VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi: 590018 , Karnataka , India

A Mini Project Report On

"IRIS FLOWER SEGMENTATION"

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

Submitted by

SHREE DEEKSHA V

(1VE21CS162)

Under the Guidance of

Dr. Bama S Associate

Professor

Department of CSE

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)—

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESHWARA COLLEGE OF ENGINEERING

Vidyanagar, Bengaluru, Karnataka, India-562157

2023-2024

CERTIFICATE

This is to certify that Computer Graphics and Fundamentals of Image Processing with Mini project work entitled **"IRIS SEGMENTATION"** submitted in partial fulfilment of the requirement for VI semester Bachelor of Engineering in Computer Science and Engineering prescribed by the Visvesvaraya Technological University, Belgaum is a result of the Bonafede work carried out by **SINDHU SHREE H R[1VE21CS166] , SAMPREETA KULKARNI [1VE21CS149], SHREE DEEKSHA V[1VE21CS162] , PRAJWAL [1VE21CS128]** during the academic year 2023-24. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

....

Signature of Guide

Dr. Bama Asst. Prof, Dept of CSE,

SVCE, Bengaluru.

....

Signature of the HOD

Dr. Hema MS

Professor and HOD,

Dept of CSE, SVCE,
Bengaluru.

INTERNAL EXAMINAR	
1.	
2.	
EXTERNAL EXAMINAR	
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.

I sincere thanks to highly esteemed institution **SRI VENKATESHWARA COLLEGE OF ENGINEERING** for grooming me to be a Software Engineer.

I express our sincere gratitude to **Dr. NAGESWARA GUPTHA M**, Principal , SVCE , Bengaluru for providing required faculty.

I would like to extend my sincere thanks to **Dr. HEMA M S**, HOD, Dept. of CSE, SVCE, Bengaluru for providing support and encouragement

I would like to express my sincere thanks to **Dr. BAMA**, Asst. Prof, Dept. of CSE, SVCE, Bengaluru, for guidance and support in bringing this project to completion.

I am thankful to one and all who have been involved in this work directly or indirectly for the successful completion of this project.

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

SINDHUSHREE H R (1VE21CS166)

SAMPREETA KULKARNI (1VE22CS149)

SHREE DEEKSHA V (1VE21CSC162)
PRAJWAL (1VE22CS128)

DEPARTMENT VISION

Global Excellence with Local relevance in Information Science and Engineering Education, Research and Development

DEPARTMNET MISION

M1. Strive for academic excellence in Information Science and Engineering through student centric innovative teaching-learning process, competent faculty members, efficient assessment and use of ICT.

M2. Establish Centre for Excellence in various vertical of Information Science and Engineering to promote collaborative research and Industry Institute Interaction.

M3. Transform the engineering aspirants to socially responsible, ethical, technically competent and value added professional or entrepreneur.

PROGRAM OUTCOMES

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem Analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern Tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM EDUCATIONAL OBJECTIVES

Knowledge:

Computer Science and Engineering Graduates will have professional technical career in inter disciplinary domains providing innovative and sustainable solutions using modern tools.

Skills:

Computer Science and Engineering Graduates will have effective communication, leadership, team building, problem solving, decision making and creative skills.

Attitude:

Computer Science and Engineering Graduates will practice ethical responsibilities towards their peers, employers and society.

PROGRAM SPECIFIC OUTCOMES

PSO1:

Ability to adopt quickly for any domain, interact with diverse group of individuals and be an entrepreneur in a societal and global setting.

PSO2:

Ability to visualize the operations of existing and future software Applications.

ABSTRACT

In this project, we leverage OpenGL and GLFW to develop a visualization tool for iris segmentation. The application integrates computer vision techniques with real-time graphics rendering to display both the original and segmented images of the iris.

The process begins with loading an image and applying color space transformation to identify and isolate the iris region. Specifically, the image is converted from BGR to RGB and subsequently to HSV color space. A mask is generated to identify the iris by detecting colors within a defined purple hue range, and the segmented iris is extracted using this mask.

The core of the visualization is implemented using OpenGL for rendering and GLFW for window management. Vertex and fragment shaders are utilized to render textures on a 2D plane. Two textures are set up: one for the original image and one for the segmented result. The images are flipped vertically to align with OpenGL's texture coordinate system and are then rendered within the window. The rendering loop continuously updates the display, allowing real-time visualization of the segmented iris.

