Comprehensive Project Plan: Enhancing School Safety and Security

Objective

Develop a comprehensive system that utilises multiple camera feeds to detect and notify school authorities of significant unwanted and unusual behaviour occurring on school premises, enhancing safety and security through real-time monitoring and alerting

---

Project Components and Workflow

1. Frame Aggregation

Objective

Capture and aggregate multiple frames from each camera feed into a single image that summarizes motion during a specific time window.

Implementation Details

- *Input*:
    - Camera feeds from a camera connected via a DVR.
    - User-defined parameters: fps (frames per second) and duration (in seconds).
- *Process*:
    - Frames are extracted from each camera feed at the specified fps.
    - Frames are resized and combined into a grid structure using a modular * RTSHandler and onvif_handler* class.
    - The combined image effectively captures motion from the camera feed for the specified time window.
    - DVR Feed Configuration and Processing Workflow:

```
- Configure the DVR with a duration and delay to retrieve feeds starting
from duration + delay seconds behind real-time.
- Aggregate frames into a single image using duration and user_fps.
- Maintain a consistent delay to prevent conflicts with DVR recordings.
- Adjust the delay to match processing time (aggregation, inference, output)
to stay synchronized.
- Ensure seamless integration of DVR feeds with the behavior detection
system.
```

- *Output*:
    - A single aggregated image ready for behavior analysis.

Tools

- OpenCV for frame extraction and resizing.
- Python's threading to ensure parallel frame extraction from all cameras.

---

2. Behavior Analysis Model

Objective

Analyze the aggregated image and assign a behavior score between 0 (normal) and 4 (severe misconduct).

Implementation Details

- *Input*:
  - Aggregated image from the Frame Aggregator.
- *Process*:
  - A machine learning model is trained on ethically sourced data to detect behavior patterns.
  - The model predicts a score based on the severity of behavior observed in the image.
- *Output*:
  - Behavior score (0-4).
  - Behavior category for interpretation (e.g., "Normal", "Minor Misconduct", "Severe Misconduct").

Tools

- PyTorch for model training and inference.
- A machine learning model (Resnet101) to classify image.

---

3. Trespassing Detection

Objective

Detect unauthorized motion on the premises after school hours to prevent trespassing incidents.

Implementation Details

- **Input:**
    - Live camera feeds during non-operational hours.
- **Process:**

    Background Subtraction

    - The script uses cv2.createBackgroundSubtractorMOG2 to differentiate moving objects (foreground) from the static background in video frames.
    - Parameters like _detectShadows, _varThreshold, and _history are tuned to account for environmental changes such as lighting variations while maintaining detection accuracy.

    Noise Reduction

    - The foreground mask is processed using cv2.morphologyEx with the cv2.MORPH_CLOSE operation.
    - This step removes small holes and fills gaps in the detected motion areas, ensuring a cleaner mask.

    Binary Thresholding

    - cv2.threshold converts the processed foreground mask into a binary image, highlighting moving regions with pixel values of 255 and suppressing the rest to 0.

    Contour Detection

    - The script uses cv2.findContours to identify the boundaries of detected motion regions in the binary mask.

    Motion Validation

    - For each contour, the largest one is selected using max(contours, key=cv2.contourArea).
    - If the contour area exceeds _countourThreshold, it is considered valid motion, indicating potential trespassing.

