**Chapter 1**

**INTRODUCTION**

#### Computer Graphics

Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results. Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. Th advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

#### OpenCV

**OpenCV Interface:**

OpenCV (Open Source Computer Vision Library) is a comprehensive open-source library for computer vision, image processing, and machine learning. It provides a wide range of functions and algorithms to manipulate images and perform various tasks in computer vision applications. OpenCV offers interfaces to work with images, videos, and camera streams, as well as tools for feature extraction, object detection, tracking, and more.

Most functionalities in OpenCV are accessed through functions in several core modules:

1. **Main Core (cv2)**:
   * This module contains basic data structures and functions for manipulating images and matrices. Functions typically start with cv2.
2. **Image Processing (cv2)**:
   * The cv2 module includes functions for various image processing tasks such as filtering, transformations, color space conversions, and histogram operations.
3. **Video I/O (cv2)**:
   * Provides functions to capture video from cameras or video files, and to write video streams to file.

**OpenCV Overview:**

* OpenCV is designed to provide a uniform interface across different hardware and operating systems, making it widely portable.
* It offers a rich set of functionalities for both 2D and 3D image processing, including support for various file formats and camera interfaces.
* OpenCV functions are optimized for performance and efficiency, often leveraging hardware acceleration where available.
* OpenCV provides a vast library of optimized algorithms and functions for tasks such as image filtering, feature detection, object recognition, and video analysis.

**Chapter 2**

**SYSTEM SPECIFICATION AND REQUIREMENTS**

**2.1 SOFTWARE REQUIREMENTS**

* **Programming language:** Python
* **Libraries:** OpenCV, TensorFlow, Keras, FER, Streamlit
* **Operating system:** Any OS with Python support (Linux, Windows, macOS)
* **Additional Software:** Webcam or camera

**2.2 HARDWARE REQUIREMENTS**

* **Processor:** Dual Core Processor
* **RAM:** 2GB or more
* **Storage:** 40GB Hard disk or more
* **Input Devices:** Mouse, Keyboard, Webcam or camera

**2.3 FUNCTIONAL REQUIREMENTS**

#### OpenCV, TensorFlow, Keras, FER, and Streamlit:

To implement a real-time emotion detection system, several Python libraries are utilized:

* **OpenCV:** Used for video capture and image processing. It provides the functionality to access the webcam and process frames in real-time.
* **TensorFlow & Keras:** These libraries are used for deep learning and neural network models to analyze and predict emotions from facial expressions.
* **FER (Facial Emotion Recognition):** A Python library built on top of TensorFlow/Keras that simplifies the emotion detection process. It uses MTCNN for face detection and provides easy-to-use functions to detect emotions.
* **Streamlit:** A framework to create the web interface for the real-time emotion detection system. It allows for easy integration of the emotion detection functionality into a web app.

**Chapter 3**

**ABOUT THE PROJECT**

**3.1 INTRODUCTION**

The Real-Time Emotion Detection Website is a cutting-edge project that uses advanced computer vision and deep learning techniques to analyze facial expressions and detect emotions in real-time. Built with Python and powerful libraries like Streamlit, TensorFlow, Keras, and OpenCV, the application captures and analyzes live video streams from a webcam. The intuitive interface allows users to interact with the system, which detects emotions such as happiness, sadness, anger, and surprise. The project features real-time video display with emotion labels and bounding boxes, optimized for smooth processing. This project showcases the practical application of computer vision and machine learning in creating responsive and interactive web applications.

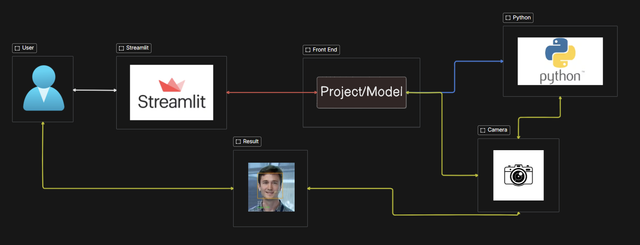
**3.2 OBJECTIVES**

The aim of this project is to develop a real-time emotion detection website using advanced computer vision and deep learning techniques. It captures and analyzes live video streams from a webcam to detect emotions like happiness, sadness, anger, and surprise. The project also seeks to create a user-friendly interface with Streamlit, allowing easy interaction with the system, and to apply the skills learned in class to demonstrate the practical application of these technologies.