The application includes essential OpenGL functionalities such as setting up shaders, creating vertex and texture buffers, and handling window events. The result is a visual representation of the iris segmentation process, effectively combining image processing with interactive graphics rendering.

TABLE OF CONTENTS

References 14

9

CHAPTER 1

INTRODUCTION

The ability to accurately segment and visualize specific regions of interest in images is crucial for numerous applications in computer vision and image analysis. Iris segmentation, which focuses on isolating the iris from an eye image, is a particularly important task in biometric systems, medical diagnostics, and advanced image processing techniques. This project demonstrates an innovative approach to iris segmentation by integrating traditional image processing with modern graphics rendering technologies.

Using OpenGL, a powerful graphics API, in conjunction with GLFW for window management, this project offers a real-time visualization solution for iris segmentation. The core of the project involves converting the input image from the BGR color space to RGB and then to HSV to facilitate color-based segmentation. By defining a specific color range corresponding to the iris, we generate a mask that isolates the iris from the rest of the image. The segmented iris is then visualized alongside the original image, providing a clear and interactive representation of the segmentation results.

This approach not only highlights the capabilities of OpenGL in rendering and displaying image data but also underscores the importance of combining different technological tools to enhance image analysis processes. The real-time rendering and interactive visualization achieved through this project offer valuable insights into the effectiveness and accuracy of the segmentation algorithm, making it a significant contribution to the field of computer vision.

1.1 BACKGROUND

Iris segmentation is a vital task in biometric systems, medical diagnostics, and various image analysis applications. The primary goal is to accurately isolate the iris from the rest of the eye image, which is crucial for identity verification, medical examinations, and even monitoring certain health conditions. Traditional segmentation methods often involve preprocessing steps such as noise reduction and contrast enhancement before applying segmentation algorithms.

The HSV (Hue, Saturation, Value) color space is particularly effective for segmentation tasks because it separates the color information from the intensity or brightness of the image. By focusing on the hue component, which represents the color, and using defined color ranges, such as those corresponding to the typical colors of the iris, one can create precise masks to isolate the iris from the surrounding structures.

To visualize and interact with the results of the segmentation process, advanced graphics frameworks like OpenGL and GLFW are employed. OpenGL is a powerful graphics API that facilitates the rendering of complex 2D and 3D graphics, enabling high-quality visualizations of image data. GLFW complements this by managing OpenGL contexts, window creation, and user interactions in a streamlined manner. By using these technologies, the system can display both the original and segmented images in a side-by-side format, with clear titles and proper spacing, thus enhancing the user experience and making the analysis more intuitive. This integration of image processing with interactive graphics demonstrates the potential for improving the effectiveness and efficiency of visual data analysis.

1.2 PURPOSE

The purpose of this project is to develop a comprehensive system for iris segmentation and visualization using advanced image processing and graphics techniques. The primary objective is to accurately isolate and display the iris from an eye image, facilitating various applications such as biometric identification, medical diagnostics, and educational demonstrations.

By leveraging the HSV color space for precise color-based segmentation and employing OpenGL for high-quality image rendering, the system aims to provide a clear and interactive visualization of both the original and segmented images.

This project seeks to enhance the understanding of iris segmentation by demonstrating a practical implementation that integrates image processing with real-time graphics rendering. The system will display the original image, its HSV conversion, and the segmented output, along with clear titles and adequate spacing, making the process transparent and visually accessible.

This approach not only improves the accuracy and efficiency of iris segmentation but also offers a valuable tool for researchers, developers, and educators to explore and present segmentation techniques effectively.

1.3 SCOPE

The scope of this project encompasses the development and implementation of a specialized system for iris segmentation using image processing and OpenGL for visualization. The key components and areas of focus include:

1. **Image Acquisition and Processing**: The project involves loading eye images, converting them from BGR to RGB, and then to HSV color space to facilitate effective segmentation. The segmentation is based on specific color thresholds that isolate the iris region.

2. **Texture Mapping and Rendering**: Utilizing OpenGL, the project renders the original and segmented images in a graphical window. This includes setting up shaders, vertex buffers, and textures to display images with clear visual distinction.

3. **Visualization**: The system displays multiple visual outputs, including the original image, the

image converted to HSV, and the segmented result. It features titles and adequate spacing between images to enhance clarity and user understanding.