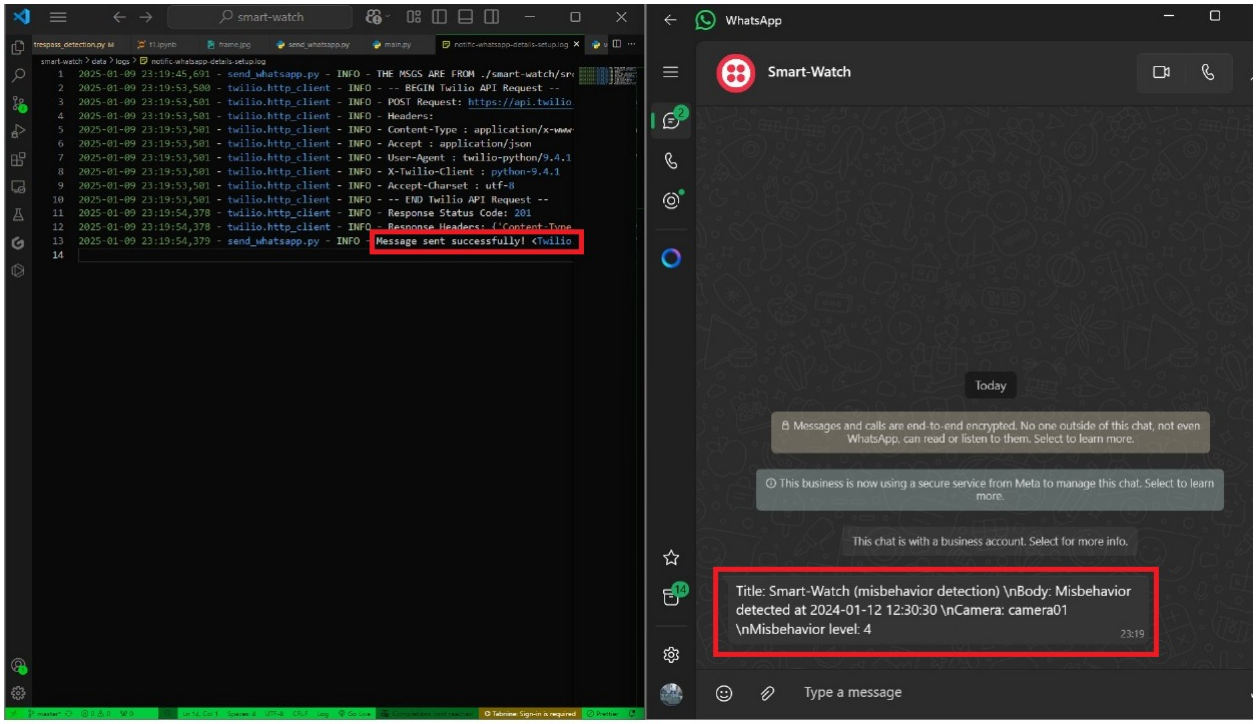
    Visual Feedback

    - A bounding box is drawn around the detected motion using cv2.rectangle.
    - "Motion Detected" is printed to the console for real-time feedback.

    Multi-Camera Support

- The function get_motions processes frames from multiple video streams simultaneously.
- It returns a dictionary with the motion detection status (True or False) for each camera.

- *Output*:
  - Real-time notification sent to authorities.



Tools

- OpenCV for motion detection.
- Custom rules for time-based operation (only active after school hours).

