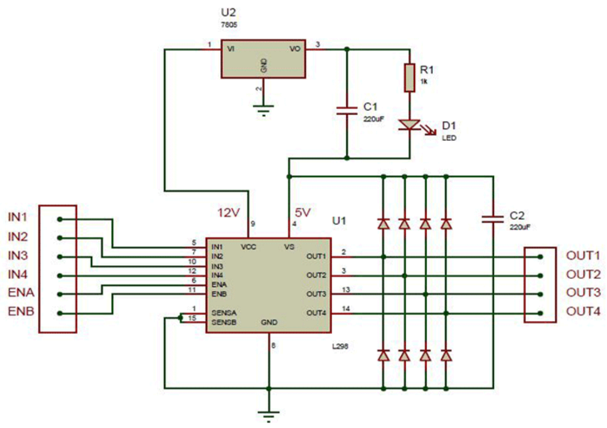


4.3) L298N MOTOR DRIVER

The widely used L298 Dual H-Bridge Motor Driver Integrated Circuit serves as the foundation for this dual bidirectional motor driver. You can simply control two motors with up to 2A each in both directions using this circuit. It works great for robotic applications and connects easily to microcontrollers with only a few control lines needed for each motor. Relays, TTL logic gates, basic manual switches, and other devices can all be interfaced with it. This board has an internal +5V regulator, power LED indicators, and protective diodes.



CIRCUIT DAIGRAM L298



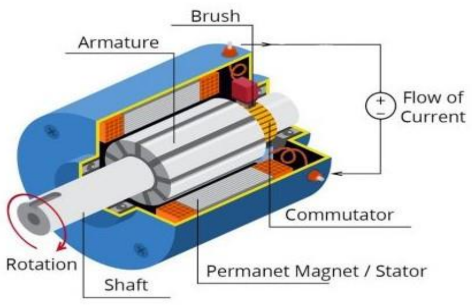
4.4) DC GEARED MOTOR

For the elevator's vertical to and fro motion, a gearless DC motor is used. It acknowledges PWM pulses from the microprocessor and maintains the wheels' speed as a result.

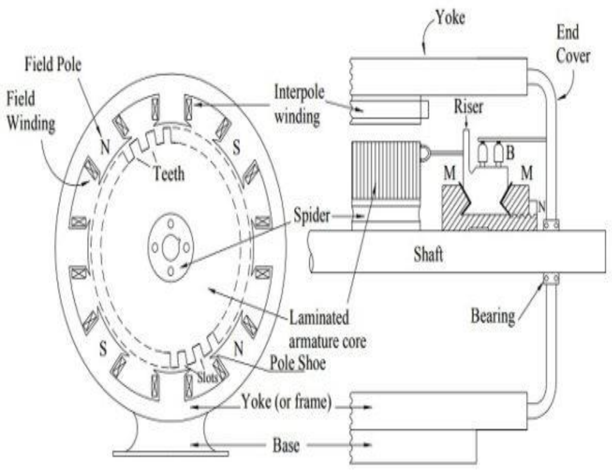
Features of a DC motor with gears
1. Shaft rotation in both directions
2. Allows for PWM pulses for speed control
3. Operates with a variety of voltages
4. A 50:1 gear ratio offers sufficient torque for smooth, easy elevator motion.
5. More dependable Rugged Design

Technical Information
1. 12V Rated Voltage
2. 3V–12V operating voltage range
3. 50 mA for current with no load
4. 500mA Current at Full Load
5. A 50:1 gear ratio
6. 100 RPM is the motor shaft speed.
7. 2.7 KgF-cm of torque.

DC MOTOR BUILD



DC MOTOR SCHEMATIC DAIGRAM



5. SOFTWARE PRESCRIPTIONS

OpenCV

OpenCV is a computer vision library that is available for free (see http://opensource.org). The library may be used with Linux, Windows, and Mac OS X and is developed in C and C++. Interfaces for Python, Ruby, Matlab, and other languages are actively being developed. OpenCV was created with a significant emphasis on real-time applications and processing performance. OpenCV is a multicore processor-capable programme that is written in C that has been optimised. You can purchase Intel's Integrated Performance Primitives (IPP) libraries [IPP], which contain low-level optimised routines in a variety of algorithmic domains, if you want further automatic optimization on Intel architectures [Intel]. If the library is installed, OpenCV will utilise the relevant IPP library at runtime. One of OpenCV's objectives is to offer an easy-to-use computer vision infrastructure that enables anyone to quickly develop moderately complex vision applications. Over 500 functions in the OpenCV library cover a wide range of vision applications, such as manufacturing product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics. Due to the close relationship between computer vision and machine learning, OpenCV also includes a complete, all-encompassing Machine Learning Library (MLL). This section of the library focuses on the identification and clustering of statistical patterns. The MLL is extremely helpful for the vision tasks at the centre of OpenCV's goals, but it is sufficiently all-purpose to be applied to any machine learning issue.