**3.3 BUILT IN FUNCTIONS**

* **cv2.VideoCapture()**: Opens the webcam for capturing video frames.
* **cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)**: Converts the color space of the captured frame from BGR to RGB.
* **FER(mtcnn=True)**: Initializes the FER emotion detector with MTCNN for face detection.
* **detector.detect\_emotions(rgb\_frame)**: Detects emotions in the given RGB frame.
* **cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)**: Draws a rectangle around the detected face.
* **cv2.putText()**: Adds text to the frame, used to display the detected emotion and its score.
* **st.error("Error: Could not open webcam."):** Displays an error message in the Streamlit app if the webcam cannot be opened.
* **st.stop():** Stops the execution of the Streamlit app.
* **uuid.uuid4():** Generates a unique identifier, which can be used for creating unique keys.
* **cv2.getTextSize(text, font, font\_scale, font\_thickness):** Calculates the size of the text for drawing purposes.
* **st.image(cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)):** Updates the Streamlit image element with the latest frame from the webcam.
* **cap.read():** Reads a frame from the webcam.
* **st.write('This application detects faces and their emotions in real-time using your webcam.'):** Adds descriptive text to the Streamlit app.
* **st.image([]):** Initializes a Streamlit image element to display video frames in the app.
* **st.title('Real-time Emotion Detection'):** Sets the title of the Streamlit app.
  1. **USER DEFINED FUNCTIONS**
* **draw\_text(frame, text, x, y) :** This function is designed to draw text with a background rectangle on an image frame. It can be used to annotate images with information such as detected emotions and their scores.
* font\_scale, font, and font\_thickness define the appearance of the text.
* **cv2.getTextSize()** is used to calculate the size of the text box.
* **cv2.rectangle()** draws a filled rectangle to create a background for the text.
* **cv2.putText()** draws the actual text on the frame.

**3.5 DATA FLOW DIGRAM**



**Fig 3.1 – Data Flow Diagram**

**3.5 SOURCE CODE / PSUEDO CODE**

import os

os.environ['TF\_ENABLE\_ONEDNN\_OPTS'] = '0'

import streamlit as st

import cv2

from fer import FER

import numpy as np

import uuid # Import UUID library to generate unique keys

# Initialize the FER emotion detector with MTCNN for face detection

try:

detector = FER(mtcnn=True)

except Exception as e:

st.error("Error initializing FER with MTCNN: {}".format(e))

st.stop()

# Streamlit app setup

st.title('Real-time Emotion Detection')

st.write('This application detects faces and their emotions in real-time using your webcam.')

st.title('Team Members')

st.write('ANISH KUMAR 1AY21CS028')

st.write('ADITYA KHATRIYA 1AY21CS018')

st.write('ADITYA JYOTI SAHU 1AY21CS017')

st.write('ADITYA ARUN KUMAR 1AY21CS016')

# Initialize the webcam

cap = cv2.VideoCapture(0)

if not cap.isOpened():

st.error("Error: Could not open webcam.")

st.stop()

# Function to draw text with background

def draw\_text(frame, text, x, y):

font\_scale = 0.6

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_thickness = 2

text\_size = cv2.getTextSize(text, font, font\_scale, font\_thickness)[0]

text\_x = x

text\_y = y - text\_size[1]

cv2.rectangle(frame, (text\_x, text\_y), (text\_x + text\_size[0], text\_y + text\_size[1]), (0, 0, 0), cv2.FILLED)

cv2.putText(frame, text, (text\_x, y), font, font\_scale, (255, 255, 255), font\_thickness)

# Streamlit video capture and processing

frame\_window = st.image([])

while True:

ret, frame = cap.read()

if not ret:

st.error("Error: Could not read frame from webcam.")

break

rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

result = detector.detect\_emotions(rgb\_frame)

for face in result:

(x, y, w, h) = face['box']

cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

dominant\_emotion = max(face['emotions'], key=face['emotions'].get)

score = face['emotions'][dominant\_emotion]