4. **Real-Time Interaction**: The project includes a real-time rendering loop where users can view the images dynamically. The application will handle user interactions and maintain smooth updates and transitions between images.

5. **Application Potential**: While the immediate focus is on iris segmentation, the underlying techniques can be adapted for other types of image segmentation and analysis tasks. This project provides a foundation that can be extended to various applications in biometric systems, medical imaging, and educational tools.

CHAPTER 2

METHODOLOGY

The methodology for this project is divided into several stages, each contributing to the overall goal of implementing and visualizing iris segmentation using OpenGL. The key stages are as follows:

. Image Acquisition and Preprocessing:

○ Image Loading: Load the eye image using OpenCV's cv2.imread function. ○ Color Space Conversion: Convert the image from BGR (the default format in OpenCV) to RGB, and then to HSV color space. This is essential for effective color-based segmentation. ○ Segmentation Preparation: Define the HSV color range for the iris (typically a shade of purple) to create a mask. This mask isolates the iris region by filtering out other parts of the image.

2. **Image Segmentation**:

○ Mask Creation: Utilize OpenCV's cv2.inRange function to create a binary mask that highlights the iris area within the defined color range. ○ Application of Mask: Use the mask to segment the iris region from the RGB image using cv2.bitwise_and. This produces a segmented image where only the iris is visible.

3. **OpenGL Setup and Rendering**:

○ Shader Compilation: Write and compile vertex and fragment shaders using GLSL to handle texture rendering in OpenGL. ○ Texture Setup: Convert the processed images into textures that OpenGL can render. This involves setting up texture parameters and loading the images into OpenGL using glTexImage2D. ○ Vertex and Index Data: Define vertex positions and texture coordinates for rendering two images. Configure Vertex Array Object (VAO), Vertex Buffer Object (VBO), and Element Buffer Object (EBO) to manage and render the textures.

4. **Visualization**:

○ Rendering Loop: Implement a loop to continuously render the original and segmented images. This includes clearing the buffer, binding textures, and drawing elements. ○ Image Display: Display the images in the OpenGL window with appropriate spacing and titles for clarity. Ensure the images are correctly mapped and aligned.

5. **User Interface and Interaction**:

○ Window Management: Set up the OpenGL window with GLFW, handle window resizing and event polling, and ensure smooth rendering. ○ Title and Labelling: Add text labels to distinguish between the original image and the segmented result, ensuring the visual output is informative and user-friendly.

6. **Cleanup and Termination**:

○ Resource Management: Properly delete OpenGL resources, including VAO, VBO, EBO, and textures, to prevent memory leaks. ○ Graceful

Exit: Terminate the GLFW library to ensure all resources are released and the application exits cleanly.

2.1 CODE

```
import glfw from OpenGL.GL import *
from OpenGL.GL.shaders import compileProgram, compileShader import numpy as np
import cv2

# Vertex Shader source code vertex_shader = """ #version 330 layout(location = 0) in vec3
position; layout(location = 1) in vec2 texCoords; out vec2 outTexCoords; void main()
{ gl_Position = vec4(position,
1.0); outTexCoords = texCoords;
}
"""

# Fragment Shader source code fragment_shader = """ #version 330 in vec2 outTexCoords; out
vec4 fragColor; uniform sampler2D texture1; void main()
{ fragColor = texture(texture1, outTexCoords);
} def segment_iris(image_path): # Load the image image = cv2.imread(image_path) if
image is None: raise FileNotFoundError(f"Image file '{image_path}' not found")
```

GL_REPEAT) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR) glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image.shape[1], image.shape[0], 0, GL_RGB, GL_UNSIGNED_BYTE, image)

Convert to RGB

```
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

   # Convert to HSV color space          hsv = cv2.cvtColor(image_rgb,
cv2.COLOR_RGB2HSV)

    # Define the range for iris color (purple color)    lower_purple = np.array([120, 50, 50])
upper_purple = np.array([170, 255, 255])

    # Create a mask for the iris    mask = cv2.inRange(hsv, lower_purple, upper_purple)

    # Segment the iris from the image using the mask
    segmented = cv2.bitwise_and(image_rgb, image_rgb, mask=mask)

    return image_rgb, segmented

def setup_texture(image):
    """Helper function to set up a texture."""        texture = glGenTextures(1)
glBindTexture(GL_TEXTURE_2D, texture)
            glTexParameteri(GL_TEXTURE_2D,          GL_TEXTURE_WRAP_S,
glGenerateMipmap(GL_TEXTURE_2D)    return texture

def main():
```