Display a picture
When you want to display an image in a window, use the cv2.imshow() function. The window adjusts itself to the size of the image. A string representing the window name is the first argument. The second point is our reputation. As many windows as you like can be made, each with a unique window name.
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
The window will appear as seen in the following screenshot (on a Fedora-Gnome computer):

Keyboard binding function cv2.waitKey(). The millisecond time is its main point. The function waits for any keyboard event for a predetermined number of milliseconds. The programme continues if any key is pressed during that period. It waits endlessly for a keystroke if 0 is passed. Additionally, it may be configured to recognise particular keystrokes, such as if key A is pushed, etc., which we shall cover shortly.
The windows we generated are all simply destroyed by calling cv2.destroyAllWindows(). Use the function cv2.destroyWindow() and supply the particular window name as an argument if you wish to delete a specific window.
Video from Camera Capture
We frequently need to use cameras to record live streams. This has a fairly straightforward interface thanks to OpenCV. Let's record a video from the camera (I'm using my laptop's built-in webcam), turn it into grayscale video, and then display it. Just one small thing to get things going.
You must create a VideoCapture object in order to record video. Either the device index or the filename of a video can be used as its parameter. Just a number to identify which camera, the device index. Typically, only one camera will be linked (as in my case). Thus, I just pass 0. (or -1). By moving on to 1 and beyond, you can choose the second camera. After that, you can take individual frames of video. Don't forget to release the capture, though, at the conclusion.
import np for Numpy
while(True), import cv2 cap = cv2.VideoCapture(0):
Frame-by-frame capture with ret, frame = cap.read ()
Gray = cv2.cvtColor(frame, cv2.COLOR BGR2GRAY) is where our activities on the frame begin.
If cv2.waitKey(1) & 0xFF == ord('q'), then display the resulting frame by using cv2.imshow('frame',gray):
break
# Release the capture after everything is finished by calling cap.release() cv2.destroyAllWindows(), and cap.read(), which returns a bool (True/False). Frame will be True if it has been correctly read. So, by looking at this return value, you may determine when the video ends.
On occasion, cap might not have started the capture. This code displays error in that scenario. The cap.isOpened method allows you to determine whether it has been initialised or not (). Okay if that is true. If not, open it with cap.open ().
Additionally, you can access some of this video's features by utilising the cap.get(propId) method, where propId is a number between 0 and 18. If it applies to that particular film, each number represents a particular attribute, and complete information can be found here: Property Identifier. With the aid of cap.set, several of these values can be changed (propId, value). The new value you desire is value.
For instance, using cap.get(3) and cap.get, I can determine the frame's width and height (4). By default, it provides me 640x480. I do however want to change it to 320x240. You only need to use ret = cap.set(3,320) and ret = cap.set (4,240).
using a file to play a video

Just replace camera index with video file name, and it will work the same as capturing from a camera. Use the proper amount of time for cv2.waitKey while showing the frame (). If it's too low, the video will play very quickly, and if it's too high, it will play slowly (Well, that is how you can display videos in slow motion). In most situations, 25 milliseconds will be adequate.

```
import np for Numpy
while (cap.isOpened()): import cv2 cap = cv2.VideoCapture('vtest.avi'):
frame = cap.read ret ()
grey = (frame, cv2.COLOR BGR2GRAY)
cv2.imshow('frame',gray)
If cv2.waitKey(1) and 0xFF are equal to ord('q'):
break \scap.release()
cv2.destroyAllWindows()
```

To identify and depict important areas of the face, facial landmarks are employed, such as:

1. Eyes
2. Eyebrows
3. Nose
4. Mouth
5. Jawline

Face alignment, head pose estimation, face switching, blink detection, and many more tasks have all been accomplished with great effectiveness using facial landmarks.

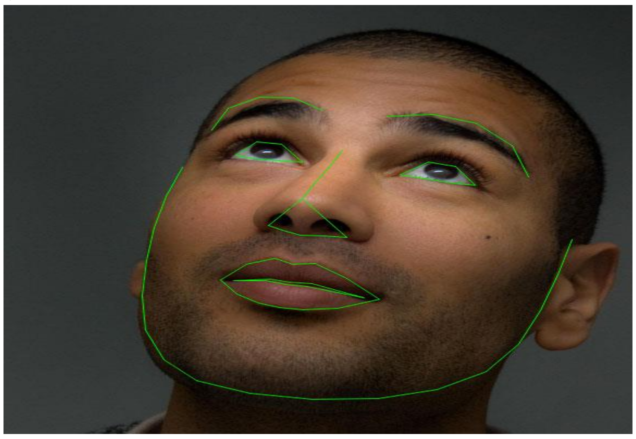


Figure 1: Key face characteristics in an image are labelled and identified using facial landmarks.

