VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi: 590 018

A Mini Project Report

On

"IRIS SEGMENTATION"

Submitted in partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering in Computer Science and Engineering

Submitted by

ABHISHEK PANDEY (1VE21CS004)

Under the Guidance of

MRS. RANJANA THAKURIA
Assistant Professor
Dept. of CSE,
SVCE, Bengaluru.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESHWARA COLLEGE OF ENGINEERING

Affiliated to VTU Belgaum & Approved by AICTE New Delhi) an ISO 9001:2008 Certified, Kempegowda International Airport Road, Vidyanagar, Bengaluru, Karnataka, India-562157

2023– 2024

SRI VENKATESHWARA COLLEGE OF ENGINEERING
Vidyanagar, Bengaluru, Karnataka, India-562157

Department of Computer Science & Engineering

SRI VENKATESHWARA GROUP OF INSTITUTIONS
★ SVCE ★
Estd. : 2001

Accredited by NAAC & NBA*

SVCE BENGALURU

SRI VENKATESHWARA COLLEGE OF ENGINEERING

— Affiliated to VTU, Approved by AICTE, Recognised by UGC u/s 2(f) & 12(B)—

CERTIFICATE

This is to certify that Mini Project entitled **"IRIS SEGMENTATION"** is submitted by **ABHISHEK PANDEY** bearing **USN 1VE21CS004** respectively on partial fulfillment of sixth semester, Bachelor of Engineering in Computer Science and Engineering, Visvesvaraya Technological University for the academic year 2023-2024.

………………………………	………………………………..
Signature of Guide	**Signature of the HOD**

Name of the Examiner	**Signature with date**
1.	
2.	

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without complementing those who made it possible, whose guidance and encouragement made our efforts successful.

My sincere thanks to highly esteemed institution SRI VENKATESHWARA COLLEGE OF ENGINEERING for grooming up me in to be software engineer.

I express our sincere gratitude to **Dr. Nageswara Guptha M**, Principal, SVCE, Bengaluru for providing the required facility.

I am extremely thankful to **Dr. Hema M S**, HOD of CSE, SVCE for providing support and encouragement.

I am grateful to **Mrs. Ranjana Thakuria**, Asst. Professor, Dept. of CSE, SVCE who helped me to complete this project successfully by providing guidance, encouragement and valuable suggestion during entire period of the project. I thank all my computer science staff and others who helped directly or indirectly to meet my project work with grand success.

Finally, I am grateful to my parents and friends for their invaluable support guidance and encouragement.

ABHISHEK PANDEY [1VE21CS004]

ABSTRACT

This project focuses on developing a computer program to automatically identify and isolate the iris region in an eye image. The iris, the colored part of the eye, contains unique patterns that can be used for identification. By accurately separating the iris from the rest of the eye, we can analyze these patterns for various applications like security systems or medical diagnosis. This project explores different techniques to achieve precise iris segmentation and evaluates their effectiveness.

Contents

CHAPTER-1

INTRODUCTION

1.1 BACKGROUND

The iris is the colorful part of your eye. It's like a unique pattern, different for everyone. This makes it a great way to identify people, similar to using fingerprints.

To use the iris for identification, we need a clear picture of just the iris. This means cutting out the iris from the rest of the eye image. That's where iris segmentation comes in. It's like finding and cutting out a shape from a puzzle.

But it's not easy. Eyes come in different sizes and shapes, and there's often stuff like eyelashes or glare that can hide the iris. So, finding the exact edges of the iris is tricky.

That's why iris segmentation is important**.** It's the first step to using iris recognition for things like unlocking phones, security systems, or medical checks.

1.2 OBJECTIVE

- To Locate the exact boundaries of the colored part of the eye (iris) in a picture.
- To Cut out the iris from the rest of the eye image, like cutting out a shape from paper.
- To Make sure the iris image is clear and without any extra parts like eyelashes or eyelids.
- To Get the iris ready for further processing, like measuring or comparing it to other irises.
- To accurately identify and isolate the iris from an eye image so it can be used for recognition or other purposes.