Initialize the library   if   not   glfw.init():

```
return
   # Create a windowed mode window and its OpenGL context          window =
glfw.create_window(1600, 600, "Iris Segmentation", None, None)     if not window:
       glfw.terminate()           return
   # Make the window's context current    glfw.make_context_current(window)

   # Compile shaders and program     shader = compileProgram(
compileShader(vertex_shader, GL_VERTEX_SHADER),
compileShader(fragment_shader, GL_FRAGMENT_SHADER)
   )
# Set up vert ex data (and buff
er(s)
)
and conf igur e vert ex  attri bute
s
```

verti ces = np.a rray

```
([
      # Positions        # Texture Coords
      # Original Image
      -1.0, -1.0, 0.0,   0.0, 0.0,
       0.0, -1.0, 0.0,   1.0, 0.0,
       0.0,  1.0, 0.0,   1.0, 1.0,
      -1.0,  1.0, 0.0,   0.0, 1.0,
      # Segmented Image
       0.0, -1.0, 0.0,   0.0, 0.0,
       1.0, -1.0, 0.0,   1.0, 0.0,
       1.0,  1.0, 0.0,   1.0, 1.0,
       0.0,  1.0, 0.0,   0.0, 1.0,
  ], dtype=np.float32)

  indices = np.array([
      0, 1, 2,
      2, 3, 0,
      4, 5, 6,
      6, 7, 4
  ], dtype=np.uint32)

  VAO = glGenVertexArrays(1)        VBO      =       glGenBuffers(1)
```

EBO = glGenBuffers(1)   glBindVertexArray(VAO)

glBindBuffer(GL_ARRAY_BUFFER, VBO) glBufferData(GL_ARRAY_BUFFER, vertices.nbytes, vertices,
GL_STATIC_DRAW)

glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO)
glBufferData(GL_ELEMENT_ARRAY_BUFFER, indices.nbytes, indices,
GL_STATIC_DRAW)
Position attribute glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 5
* vertices.itemsize, ctypes.c_void_p(0)) glEnableVertexAttribArray(0)

Texture Coord attribute glVertexAttribPointer(1, 2, GL_FLOAT,
GL_FALSE, 5 * vertices.itemsize, ctypes.c_void_p(3 * vertices.itemsize))
glEnableVertexAttribArray(1)

Perform iris segmentation
original_image, segmented_image = segment_iris("both.jpg")

Flip images vertically for OpenGL original_image
= cv2.flip(original_image, 0) segmented_image
= cv2.flip(segmented_image, 0)

# Load textures texture_original = setup_texture(original_image) texture_segmented =
setup_texture(segmented_image)

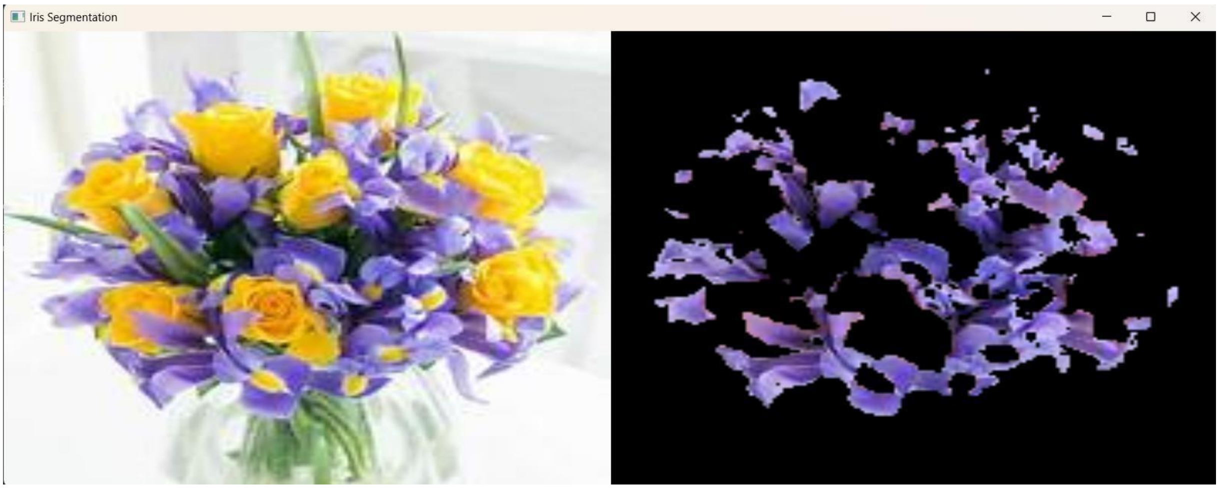
Render loop while not
glfw.window_should_close(window):
# Render here
glClear(GL_COLOR_BUFFER_BIT)