The problem of shape prediction includes the subset of recognising facial landmarks. A shape predictor aims to identify important points of interest along the shape given an input image (and typically a ROI that identifies the object of interest).

Our objective is to identify significant facial features on the face using shape prediction techniques in the setting of facial landmarks.

Therefore, identifying face landmarks requires two steps:

Locate the face in the image as the first step.

2. Locate the important facial structures on the face ROI.

One of the first steps is face detection, which can be done in several ways.

OpenCV's in-built Haar cascades could be used.

For the purpose of face detection, we might use a pre-trained HOG + Linear SVM object detector.

For face localisation, we could even employ deep learning-based methods.

The actual algorithm that was employed to find the face in the image is irrelevant in either scenario. Instead, what matters is that we discover the face bounding box—that is, the (x, y)-coordinates of the face in the picture—by some means.

We can next use Step #2, which involves identifying important facial structures in the face region, given the face region.

Different facial landmark detectors exist, however they all generally aim to identify and localise the following facial regions:

1. Mouth
2. The right brow
3. The left brow
4. Right eye
5. Left eye
6. Nose
7. Jaw

A training set of labelled facial landmarks on an image is used as the first step in this method. The particular (x, y) coordinates of the regions surrounding each facial structure are specified in the manual labelling of these images.

Priors, or more precisely, the likelihood of the separation between adjacent input pixel pairs.

An ensemble of regression trees is trained using this training data to predict the positions of face landmarks solely from the pixel intensities (i.e., no "feature extraction" is happening).

The outcome is a face landmark detector with high-quality predictions that can be used to identify facial landmarks in real-time.

The 68 (x, y)-coordinates that correspond to facial structures on the face are estimated using the pre-trained facial landmark detector found inside the dlib library.

The graphic below shows how the 68 coordinates' indexes can be seen.

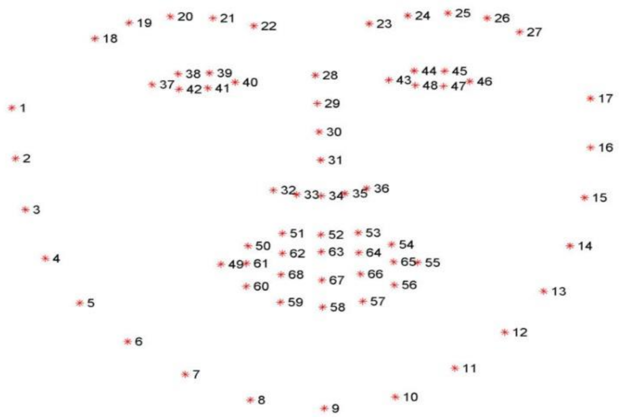


Figure 2: 68 face landmark coordinates from the iBUG 300-W dataset are visualised

The dlib facial landmark predictor was trained on the 68-point iBUG 300-W dataset, which contains these annotations. Six (x, y)-coordinates are used to represent each eye, beginning at the left corner (as if you were staring at the subject) and moving clockwise around the remainder of the area:

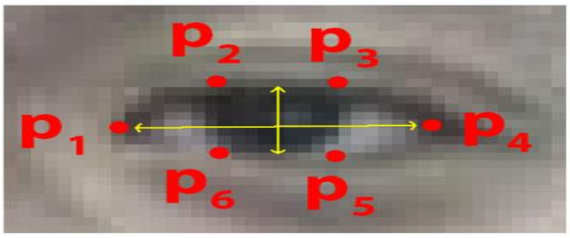


Figure 3: six facial landmarks that are connected to the eye.

One important lesson from this figure is that there is a correlation between the breadth and height of these coordinates. We can then create an equation that depicts this relation termed the eye aspect ratio (EAR) based on the research done by Soukupová and ech in their 2016 paper, Real-Time Eye Blink Detection using Facial Landmarks:

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

the formula for the eye aspect ratio.
where the points p1,..., and p6 are 2D facial landmarks.
Since there is only one set of horizontal points but two sets of vertical points, the denominator of this equation is weighted suitably while the numerator computes the distance between vertical eye landmarks and the denominator computes the distance between horizontal eye landmarks.
Horizontal and vertical landmarks
# (X, Y)-coordinates of vertical eye landmarks
Euclidean distance A = (eye[1], eye[5])
Euclidean distance = B (eye[2], eye[4])
Calculate the euclidean distance between the (x, y)-coordinates of the horizontal eye landmarks.
Euclidean distance = C (eye[0], eye[3])
The user looks left if the horizontal ratio is greater than a predetermined threshold number; else, the user looks right. If neither of the threshold values is met, the user is viewing the centre. If the EAR value falls below the predetermined level, the user will blink.