CHAPTER-2

LITERATURE SURVEY

2.1 WHAT IS IRIS SEGMENTATION

Iris segmentation is the process of identifying and isolating the iris region in an eye image. It's like cutting out the colored part of the eye (the iris) from the rest of the picture.

2.2 WHY IT IS IMPORTANT

Accurate iris segmentation is crucial for iris recognition systems, where the unique patterns in the iris are used for identification. By isolating the iris, we can focus on analyzing its specific features for biometric purposes.

In simpler terms: It's like finding and cutting out a specific puzzle piece (the iris) from a larger picture (the eye) to study its unique pattern.

2.3 EXISTING SYSTEM

The iris segmentation is used for the general applications and for the research areas. Here we discussed some general areas where iris segmentation is applied and some research studies.

Here are the some of the applications that are used widely:

Biometric Security: This is the most common application used in high-security areas like airports, government building and data centers.

Financial institutions: Banks and other financial institutions use iris recognition for secure authentication.

Healthcare: Iris analysis can be used for medical diagnosis, such as detecting certain eye diseases.

Most of the application are using the edge detection, intensity-based segmentation, color-based segmentation, and some of the advanced applications are using deep learning techniques like convolutional neural networks (CNNs), attention mechanisms, and the last hybrid approaches.

CHAPTER – 3

PROPOSED SYSTEM

3.1 REQUIRENMENT SPECIFICATIONS

3.1.1 SOFTWARE REQUIREMENTS

- ✓ Operating System: Window 10 or above
- ✓ Operating System: MacOS
- ✓ Google Chrome/Mozilla Firefox/Microsoft Edge
- ✓ Visual Studio Code or any python man editor

3.1.2 HARDWARE REQUIREMENTS

- ✓ Computer with 1.1 GHz or faster processor
- ✓ Minimum 2GB of RAM or more
- ✓ 2.5 of available hard-disk space
- ✓ 5400 RPM hard drive
- ✓ Higher-resolution display

3.2 INSTALLATION

You need python, install on your machine, another requirement is OpenCV and NumPy, but they come packed with Mediapipe, as a requirement when you install through PIP (python package manager) so no need to do it manually.

In case you have already installed the Mediapipe and its less than (0.8.9.1) version then upgrade to the latest version

pip install --upgrade mediapipe

Mediapipe provides, 478 landmarks of the face, you can find more details about **Face mesh**, here we gonna focus on the IRIS landmarks only since we are going the store all the landmarks in the NumPy array, so you can access them, bypass the list of indices. here are lists of IRIS Landmarks extracted using Face Mesh point's map

LEFT_IRIS = [474,475, 476, 477]

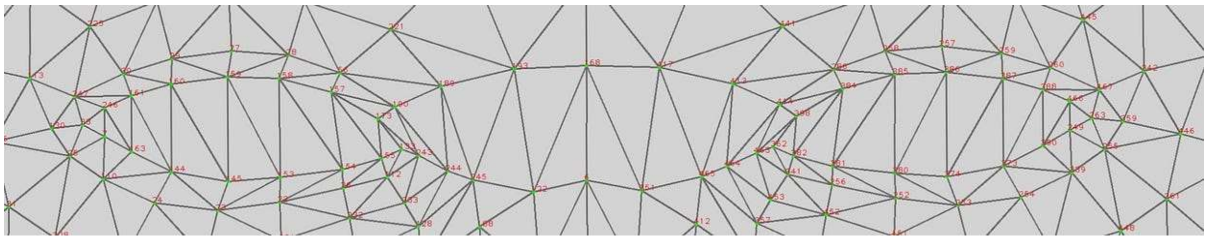
RIGHT_IRIS = [469, 470, 471, 472]

For Eyes (indices)

