MINI PROJECT REPORT ON

Computer Vision

Yeshwantrao Chavan College of Engineering

Bachelor of Technology in Computer Technology

Submitted By:

Devid Deshmukh
CT- 44

Dinesh Bodhe
CT-45

Supervised By:

Priya Kotewar

Nagar Yuwak Shikshan Sanstha's

YESHWANTRAO CHAVAN COLLEGE OF ENGINEERING,
(An Autonomous Institution Affiliated to Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur)

NAGPUR – 441 110

2025-26

Project 1

Aim: Use Different Filters for Image Smoothing

Problem Definition:

The objective is to understand and implement different image smoothing filters that help reduce noise, preserve important features, and improve the visual quality of images.

Theory:

Image smoothing, also known as blurring, is a fundamental operation in image processing. It helps remove noise and small details from an image.
There are several filters used for smoothing:

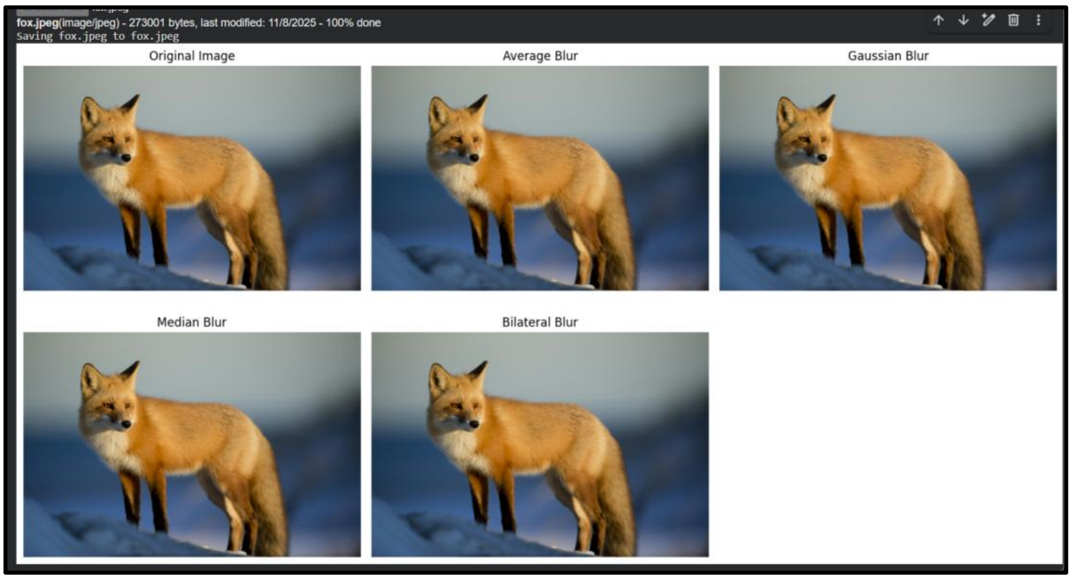
- Average Filter: Replaces each pixel with the average of its neighboring pixels.
- Gaussian Filter: Uses a Gaussian function for averaging, which provides smoother transitions and preserves edges better.
- Median Filter: Replaces each pixel with the median of neighboring pixels, effective against salt-and-pepper noise.
- Bilateral Filter: Smooths images while preserving edges by combining spatial and intensity differences.

These techniques are crucial for preprocessing in computer vision applications such as object detection and segmentation.
Procedure & Execution:
1. Import necessary libraries (OpenCV, NumPy, Matplotlib).
2. Load an input image.
3. Apply various filters (Average, Gaussian, Median, Bilateral).
4. Display and compare the results.
5. Observe the differences between each smoothing technique.

Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from tkinter import Tk, filedialog

Tk().withdraw()
img_path = filedialog.askopenfilename(title="Select an image",
filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp")])

img = cv2.imread(img_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

avg_blur = cv2.blur(img, (5,5))
gaussian_blur = cv2.GaussianBlur(img, (5,5), 0)
median_blur = cv2.medianBlur(img, 5)
bilateral_blur = cv2.bilateralFilter(img, 9, 75, 75)

titles = ['Original Image', 'Average Blur', 'Gaussian Blur', 'Median Blur',
'Bilateral Blur']
images = [img, avg_blur, gaussian_blur, median_blur, bilateral_blur]

plt.figure(figsize=(15,8))
for i in range(5):
    plt.subplot(2,3,i+1)
    plt.imshow(images[i])
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Output:



```
fox.jpeg(image/jpeg) - 273001 bytes, last modified: 11/8/2025 - 100% done
Saving fox.jpeg to fox.jpeg
```

Output Analysis:

The output shows how each filter affects the image. The Average and Gaussian filters produce general blurring, while the Median filter removes noise effectively. The Bilateral filter maintains sharp edges while reducing noise, providing the best balance between smoothness and detail preservation.