IMAGERY FOUNDATIONS

The Building Blocks of Images are Pixels.
The basic building components of an image are called pixels. Each image is made up of a group of pixels. The pixel is the finest granularity there is. A pixel is typically thought of as the "colour" or "intensity" of light that appears in a certain location in our image. Each square in an image, which can be visualised as a grid, holds a single pixel. Take a look at Figure 3.1 as an illustration. The image in Figure 3.1 is 1000 pixels wide and 750 pixels tall, or at a resolution of 1000 750. An picture can be thought of as a (multidimensional) matrix. In this instance, our matrix comprises 750 rows and 1000 columns (the width) (the height). In total, our image contains 1000 750 = 750000 pixels.
Two common approaches to represent pixels are:
1. Single-channel and grayscale
2. Color
Each pixel in a grayscale image is a scalar value between 0 and 255, where 0 represents "black" and 255 represents "white." Grayscale values range from 0 to 255, with values closer to 0 being darker and closer to 255 being lighter. Figure 3.2's grayscale

gradient image shows pixels getting lighter as they move from left to right, with darker pixels on the left. However, RGB is typically used to represent colour pixels.

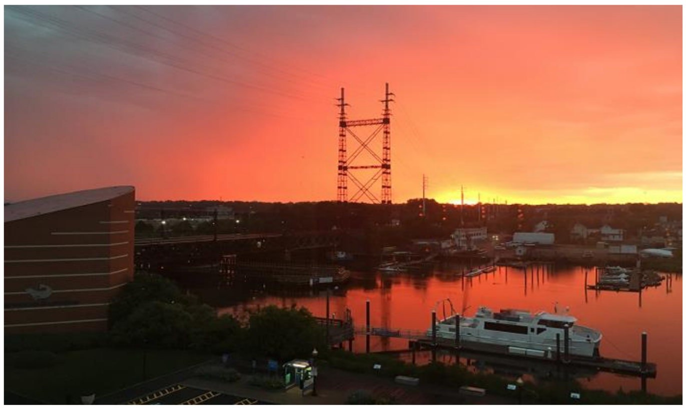


Figure 3.1: This image has a total of 750000 pixels, or 1000 pixels wide by 750 pixels height.

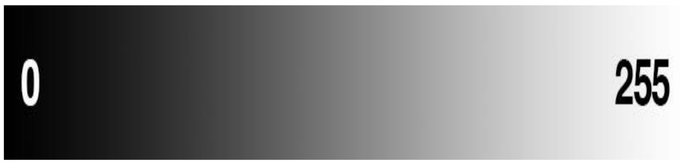


Figure 3.2: A gradient image showing black (0) to white (255) pixel values (255).

In the RGB colour space, pixels are represented by a list of three values, one for each of the three colour components—Red, Green, and Blue—instead of a single scalar value as in a grayscale or single channel image. The only thing we have to do to define a colour in the RGB colour model is to specify how much Red, Green, and Blue are present in a single pixel.
There are 256 possible "shades" for the Red, Green, and Blue channels, with values set in the range [0;255], where 0 denotes no representation and 255 shows full representation. We typically use 8-bit unsigned integers to describe the intensity because the pixel value just needs to fall inside the range [0;255]. We commonly preprocess our images by doing mean subtraction or scaling, which requires us to transform the image to a floating point data type, as we'll see after we build our first neural network. Remember this because before we apply learning algorithms to the images directly, the data types used by libraries importing images from disc (such as OpenCV) may frequently need to be transformed. Our three Red, Green, and Blue values may be combined to produce an RGB tuple, which looks like this: (red, green, blue). A specific colour is represented by this tuple in the RGB colour space. An additive colour system is one like the RGB colour space, where adding more of each colour makes the pixel brighter and closer to white. Figure 3.3 provides an illustration of the RGB colour space (left). As you can see, when red and green are combined, yellow results. Pink is produced by combining red and blue. And when we combine the three hues of red, green, and blue, we obtain white.