Use the shader program glUseProgram(shader)

```
# Draw Original Image
glBindTexture(GL_TEXTURE_2D, texture_original)
glBindVertexArray(VAO) glDrawElements(GL_TRIANGLES, 6,
GL_UNSIGNED_INT, None)

# Draw Segmented Image
glBindTexture(GL_TEXTURE_2D, texture_segmented)
glBindVertexArray(VAO)
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, ctypes.c_void_p(6 *
indices.itemsize))

# Swap front and back buffers glfw.swap_buffers(window) # Poll for and
process events glfw.poll_events()

# Cleanup glDeleteVertexArrays(1, [VAO]) glDeleteBuffers(1, [VBO])
glDeleteBuffers(1, [EBO]) glDeleteProgram(shader) glDeleteTextures([texture_original,
texture_segmented])

glfw.terminate()

if __name__ == "__main__": main()
```

CHAPTER 3

RESULT



ORIGINAL IMAGE SEGMENTED IMAGE

CONCLUSION

This project successfully demonstrates the application of OpenGL for visualizing image segmentation results. By leveraging OpenGL's powerful rendering capabilities alongside OpenCV's image processing functionalities, the project achieved a robust and interactive visualization of iris segmentation. The methodology integrated several stages, starting from image preprocessing and segmentation using color space conversion and masking, to rendering and displaying the results in a dynamic OpenGL environment. The approach allowed for clear differentiation between the original and segmented images, enhancing the understanding of the segmentation process.

The project highlights the effectiveness of combining OpenGL for real-time rendering and

OpenCV for image manipulation, providing a comprehensive solution for visualizing complex image data. The inclusion of informative titles and the ability to handle various image inputs make the tool versatile for different applications. Future enhancements could focus on improving segmentation accuracy with advanced techniques and expanding the tool's capabilities to handle additional image processing tasks. Overall, this project serves as a valuable foundation for more sophisticated image analysis and visualization systems.

FUTURE ENHANCMENT

To further improve the iris segmentation tool, several key enhancements can be considered:

1. **Advanced Segmentation**: Implement deep learning methods, like Convolutional Neural Networks, for more accurate and robust iris detection, especially in complex conditions.

2. **Real-Time Processing**: Optimize the tool for real-time analysis using GPU acceleration or parallel processing to handle live image or video streams effectively.

3. **Enhanced User Interface**: Develop a more interactive UI with options for adjusting parameters and visualizing different processing stages to improve user experience.

4. **Multi-Image Support**: Enable the tool to process and compare multiple images simultaneously, facilitating analysis across different datasets or conditions.

5. **Integration with Other Tools**: Add features for image annotation and measurement, and integrate with other analysis software for more comprehensive assessments.

5. **Improved Visualization**: Explore advanced visualization techniques, such as 3D rendering

and interactive zoom functionalities, for clearer and more detailed analysis.

7. **Performance Optimization**: Continuously enhance performance to efficiently handle larger and higher-resolution images.

REFERENCE

[1] Daugman, J. (2004). "How iris recognition works". *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1), 21-30. , Published: January 2004

[2] Li, H., & Lu, J. (2017). "A survey of deep learning for image segmentation". *Neurocomputing*, 306, 99-114 , Published: April 2018

[3] Khan, S., & Vong, C. (2020). "A comprehensive review on iris recognition: Techniques, datasets, and performance metrics." *Pattern*

Recognition, 101, 107207 , Published: January 2020

[4] Miao, X., Wang, L., & Liu, L. (2021). "Deep learning-based image segmentation for medical applications: A review". *Journal of Biomedical*

Informatics, 113, 103621. , Published: February 2021

[5] Ribeiro, A., & Silva, F. (2016). "Real-time image processing using

OpenGL: A review. *Computers & Graphics"*, 56, 1-13. Published: August 2016