**Motion tracking on video open-cv python**

# Contributed by:

Vaishnavi S

Angel Mary E

Mercy J

Department of Information Technology

**Motion Clip Recorder Project Documentation**

**1. Introduction**

**In an increasingly digital and interconnected world, the demand for efficient surveillance systems has grown rapidly. Traditional CCTV setups continuously record video, resulting in massive storage requirements and often useless footage. Modern solutions seek intelligent methods to selectively record only significant events.**

**This project, titled "Motion Clip Recorder", introduces an automated system that detects motion in a video feed and captures short video clips of these events. By employing Python and OpenCV, this solution balances automation, efficiency, and storage optimization.**

**The system is simple but powerful, capable of operating with minimal hardware (like a webcam) and has broad applications ranging from home security to industrial surveillance.**

**2. Problem Statement**

**Conventional surveillance systems face several limitations:**

**Continuous recording leads to excessive use of storage.**

**Manual effort is needed to sift through long footage to find moments of interest.**

**High cost due to sophisticated commercial surveillance equipment.**

**Lack of real-time notification or event-triggered storage in simpler systems.**

**This project addresses these issues by:**

**Detecting motion in real-time.**

**Saving only event-specific video clips.**

**Providing timestamped filenames for easy access.**

**Ensuring storage efficiency and reducing human monitoring effort.**

**3. Existing System**

**3.1 Continuous Video Recording**

**Many CCTV systems record 24/7, storing vast amounts of data. This approach, while thorough, is inefficient and expensive.**

**3.2 Manual Clip Extraction**

**Security personnel often need to manually extract footage relevant to incidents, which is labor-intensive and error-prone.**

**3.3 Motion Sensing Cameras**

**While some advanced IP cameras support motion detection, they are costly and depend on proprietary software.**

**4. Proposed System**

**The Motion Clip Recorder is a lightweight alternative using open-source tools. It:**

**Continuously analyzes frames from a webcam.**

**Detects motion using frame differencing.**

**Automatically saves a short video (e.g., 5 seconds) including a pre-buffer to ensure context is not lost.**

**Stores clips with timestamp-based filenames.**

**Operates in real-time with minimal latency.**

**Key Advantages:**

**Low hardware requirements.**

**Minimal human intervention.**

**Highly customizable (record duration, sensitivity, etc.).**

**Easy integration with existing systems.**

**5. Objectives**

**Automate motion detection using video feed.**

**Record and save only relevant motion events.**

**Reduce storage usage.**

**Improve the usability of surveillance systems for general users.**

**Deliver open-source, readable, and modifiable code.**

**6. Methodology**

**The methodology for implementing the motion clip recorder includes:**

**6.1 Video Capture**

**A video source (usually a webcam) is accessed using OpenCV.**

**Frames are resized and prepared for processing.**

**6.2 Preprocessing**

**Each frame is converted to grayscale to reduce complexity.**

**Gaussian blur is applied to remove minor noise.**

**6.3 Motion Detection**

**Frame differencing compares the current frame with the previous one**

**Thresholding highlights regions with significant change**

**Contours are extracted to identify moving objects.**

**6.4 Recording Clips**

**When motion is detected, a short clip is recorded.**

**The system keeps a buffer of frames before motion to ensure pre-motion context is included.**

**Each clip is saved with a timestamp in its filename**

**6.5 Storage**

**Clips are saved in a clips/ directory.**

**Video is stored in .avi format using OpenCV’s VideoWriter.**

**7. System Architecture**

**The system includes the following components:**

**Video Source (Camera/Webcam)**

**Frame Buffer (Deque for storing pre-motion frames)**

**Motion Detector (OpenCV frame differencing)**

**Clip Recorder (Records motion event video)**

**File Manager (Saves files with appropriate naming and directory structure)**

**8. Hardware & Software Requirements**

**Hardware:**

**Standard PC/Laptop**

**USB Webcam or built-in camera**

**Software:**

**Python 3.8+**

**OpenCV (cv2)**

**imutils**

**datetime module**

**OS module**

**Deque from collections**

**9. Implementation – Code Overview**

**import cv2**

**import imutils**

**from collections import deque**

**import os**

**from datetime import datetime**

**import time**

**# Configuration**

**FPS = 20**

**BUFFER\_SECONDS = 2**

**RECORD\_SECONDS = 5**

**# Setup**

**vs = cv2.VideoCapture(0)**

**time.sleep(2.0)**

**first\_frame = None**

**buffer = deque(maxlen=BUFFER\_SECONDS \* FPS)**

**recording = False**

**while True:**

**ret, frame = vs.read()**

**if not ret:**

**break**

**frame = imutils.resize(frame, width=640)**

**gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)**

**gray = cv2.GaussianBlur(gray, (21, 21), 0)**

**buffer.append(frame.copy())**

**if first\_frame is None:**

**first\_frame = gray**

**continue**

**frameDelta = cv2.absdiff(first\_frame, gray)**

**thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH\_BINARY)[1]**

**thresh = cv2.dilate(thresh, None, iterations=2)**

**contours = cv2.findContours(thresh.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)[0]**

**motion\_detected = False**

**for c in contours:**

**if cv2.contourArea(c) < 1000:**

**continue**

**motion\_detected = True**

**if motion\_detected and not recording:**

**recording = True**

**filename = f"clips/motion\_clip\_{datetime.now().strftime('%Y%m%d\_%H%M%S')}.avi"**

**fourcc = cv2.VideoWriter\_fourcc(\*'XVID')**

**out = cv2.VideoWriter(filename, fourcc, FPS, (640, 480))**

**for f in buffer:**

**out.write(f)**

**record\_start = time.time()**

**if recording:**

**out.write(frame)**

**if time.time() - record\_start > RECORD\_SECONDS:**

**out.release()**

**recording = False**

**key = cv2.waitKey(1)**

**if key == ord('q'):**

**break**

**vs.release()**

**cv2.destroyAllWindows()**

**10. Output**

**Saved video clips in the clips/ directory.**

**Each file is named like motion\_clip\_20250511\_153000.avi.**

**Each clip contains a 2-second buffer before motion + 5 seconds during/after motion.**

**Output preview window shows live feed with motion detection boxes.**

**11. Testing and Validation**

**Test Cases:**

**Test Case ID Input Expected Output**

**TC1 No motion No video recorded**

**TC2 Walking person in front cam Clip of ~5 seconds saved**

**TC3 Multiple motions within 10s Multiple clips saved with timestamps**

**TC4 Quick flash (e.g., light on) Ignores small, short changes**

**12. Applications**

**Home surveillance systems.**

**Office or industrial monitoring.**

**Wildlife monitoring and research.**

**Motion-triggered alert systems.**

**Energy-efficient smart cameras.**

**13. Limitations**

**Works best in well-lit environments.**

**May be triggered by irrelevant changes (e.g., shadows).**

**Frame loss may occur if system is underpowered.**

**Not optimized for very high-resolution inputs or low FPS.**

**14. Future Scope**

**Add email/phone notifications when motion is detected.**

**Integrate cloud upload of saved clips.**

**Use AI models to distinguish between humans, pets, and objects.**

**Enable configurable clip duration and frame resolution.**

**Deploy on Raspberry Pi for edge detection.**

**15. Conclusion**

**This project demonstrates a simple yet effective solution for motion-based video recording. It leverages open-source tools and basic algorithms to achieve a highly practical outcome: intelligent surveillance.**

**By saving only important footage, it significantly reduces the storage burden while maintaining effectiveness. The system is modular, efficient, and can be enhanced with further features based on user requirements.**