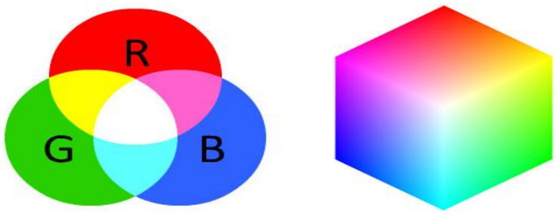


Figure 3.3: RGB colour space is additive, as seen on the left. You grow closer to white the more red, green, and blue you combine. Right: An RGB cube.

Consider the colour "white" once more to help make this example more clear. To represent this colour, we would entirely fill the red, green, and blue buckets as follows: (255, 255, 255). Then, as black is the absence of colour, we would empty each bucket to generate the colour black (0, 0, 0).
We would entirely fill the red bucket (and only the red bucket) to get a pure red colour: (255, 0, 0).
Another frequent way to represent the RGB colour space is as a cube (Figure 3.3, right) Due to the fact that an RGB colour is defined as a 3-valued tuple with each value being within the range [0;255], we may think of the RGB colour space as a cube with 256 256 256 = 16;777;216 possible colours Green, Blue, and each bucket are filled with water.
For illustration, let's think about how "much" red, green, and blue would be required to produce a single colour (Figure 3.4, top). Here, we set R=252, G=198, and B=188 to get a skin-tone-like colour tone (perhaps helpful when developing an application to determine how much skin or flesh is present in an image). As we can see, the bucket is almost completely filled, and the Red component is strongly represented.
Nearly equal numbers of Green and Blue are present. By adding these hues together, we can create a colour tone resembling that of Caucasian flesh.

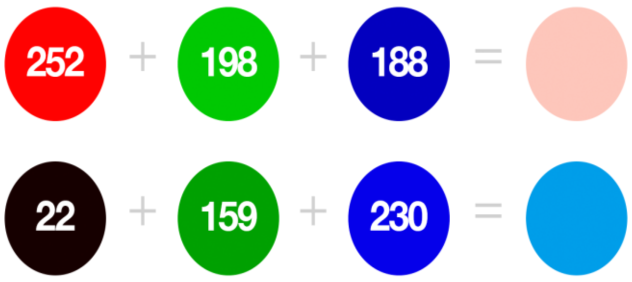


Figure 3.4: Top: A possible application for a skin detection programme involves combining different Red, Green, and Blue colour components to generate a "caucasian flesh tone."

Picture Coordinate System A grid of pixels is used to depict an image, as was indicated in Figure 3.1 previously in this chapter. Consider our grid to be like a sheet of graph paper to help illustrate this notion. The upper-left corner of the image is where the origin point (0;0) falls on this graph paper. The x and y values rise as we walk down and to the right. This "graph paper" representation is seen visually in Figure 3.6. On a piece of our graph paper, we can see the letter "I" in this case. We can observe that there are 64 total pixels in this 8 8 grid. It's vital to remember that we are counting backwards from one to zero. Python is a zero-indexed language, thus we always start at 0 when counting.

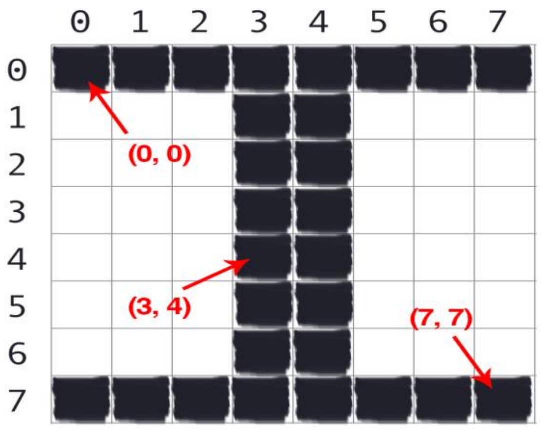


Figure 3.6 written in capital letters on a sheet of graph paper. Keeping in mind that Python is zero-indexed, we access pixels by their (x;y)- Coordinates by moving x columns to the right and y rows down.
Consider the pixel 4 columns to the right and 5 rows down is indexed by the point (3;4) as an illustration of zero-indexing, again keeping in mind that we are counting from zero rather than one. Image arrays in NumPy

Figure 3.7: displaying an example.png image on our screen after loading it from disc

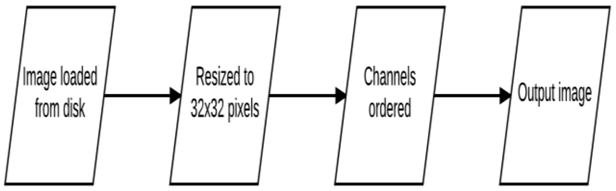
PUBLIC CV WORKING
Multidimensional NumPy arrays with shape are used by image processing libraries like OpenCV and Scikit-Image to represent RGB images (height, width, depth). When using image processing tools for the first time, readers frequently find this representation confusing because the height appears before the width, which is how we typically think about images.
Matrix notation provides the solution.
We always express the dimensions of a matrix as rows x columns. An image's height is determined by the number of rows, but its width is determined by the number of columns. The depth will continue to be the same. In view of how a matrix is built and annotated, it may seem a little weird to see the .shape of a NumPy array represented as (height, width, depth), but this representation actually makes sense.