# Left eye indices list

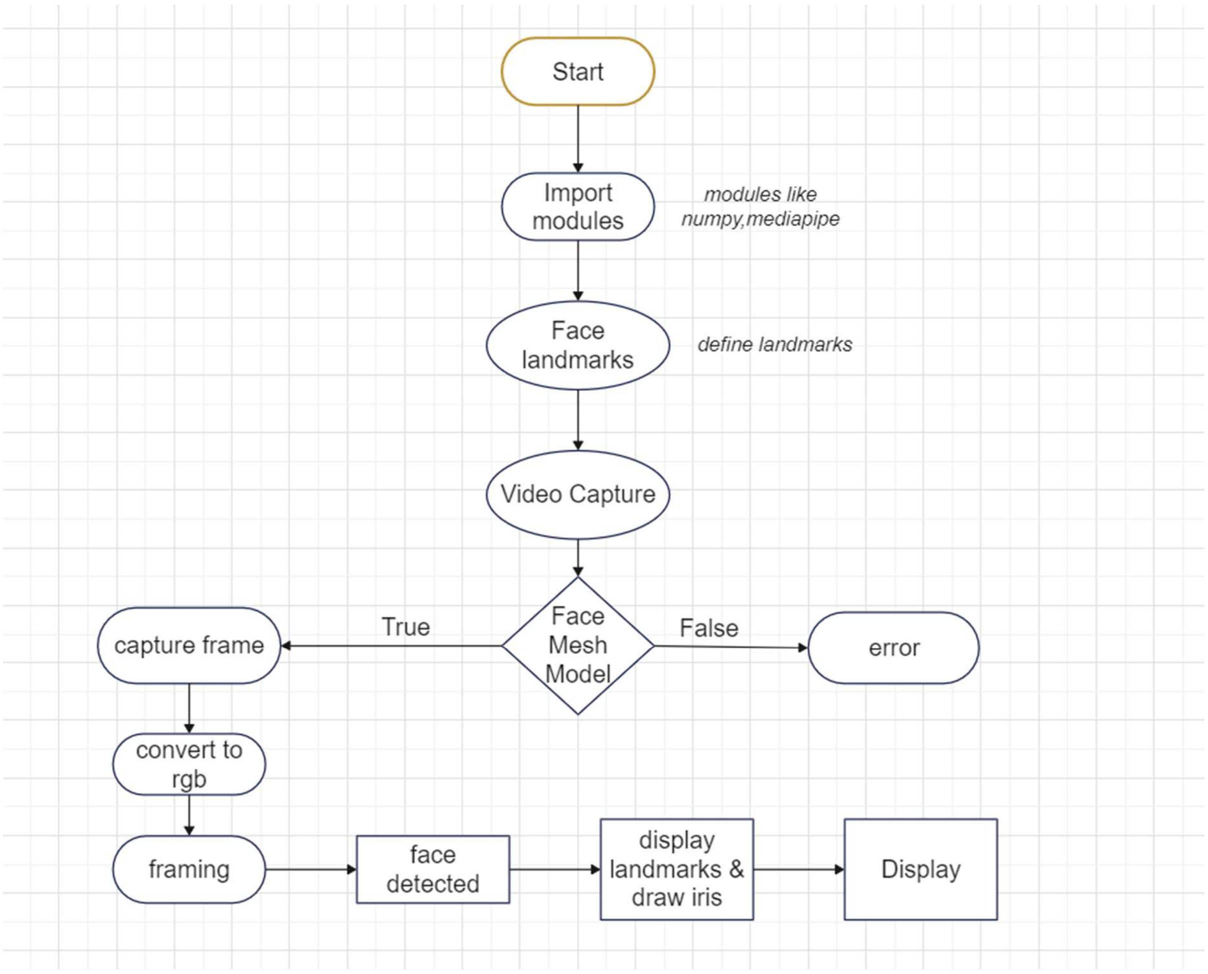
LEFT_EYE =[362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385,384, 398]# Right eye indices list

