INTRODUCTION

The Human Motion Detection Video Recorder project is a python script utilizes OpenCV and YOLOv3 , a deep learning-based object detection model , to detect human motion in a live video feed. It captures video from a specified source (like a phone's camera) , identifies humans using YOLOv3 , and draws bounding boxes around detected individuals . When a human is detected , it enters recording mode , saving a video clip of the detected motion . Upon motion cessation , it processes the recorded clip , and removes the original temporary file . The script runs until the user exits , offering real-time human motion monitoring and recording capabilities .

OBJECTIVE

The objective of this Python script Project is to perform real-time human motion detection using YOLOv3 and OpenCV . it aims to identify and track humans in a live video feed automatically , making their presence with bounding boxes . Additionally , it provides functionality to record video clips of detected motion , creating a comprehensive system for monitoring and analysing human movement in a given environment .

ALGORITHM DESIGN

1. Initialization and setup :
    - Load YOLOv3 model with weights and configuration.
    - Read class names for COCO dataset.
    - Define paths and setting for Droid Cam connection and background image.
2. Video Capture and Preprocessing :
    - Initialize video capture (from Droid Cam or other source).
    - Check successful video stream opening.
    - Set dimensions for the camera feed.
    - Load and resize the background image.
3. Main Loop :
    - Continuously read frames from the video feed.
    - Preprocess frames as blobs for YOLOv3 input.
    - Perform forward pass through the network to obtain output layers.
    - Detect humans within the frame based on YOLOv3 predictions.
    - Track detected humans by drawing bounding boxes around them.
    - Display the video feed with bounding boxes and a marker when humans are detected.
4. Recording and Processing :
    - When a human is detected :

        Initiate recording if not already recording.

        Save the recorded clip as an AVI file.
    - Upon motion cessation :

        Stop recording and release the video writer.

        Read the recorded clip, adjust its speed, and save it as a slowed-down MP4 file.

        Remove the original recorded clip.
5. User Interaction :
    - Check for the 'q' key to exit the loop and close the video windows.
6. Cleanup :
    - Release the video capture and close all OpenCV windows.

TECHNOLOGY USED

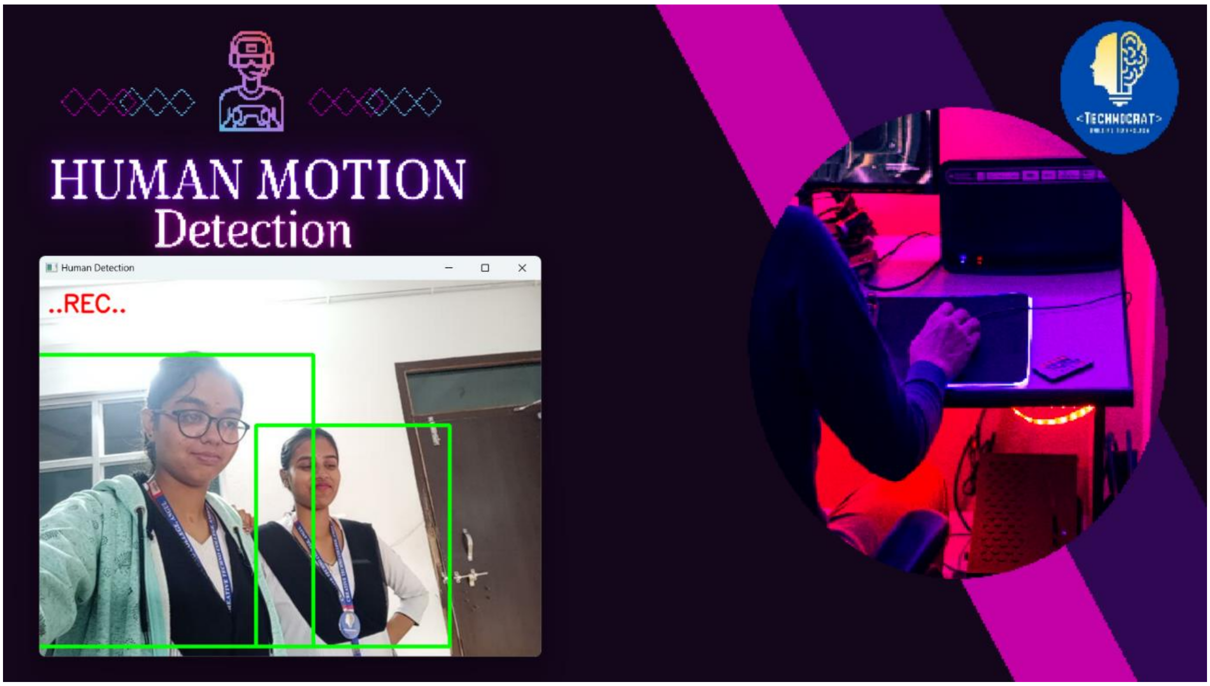
1. OpenCV (cv2): Used for video capture, frame processing, drawing bounding boxes, and displaying video streams.
2. YOLOv3: A deep learning-based object detection model utilized for identifying and localizing humans within the video feed.
3. Numpy: Used for numerical computations and array manipulations, particularly for handling data within the YOLOv3 model.
4. MoviePy: Employed for video manipulation tasks like adjusting playback speed (slowing down the recorded footage).
5. Operating System Interaction (os): Used for managing files and handling temporary video recordings.