Ordering for RGB and BGR
Remember that OpenCV stores RGB channels in the opposite order. While Red, Green, and Blue are the colours we typically conceive of, OpenCV actually maintains the pixel values in Blue, Green, and Red order.

Why does OpenCV act in this way? The simple explanation is historical factors. Because BGR ordering was common among camera makers and other software developers at the time, early OpenCV library developers decided to utilise it. Simply put, this BGR ordering was decided upon historically and is something we must now live with. It's a little caution, but it's one that you should remember when using OpenCV.

ASPECT RATIOS AND SCALING

The process of increasing or decreasing an image's width and height is known as scaling, or simply resizing. It's crucial to keep the aspect ratio in mind while scaling an image.



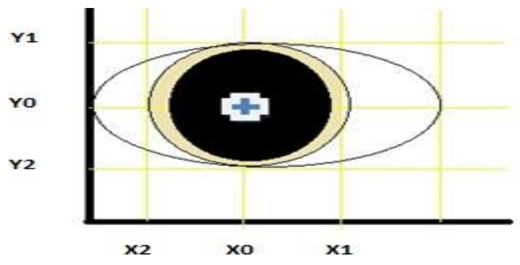
Pipeline for image pre-processing that loads a picture from disc, resizes it to 32x32 pixels, arranges the channel dimensions, and outputs the image.

PYTHON

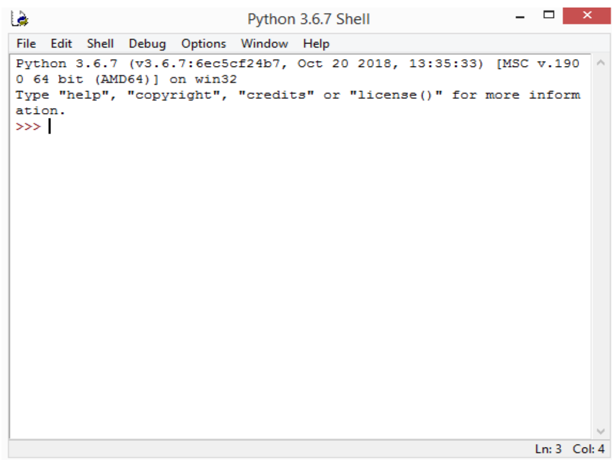
Python is a high-level, all-purpose programming language that is interpreted. Python's design philosophy places a strong emphasis on code readability through the use of noticeable indentation. Its language constructs and object-oriented methodology are intended to aid programmers in creating clean, comprehensible code for both little and big projects.
Python uses garbage collection and has dynamic typing. It supports a variety of programming paradigms, including procedural, object oriented, and structured programming (especially). Due to its extensive standard library, Python is frequently referred to as a "batteries included" language.

IDLE PYTHON

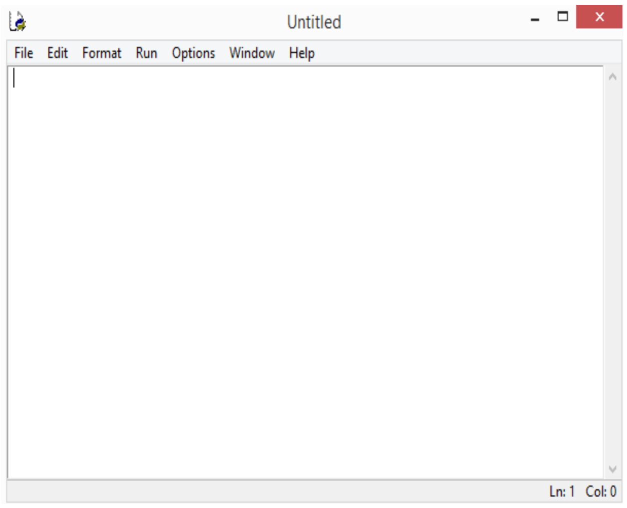
An Integrated Development and Learning Environment, also known as IDLE or even IDE, is included with every Python