RIGHT_EYE=[33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161 , 246]



Eyes Landmarks

3.3 FLOW CHART



3.4 IMPLEMENTATION

3.4.1 IMPORTING MODULES

```
import mediapipe as mp
import cv2 as cv
import numpy as npmp_face_mesh = mp.solution.face_mesh
```

3.4.2 MODE CONFIGURATION

max_num_faces: number of faces detected
refine_landmarks: refine the landmarks for Eyes, Lips and add the additional landmarks for Irises of Eye, that are not available in previous models.
min_detection_confidence: (0.0, 1) minimum detection confidence for face detection model.
min_tracking_confidence:(0.0, 1) minimum confidence for landmarks tracking, for landmarks tracker model.
loading the Face Mesh model.

```
with mp_face_mesh.FaceMesh(
        max_num_faces=1,
        refine_landmarks=True,
        min_detection_confidence=0.6,
        min_tracking_confidence=0.6
) as face_mesh:
```

because we going the run this in real time I am going to call the image a frame, which will make sense, here, first need to flip the camera frame to mirror image, by using a function from OpenCV, since Mediapipe needs RGB colour format but OpenCV uses BGR neet to change the colour, here, *cvtColor* function.

```
Frame = cv.flip(frame, 1)
rgb_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
```

when the RGB frame is processed by the *face mesh model*, it going to return, 478 landmarks each detected face, with every landmark having x, y and z values, each having a value is between **0 to 1**, other words normalized values, then we need to multiple then with the corresponding scaling to get pixels coordinates in the frame,
for the ***X***, scaling is width, ***Y*** is the height of the image, for ***Z*** it's the same as x, width

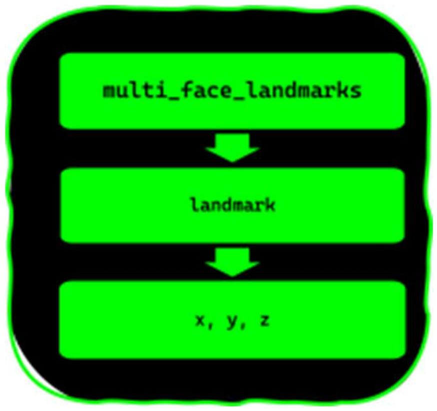
```
results = face_mesh.process(rgb_frame)
#getting width and height or frame
img_h, img_w = frame.shape[:2]
```

3.4.3 ITERATING THROUGH LANDMARKS

When we process the RGB frame we will get, each detected face, its landmarks, so we access the landmarks from the results variable we have to store them, it will be like ***results.multi_face_landmarks*** here we have all faces landmark, you can loop through them since I have detected for as ingle face, so I am going to provide index here ***results.multi_face_landmarks[0],*** which is formatted like this

```
landmark {
  x: 0.6233813166618347
  y: 0.7154796719551086
  z: -0.0638529509305954
}
```

but you still have normalised values, so you need multiply each value with proper scaling and you will get pixel coordinate, ***[x*img_w, y*img_h, z*img_w]***, but here, we just need x, and y, only, and I am going to use NumPy's multiply function to achieve that, don't forget, to convert them to integers, since OpenCV accept in as pixel coordinate as int. here is a simple one-liner, which does the job for us, end the end I have stored all the landmarks in the NumPy array (***mesh_points***) so it is easier to access bypass the list of indices *mesh_points=np.array([np.multiply([p.x, p.y], [img_w, img_h]).astype(int) for p in results.multi_face_landmarks[0].landmark])*



now we can draw the **irises** the landmarks using the OpenCV function, polyline, we already have the list of indices of irises, using them to get iris coordinate,

cv.polylines(frame, [mesh_points[LEFT_IRIS]], True, (0,255,0), 1, cv.LINE_AA)cv.polylines(frame, [mesh_points[RIGHT_IRIS]], True, (0,255,0), 1, cv.LINE_AA)

but we can turn these square shapes into circles since their function OpenCV provides enclosing circles based on provided points. named ***minEnclosingCircle*** which return, the centre (x,y) and radius of circles, ⚠ return values are floating-point, we have to turn them to int.