WORKING METHODS

1. Initialization and Setup :
    - cv2.dnn.readNet: Loads YOLOv3 model with weights and configuration.
    - open: Reads class names for COCO dataset.
    - Setting up paths, configurations, and video sources.
2. Video Processing and Detection :
    - cv2.VideoCapture: Initializes video capture from the specified source (DroidCam or other device).
    - video.read(): Reads frames from the video feed.
    - Preprocessing frames using YOLOv3 requirements (cv2.dnn.blobFromImage).
    - Forward pass through the YOLOv3 network to detect humans in the frame.
    - Drawing bounding boxes around detected humans using OpenCV functions.

3. Recording and Processing :
    - Start and stop video recording when human presence is detected or motion ceases.
    - Using cv2.VideoWriter, saves the recorded clip as an AVI file.
    - Read the recorded clip with VideoFileClip from MoviePy for speed adjustment.
    - Adjust playback speed and save the slowed-down video using write_videofile.
    - Remove the original recorded clip using os.remove after speed adjustment.
4. User Interaction :
    - Check for user input ('q' key) to exit the loop and terminate the program.
    - Displays the processed video stream and bounding boxes in OpenCV windows.
5. Cleanup :
    - Release the video capture (video.release()) and close all OpenCV windows (cv2.destroyAllWindows()).

PROJECT SCREEN SHOTS



FUTURE SCOPE

1. Improved Object Detection :
   - Implement more advanced object detection models beyond YOLOv3 for increased accuracy and efficiency.
   - Explore models trained specifically for human detection to enhance precision in identifying human movements.
2. Real-Time Analytics :
   - Integrate analytics to extract data from detected motions, such as counting the number of individuals, analysing movement patterns, or estimating crowd density.
3. Multi-Camera Support :
   - Extend functionality to support multiple cameras or video sources, enabling surveillance of larger areas or complex environments.

4. Smart Alerts and Notifications:
    - Implement a system to trigger alerts or notifications when specific movements or events are detected, enhancing its use for security or monitoring purposes.
5. Integration with AI Assistants:
    - Integrate with AI assistants or voice recognition systems to enable voice-controlled commands for starting/stopping recording or adjusting settings.
6. Cloud Integration and Storage:
    - Allow seamless integration with cloud storage platforms for storing recorded videos or facilitating remote access to footage.
7. Enhanced User Interface:
    - Develop a user-friendly graphical interface providing control over settings, playback, and viewing of recorded footage.
8. Machine Learning for Motion Pattern Analysis:
    - Employ machine learning algorithms to analyse recorded footage for pattern recognition, anomaly detection, or predictive modelling based on observed motion behaviours.
9. Optimized Performance:
    - Focus on optimizing code for faster processing and real-time performance, especially for high-resolution video feeds or computationally intensive operations.
10. Compatibility and Portability:
    - Ensure compatibility across various platforms and devices, making it easily deployable and usable across different hardware configurations.

CONCLUSION

The Python script showcases real-time human motion detection using YOLOv3 and OpenCV, enabling video capture, identification of human presence, and recording of detected motion. With potential enhancements in analytics, multi-camera support, smart alerts, and improved interfaces, it holds promise for broader applications in security, analytics, and smart environments. Continued development could transform it into a versatile tool for comprehensive motion analysis and surveillance.

REFERENCES

- ❖ YouTube: https://www.youtube.com/
- ❖ Google: https://www.google.com/