installation. These are a group of programmes that make it easier to write code. Although there are several IDEs available, Python IDLE is fairly basic, making it the ideal tool for a beginner programmer.
Python installations for Windows and Mac include Python IDLE. Python IDLE should be easy to find and download if you run Linux thanks to your package manager. After installation, you can use Python IDLE as a file editor or an interactive interpreter. One that is Interactive. The interactive interpreter, sometimes known as a shell, is the finest place to explore with Python code. Essentially, the shell is a Read-Eval-Print Loop (REPL). It reads a statement in Python, evaluates the outcome, then prints the outcome on the screen. The following statement is read after a looping backwards.
Small code snippet experiments can be done in the Python shell. Through your computer's terminal or command line application, you can access it. Python IDLE, which launches a Python shell as soon as it is opened, helps streamline your productivity.



```
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.190
0 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inform
ation.
>>> |
```



Editor for files

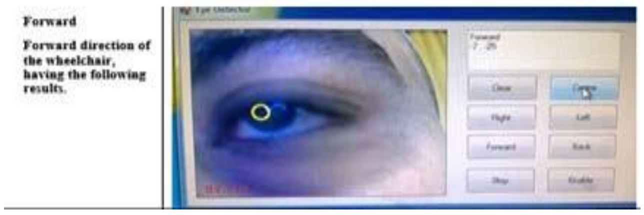
Text files must be able to be edited and saved by every coder. Lines of Python code are stored in files with the.py extension as programmes. You may easily generate and edit these files using Python IDLE. Additionally, Python IDLE has a number of helpful features that you'll find in expert IDEs, such as basic syntax highlighting, code completion, and auto-indentation. Professional IDEs have a steep learning curve and are more capable pieces of software. Python IDLE is a fantastic substitute if you're just starting out with Python programming.

6.EXPERIMENTATION AND OUTCOMES

Because the wheelchair's programming is done on an open-source platform and its parts are widely accessible worldwide. To make the patient with the camera mounted on a stand more comfortable, the wheelchair's design has been modified. As soon as the patient is seated in the wheelchair, the camera begins to capture a picture of the patient's eyes. This image is then transferred to a laptop for further processing.

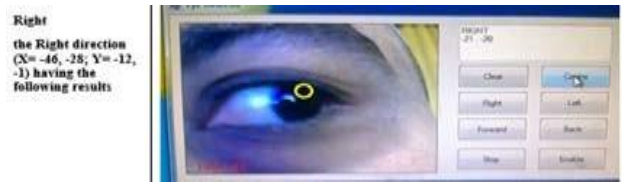
The wheelchair's bottom has a circuit base frame built so that all of its parts, including the battery, may be mounted and kept secure. Common automotive batteries, which are easily obtainable on the market, are the batteries that are used. The wheelchair's batteries are kept at the bottom so that the charging terminals may be plugged right in. Patients with quadriplegia disorders, which cause partial or complete limb and torso paralysis, have access to an eye control feature. The wheelchair mechanism used an Arduino Mega 2560 to give the command to the motor driver after receiving the date from the eye location. The wheelchairs move in response to eye movements. To further prevent any collisions or accidents, an ultrasonic sensor will be employed for obstacle detection. Figure presents the findings.

A) In a forward motion


Forward

Forward direction of the wheelchair, having the following results.