(l_cx, l_cy), l_radius = cv.minEnclosingCircle(mesh_points[LEFT_IRIS])

(r_cx, r_cy), r_radius = cv.minEnclosingCircle(mesh_points[RIGHT_IRIS])

# turn center points into np array

center_left = np.array([l_cx, l_cy], dtype=np.int32)

center_right = np.array([r_cx, r_cy], dtype=np.int32) finally draw the circle based on returns values from the ***minEnclosingCircle*** function, through ***circle*** function which draws circle image based on centre(x,y) and radius

cv.circle(frame, center_left, int(l_radius), (255,0,255), 2, cv.LINE_AA)
cv.circle(frame, center_right, int(r_radius), (255,0,255), 2, cv.LINE_AA)

finally getting the segmentation mask, which is quite simple, just you to create an empty mask(image) using NumPy's zeroes function, having the same dimension as a frame, and you can draw a white circle on the mask, you have the segmentation mask.
creating a mask, using image dimension width and height of the frame.

Mask = np.zeros((img_h, img_w), dtype=np.uint8)

drawing white circles on mask

cv.circle(mask, center_left, int(l_radius), (255,255,255), -1, cv.LINE_AA)cv.circle(mask, center_right, int(r_radius), (255,255,255), -1, cv.LINE_AA)

CHAPTER - 4

4. SAMPLE CODES

4.1 MAIN.PY

```
from operator import rshift
import cv2 as cv
import numpy as np
import mediapipe as mp
mp_face_mesh = mp.solutions.face_mesh
LEFT_EYE =[ 362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385,384, 398
]
# right eyes indices
RIGHT_EYE=[ 33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161 , 246 ]
LEFT_IRIS = [474,475, 476, 477]
RIGHT_IRIS = [469, 470, 471, 472]
cap = cv.VideoCapture(0)
with mp_face_mesh.FaceMesh(
    max_num_faces=1,
    refine_landmarks=True,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5
) as face_mesh:
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frame = cv.flip(frame, 1)
        rgb_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
        img_h, img_w = frame.shape[:2]
        results = face_mesh.process(rgb_frame)
        if results.multi_face_landmarks:
            print(results.multi_face_landmarks[0].landmark)
            mesh_points=np.array([np.multiply([p.x, p.y], [img_w, img_h]).astype(int) for p in
results.multi_face_landmarks[0].landmark])
            print(mesh_points.shape)
            cv.polylines(frame, [mesh_points[LEFT_IRIS]], True, (0,255,0), 1, cv.LINE_AA)
            cv.polylines(frame, [mesh_points[RIGHT_IRIS]], True, (0,255,0), 1, cv.LINE_AA)
            # (l_cx, l_cy), l_radius = cv.minEnclosingCircle(mesh_points[LEFT_IRIS])
            # (r_cx, r_cy), r_radius = cv.minEnclosingCircle(mesh_points[RIGHT_IRIS])
            # center_left = np.array([l_cx, l_cy], dtype=np.int32)
            # center_right = np.array([r_cx, r_cy], dtype=np.int32)
            # cv.circle(frame, center_left, int(l_radius), (255,0,255), 1, cv.LINE_AA)
            # cv.circle(frame, center_right, int(r_radius), (255,0,255), 1, cv.LINE_AA)

        cv.imshow('img', frame)
        key = cv.waitKey(1)
        if key ==ord('q'):
            break
cap.release()
cv.destroyAllWindows()
```

