PDF Title: Q4 Report

Face Blurring in Video – Project Report

1. Introduction

The purpose of this project is to create a Python program that detects and blurs faces in a video file. This program can be used for privacy protection in CCTV recordings, online videos, or public datasets. The system uses computer vision techniques to automatically detect faces and apply Gaussian blur to obscure identities, while saving the processed video for future use.

2. Objectives

- Capture a video file as input.
- Detect all faces in each frame using a pre-trained deep learning model.
- Apply Gaussian blur to every detected face.
- Display the processed video in real-time.
- Save the processed video to a new file for further use.

3. Tools and Technologies

- **Programming Language:** Python 3.x
- **Libraries:**
- OpenCV – for video processing, face detection, and image manipulation.
- NumPy – for numerical operations on arrays.
- **Pre-trained Models:**
- deploy.prototxt and res10_300x300_ssd_iter_140000.caffemodel – OpenCV DNN-based face detector.

Optional: FFmpeg – for converting videos to supported formats if necessary.

4. Methodology

Step 1: Video Input

- The program takes an existing video file as input.
- OpenCV's VideoCapture is used to read frames from the video.
- If the video cannot be opened, the program notifies the user to check the file path or format.

Step 2: Face Detection

- Faces are detected using OpenCV's deep learning-based DNN model (res10_300x300_ssd_iter_140000.caffemodel).
- Each frame is converted into a blob and fed to the network.
- Bounding boxes are generated around detected faces based on a confidence threshold.

Step 3: Face Blurring

- Each detected face is extracted as a region of interest (ROI).
- Gaussian blur is applied to the ROI.
- The blurred ROI is replaced in the original frame.

Step 4: Video Display

- The processed frames are displayed in real-time using cv2.imshow.
- Users can exit the display early by pressing q.

Step 5: Saving the Video

- OpenCV's VideoWriter saves the processed frames to a new file.
- The output video retains the original frame size and FPS for consistent playback.
- The codec used is mp4v for MP4 format; alternatively, XVID can be used for AVI files.

5. Challenges Faced

- Video compatibility: Some MP4 files could not be read by OpenCV due to codec issues. This was resolved by recommending video conversion using FFmpeg or using AVI format.
- Model file availability: The DNN model files (.caffemodel and .prototxt) are not included with OpenCV by default and had to be downloaded manually.
- Real-time performance: Processing high-resolution videos in real-time is resource-intensive. Downscaling or using optimized kernels helped improve speed.

6. Assumptions

- Input video exists and is readable by OpenCV.
- Model files are present in the working directory.
- Gaussian blur is sufficient for anonymization; pixelation could be added as an alternative.
- Output video format is compatible with the user's media player.

7. Results

- The program successfully detects and blurs faces in videos.
- Blurred faces are unrecognizable, preserving privacy.
- The output video is saved in the same format and frame size as the original.
- Live preview allows the user to verify processing before saving.

8. Conclusion

This project demonstrates a practical application of computer vision techniques for privacy protection. Using Python and OpenCV, we were able to automatically detect and blur faces in videos efficiently. The program can be extended in the future to include multiple anonymization methods (pixelation, masking), real-time webcam processing, or batch processing of multiple videos.

9. Future Work

- Add pixelation/mosaic blur options for face anonymization.
- Enable real-time webcam face blurring.
- Integrate multiple face detection models for improved accuracy.
- Allow batch processing of multiple video files automatically.

10. References

1. OpenCV Documentation: https://docs.opencv.org/
2. OpenCV DNN Face Detector: https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector
3. NumPy Documentation: https://numpy.org/doc/
4. FFmpeg Documentation: https://ffmpeg.org/documentation.html