Conclusion:

Different filters serve unique purposes in image processing. While simple filters smooth images uniformly, advanced filters like Bilateral help maintain edges, making them suitable for advanced vision tasks.

Project 2

Aim: Implement Facial Expression Recognition

Problem Definition:

Facial expression recognition is a vital area of computer vision that enables machines to interpret human emotions. The goal is to build a model capable of recognizing different emotional states from facial images automatically.

Theory:

Facial Expression Recognition (FER) identifies emotions from facial cues. Using deep learning, FER models analyze facial landmarks and extract features that correspond to expressions like happiness, sadness, anger, or surprise. The DeepFace library simplifies this process by combining pre-trained deep neural networks for emotion detection. It uses convolutional neural networks (CNNs) to analyze facial regions and classify emotions.

Procedure & Execution:

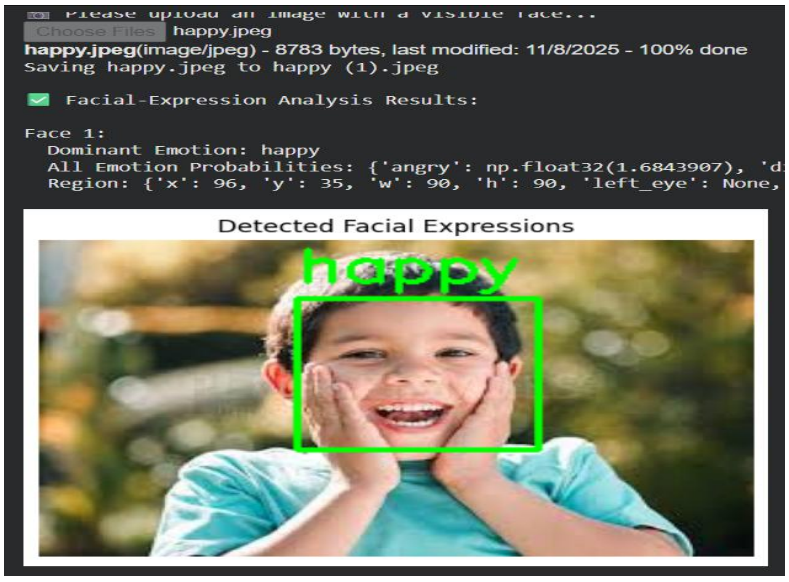
1. Install required libraries (DeepFace, OpenCV, Matplotlib).
2. Upload facial images.
3. Analyze each image using DeepFace to detect emotions.
4. Draw bounding boxes and emotion labels on faces.
5. Display the results and interpret outcomes.

Code:

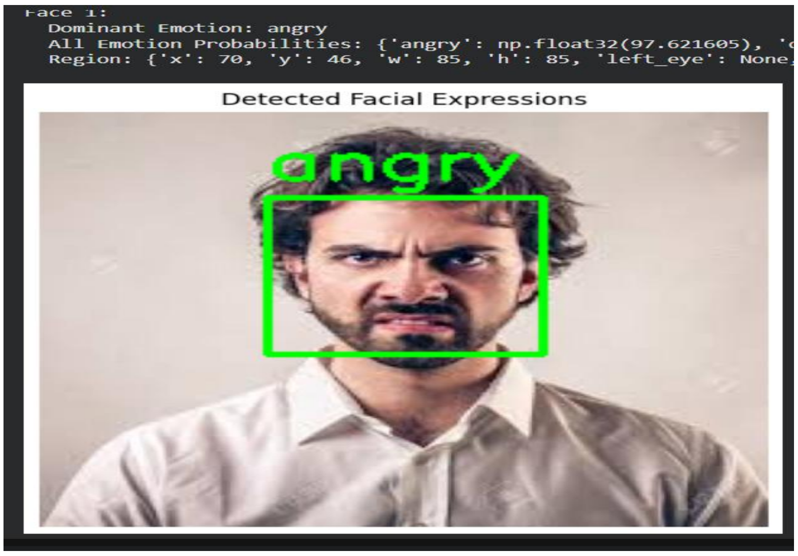
```
from deepface import DeepFace
import cv2
import matplotlib.pyplot as plt
from google.colab import files

uploaded = files.upload()
image_path = list(uploaded.keys())[0]

analysis = DeepFace.analyze(img_path=image_path, actions=['emotion'],
enforce_detection=False)

if isinstance(analysis, dict):
    analysis = [analysis]
```

```
img = cv2.imread(image_path)
for face in analysis:
    region = face.get('region', {})
    x, y, w, h = region.get('x',0), region.get('y',0), region.get('w',0),
region.get('h',0)
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
    cv2.putText(img, face['dominant_emotion'], (x, y-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,0), 2)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.title('Detected Facial Expressions')
plt.show()
```

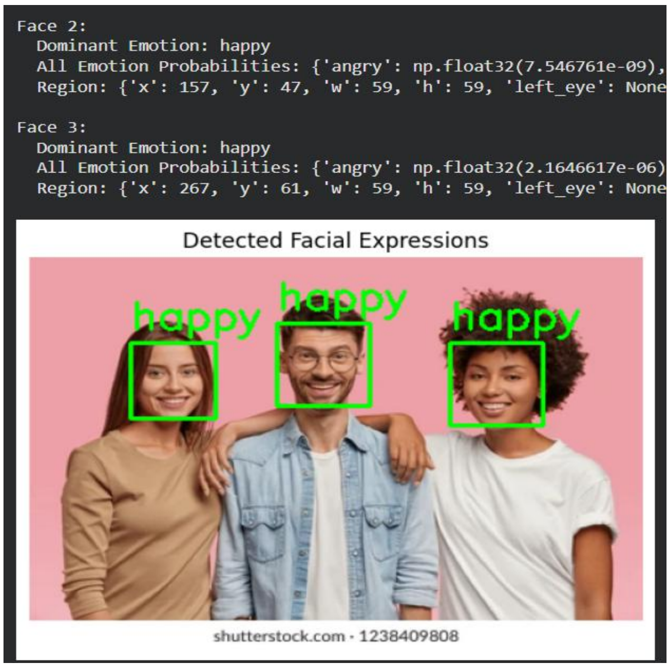
Output:



```
Please upload an image with a visible face...
Choose Files happy.jpeg
happy.jpeg(image/jpeg) - 8783 bytes, last modified: 11/8/2025 - 100% done
Saving happy.jpeg to happy (1).jpeg

✅ Facial-Expression Analysis Results:

Face 1:
  Dominant Emotion: happy
  All Emotion Probabilities: {'angry': np.float32(1.6843907), 'd
  Region: {'x': 96, 'y': 35, 'w': 90, 'h': 90, 'left_eye': None,
```



```
Face 1:
  Dominant Emotion: angry
  All Emotion Probabilities: {'angry': np.float32(97.621605), '
  Region: {'x': 70, 'y': 46, 'w': 85, 'h': 85, 'left_eye': None
```



```
Face 2:
  Dominant Emotion: happy
  All Emotion Probabilities: {'angry': np.float32(7.546761e-09),
  Region: {'x': 157, 'y': 47, 'w': 59, 'h': 59, 'left_eye': None

Face 3:
  Dominant Emotion: happy
  All Emotion Probabilities: {'angry': np.float32(2.1646617e-06)
  Region: {'x': 267, 'y': 61, 'w': 59, 'h': 59, 'left_eye': None
```

Output Analysis:
The system accurately detected and labeled various facial expressions such as happy, angry, and fearful. Each detected face is enclosed in a bounding box with its corresponding emotion label. The model works well for single and multiple faces in an image.

Conclusion:

Facial Expression Recognition using DeepFace effectively identifies emotions from images. This technology can be integrated into human-computer interaction systems, emotion analytics, and security applications.

GitHub Repository Link

Project Repository: https://github.com/devid-deshmukh/CV_Practicals_CT_44