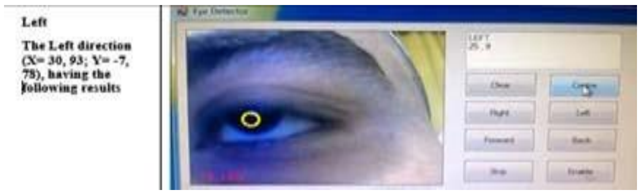
B) The Correct Direction



Right

the Right direction (X= -46, -28; Y= -12, -1) having the following results

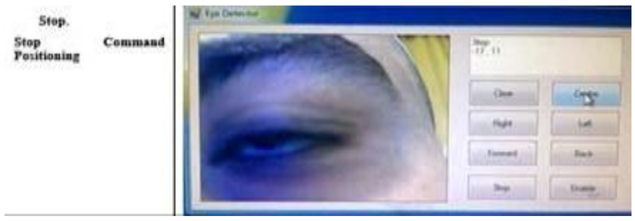
C) Move to the Left



Left

The Left direction (X= 30, 93; Y= -7, 78), having the following results

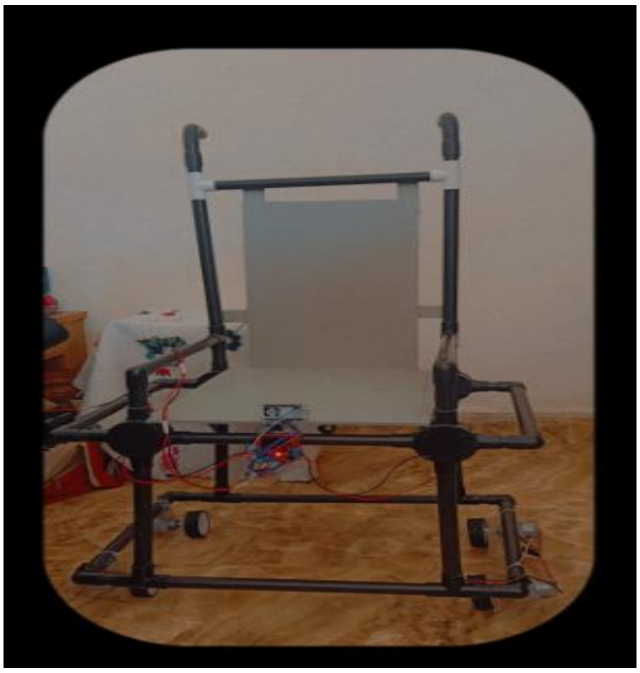
D) Stop



Stop.

Stop Positioning Command

SEAT FRAME



7. CONCLUSION AND NEXT WORK

This study was created to create a portable and affordable wheelchair that is controlled by eye movements. It is rendered in this paper as an eye-controlled wheelchair. Because it can be difficult for impaired persons to control their movements, the eye control approach is particularly practical for people with disabilities because it doesn't require the use of any other body parts. The functionality of the wheelchair may be impacted by the fact that this prototype even functions when the photograph is taken in a dark environment. Additional elements will be included to make it more engaging for persons with disabilities. Its performance will significantly improve with the addition of a solar system and a health monitoring system. In this study, we use Visual Studio to detect the eye using a camera, but in the future, the eye may be detected using an IR sensor, for example.

REFERENCES:

1. Venugopal, Divya, Joseph Amudha, and C. Jyotsna. "Developing anapplication using eye tracker." 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, 2016.

2. Pavani, M. Lakshmi, AV BhanuPrakash, MS ShwethaKoushik, J. Amudha, and C. Jyotsna. "Navigation Through Eye-Tracking for Human–Computer Interface." In Information and Communication Technology for Intelligent Systems, pp. 575-586. Springer, Singapore, 2019.
3. Huang, Jeffrey, and Harry Wechsler. "Eye detection using optimal wavelet packets and radial basis functions (rbfs)." International Journal of Pattern Recognition and Artificial Intelligence 13.07 : 1009-1025,1999
4. Dutta, Palash, and DebotoshBhattacharjee. "Face detection using generic eye template matching." 2014 2nd International Conference on Business and Information Management (ICBIM). IEEE, 2014.
5. Akashi, T., Wakasa, Y., Tanaka, K., Karungaru, S., &Fukumi, M, "Using genetic algorithm for eye detection and tracking in video sequence". Journal of Systemics, Cybernetics and Informatics, 5(2), 72-78, 2007
6. Nishimura, Toshiya, et al. "Eye interface for physically impaired people by genetic eye tracking." SICE Annual Conference 2007. IEEE, 2007.
7. Al-Mamun, Hawlader Abdullah, et al. "Eye detection in facial image by genetic algorithm driven deformable template matching." International Journal of Computer Science and Network Security 9.8 : 287-294, 2009
8. Perez, Claudio A., et al. "Face and iris localization using templates designed by particle swarm optimization." Pattern recognition letters 31.9 :857-868, 2010
9. Al-Rahayfeh, Amer, and MiadFaezipour. "Eye tracking and head movement detection: A state-of-art survey." IEEE journal of translational engineering in health and medicine 1 : 2100212-2100212, 2013
10. Santos, Rafael, et al. "Eye gaze as a human-computer interface." Procedia Technology 17 : 376-383, 2014
11. Orman, Zeynep, AbdulkadirBattal, andErdemKemer. "A study on face, eye detection and gaze estimation." IJCSES2.3 : 29-46, 2011
12. Akashi, Takuya, et al. "Using genetic algorithm for eye detection and tracking in video sequence." Journal of Systemics, Cybernetics andInformatics 5.2 : 72-78, 2007
13. Nandakumar, Hitha, and J. Amudha. "A comparative analysis of a neuralbasedremote eye gaze tracker." 2014 International Conference on Embedded Systems (ICES). IEEE, 2014.
14. Torricelli, Diego, et al. "A neural-based remote eye gaze tracker under natural head motion." Computer methods and programs in biomedicine 92.1 : 66-78, 2008