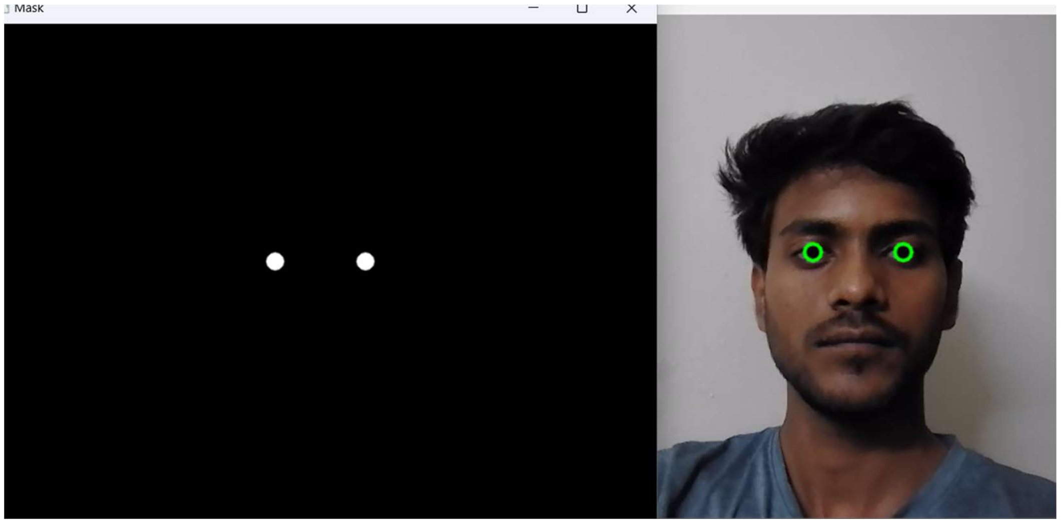
4.2 SEGMENTATION_MASK.PY

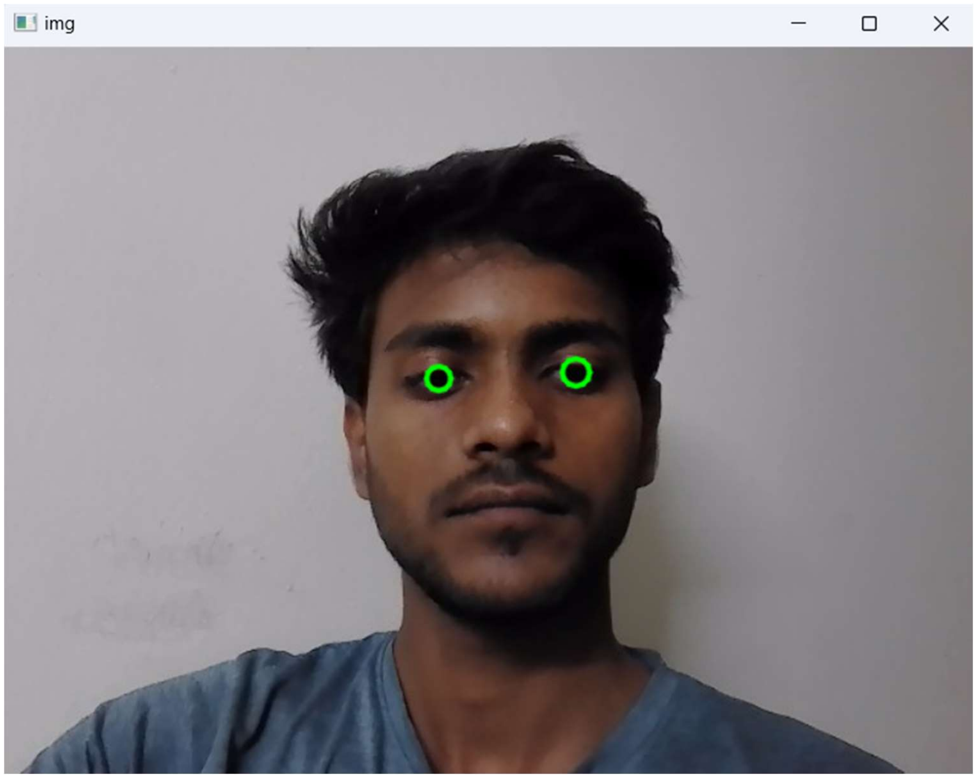
```
import cv2 as cv
import numpy as np
import mediapipe as mp
mp_face_mesh = mp.solutions.face_mesh

# left eyes indices
LEFT_EYE =[ 362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385,384, 398
]
# right eyes indices
RIGHT_EYE=[ 33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161 , 246 ]

# irises Indices list
LEFT_IRIS = [474,475, 476, 477]
RIGHT_IRIS = [469, 470, 471, 472]

cap = cv.VideoCapture(0)

with mp_face_mesh.FaceMesh(
    max_num_faces=1,
    refine_landmarks=True,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5
) as face_mesh:

    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frame = cv.flip(frame, 1)

        rgb_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
        img_h, img_w = frame.shape[:2]
        results = face_mesh.process(rgb_frame)
        mask = np.zeros((img_h, img_w), dtype=np.uint8)

        if results.multi_face_landmarks:
            # print((results.multi_face_landmarks[0]))

            # [print(p.x, p.y, p.z ) for p in results.multi_face_landmarks[0].landmark]

            mesh_points=np.array([np.multiply([p.x, p.y], [img_w, img_h]).astype(int)