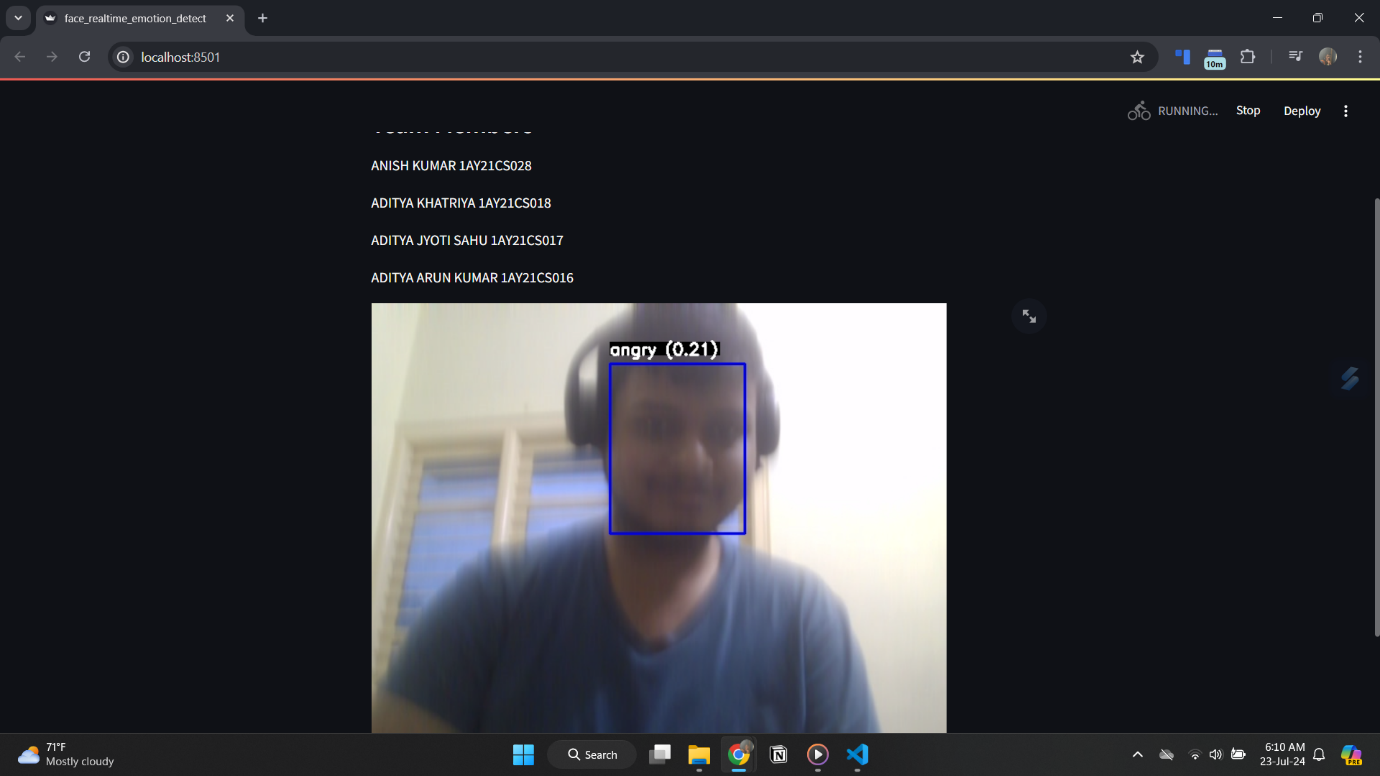
draw\_text(frame, f'{dominant\_emotion} ({score:.2f})', x, y - 10)

frame\_window.image(cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB))

cap.release()

**Chapter 4**

**RESULT**



**Fig 4.1 – Result**

**Chapter 5**

**CONCLUSION AND FUTURE WORK**

The Real-Time Emotion Detection Website has been successfully implemented on Python, leveraging Streamlit and a suite of powerful libraries including OpenCV, TensorFlow, Keras, and FER. This application offers a seamless interface for users to detect emotions in real-time through facial expressions captured via a webcam.

**FUTURE WORK:**

* **Support for Multiple Cameras**: Extend the application to support multiple cameras for simultaneous emotion detection from different viewpoints.
* **Pattern Filling**: Integrate pattern filling capabilities to enhance visual feedback and customization options for users.
* **3D Transformations**: Implement support for 3D transformations to enable more dynamic and immersive visualization of detected emotions.
* **Layer Transparency**: Introduce transparency settings for layers in the user interface, enhancing clarity and customization options.

**REFERENCES**

[1] **OpenCV Documentation**: Official documentation for OpenCV library functions and examples.

[2] **TensorFlow Documentation**: Official documentation for TensorFlow, used for deep learning models.

[3] **Keras Documentation**: Official documentation for Keras, a high-level neural networks API.

[4] **FER (Facial Expression Recognition) GitHub Repository**: Repository providing FER library for facial emotion detection.

[5] **Streamlit Documentation**: Official documentation for Streamlit, used for building web interfaces in Python.

[6] **MTCNN (Multi-task Cascaded Convolutional Networks) Paper**: Original paper on MTCNN used for face detection in the FER library. Paper: Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multi-task cascaded convolutional networks. IEEE Signal Processing Letters, 23(10), 1499-1503.

[7] **Deep Learning for Emotion Recognition in Images and Video**: Research paper providing insights into deep learning techniques for emotion recognition. Paper: Kim, B. H., Lee, Y. J., & Roh, J. H. (2016). Deep learning for emotion recognition: A survey. Frontiers in Psychology, 7, 1-16.