            for p in results.multi_face_landmarks[0].landmark])

            # cv.polylines(frame, [mesh_points[LEFT_IRIS]], True, (0,255,0), 1, cv.LINE_AA)
            # cv.polylines(frame, [mesh_points[RIGHT_IRIS]], True, (0,255,0), 1, cv.LINE_AA)

            (l_cx, l_cy), l_radius = cv.minEnclosingCircle(mesh_points[LEFT_IRIS])
            (r_cx, r_cy), r_radius = cv.minEnclosingCircle(mesh_points[RIGHT_IRIS])
            center_left = np.array([l_cx, l_cy], dtype=np.int32)
            center_right = np.array([r_cx, r_cy], dtype=np.int32)
            cv.circle(frame, center_left, int(l_radius), (0,255,0), 2, cv.LINE_AA)
            cv.circle(frame, center_right, int(r_radius), (0,255,0), 2, cv.LINE_AA)

            # cv.circle(frame, center_left, 1, (0,255,0), -1, cv.LINE_AA)
            # cv.circle(frame, center_right, 1, (0,255,0), -1, cv.LINE_AA)

            # drawing on the mask
            cv.circle(mask, center_left, int(l_radius), (255,255,255), -1, cv.LINE_AA)
            cv.circle(mask, center_right, int(r_radius), (255,255,255), -1, cv.LINE_AA)

        cv.imshow('Mask', mask)
        cv.imshow('img', frame)
        key = cv.waitKey(1)
        if key ==ord('q'):
            break
cap.release()
cv.destroyAllWindows()
```

CHAPTER - 5

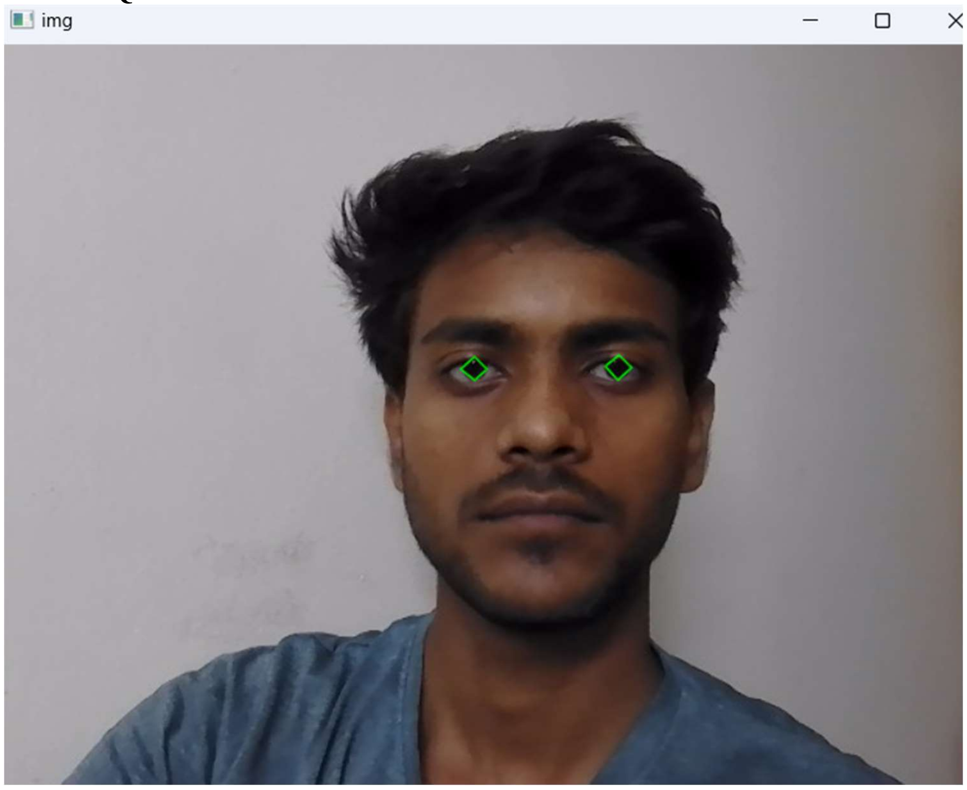
5.SCREEN SHOT

MASK





WITH SQUARE



CHAPTER – 6
CONCLUSION

The iris, with its unique and stable pattern, holds immense potential for applications in both security and medicine. To harness this potential, accurately isolating the iris region from an eye image is a critical initial step. This research delves into a variety of techniques designed to achieve this precise segmentation.

By systematically comparing the performance of these methods, we aim to contribute meaningfully to the advancement of iris recognition and analysis. Our goal is to develop robust and reliable iris segmentation algorithms that can effectively handle the challenges posed by diverse eye image conditions. Ultimately, this research seeks to lay a strong foundation for future developments in iris-based technologies, enabling more secure and efficient systems for identification and medical diagnosis.

CHAPTER – 7

FUTURE SCORE AND ENHANCEMENTS

In simpler terms, this means what can be done next to improve the project. For example, the project could be made better by:

- **Testing on more types of eye images:** Try the program on different kinds of photos, like blurry or low-light images.
- **Improving accuracy:** Make the program even better at finding the exact iris shape.
- **Speeding up the process:** Make the program work faster.
- **Trying new techniques:** Explore different ways to find the iris.
- **Combining with other features:** Use the iris information with other eye details for more advanced tasks.

REFERENCES

- https://r.search.yahoo.com/_ylt=Awr.w1QXG6lmiAQAuw9XNyoA;_ylu=Y29sbwNncTEEcG9zAzEEdnRpZAMEc2VjA3Ny/RV=2/RE=1723568151/RO=10/RU=https%3a%2f%2fwww.python.org%2fdoc%2f/RK=2/RS=SroBzSdxdmAefLkjxA.W4vi3Lh8-
- https://ai.google.dev/
- https://r.search.yahoo.com/_ylt=Awr99RNPG6lm6ZYB04hXNyoA;_ylu=Y29sbwNncTEEcG9zAzEEdnRpZAMEc2VjA3Ny/RV=2/RE=1723568207/RO=10/RU=https%3a%2f%2fpypi.org%2fproject%2fmediapipe%2f/RK=2/RS=a_rkKyJhmP72ZVvOWQOt_O8QdM0-
- https://stackoverflow.com/