---

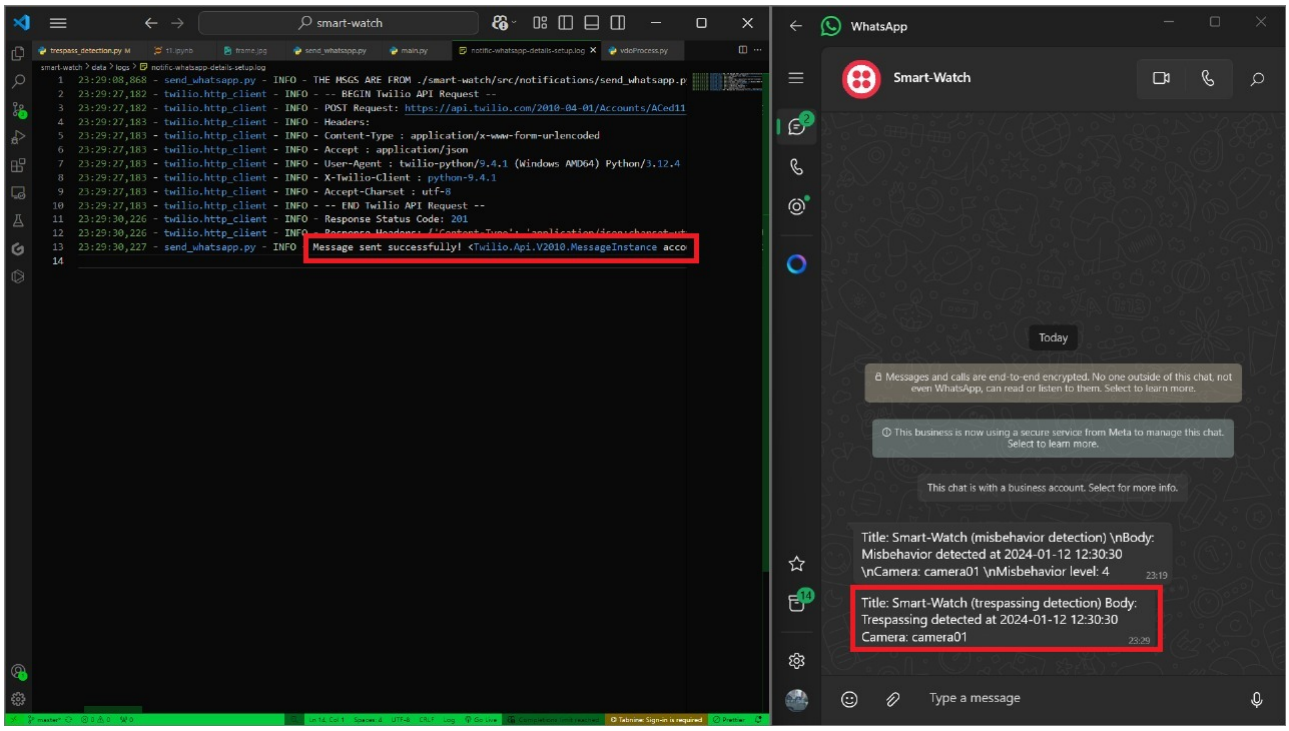
4. Notification System

Objective

Notify relevant authorities in real-time about trespassing or unacceptable behavior.

Implementation Details

- *Input*:
  - Alerts generated by the Behavior Analysis Model or Trespassing Detection Module.
- *Process*:
  - Configurable notification preferences (e.g., Email, WhatsApp, SMS, Push Notifications).
  - Notifications include details about the incident (e.g., camera ID, timestamp, behavior score).
- *Output*:
  - Real-time notifications sent to predefined users.



Tools

- Twilio or similar APIs for SMS and WhatsApp notifications.
- PushBullet or email libraries for additional channels.

Prototype Features

1. *Real-Time Monitoring*:
    - Continuous processing of live camera feeds.
    - Aggregation of frames for motion analysis.
2. *Behavior Analysis*:
    - AI model predicts the severity of behavior in real-time.
3. *Trespassing Alerts*:
    - Unauthorized motion detection triggers immediate alerts.
4. *Notification System*:
    - Configurable and reliable notification delivery through multiple channels.
5. *Resource Efficiency*:
    - Optimized to work with limited resources (e.g., offline DVRs, 720p/1080p cameras).

Testing and Deployment

1. *Initial Testing*:

- Test prototype on a simulated dataset to validate functionality.
- Evaluate accuracy of behavior predictions and motion detection.

2. *Campus Testing*:

- Deploy the system in a controlled environment at the college campus.
- Monitor system performance under real-world conditions.

3. *Performance Metrics*:

- Behavior Analysis Model Accuracy.
- Trespassing Detection Precision and Recall.
- Notification Delivery Reliability.

4. *Refinement*:

- Incorporate feedback from testing to improve features and performance.

Conclusion

Using multiple frames to create a single image simplifies video analysis while conserving essential behavior data, while capturing key motion patterns. The device integrates with DVRs and avoids recording conflicts through adjustable delays. With ResNet101, it assigns behavior scores, which help authorities prioritize responses to incidents.

In order to identify motion in real-time from camera feeds, the system relies on background subtraction for trespassing detection. Using pattern matching operations and validating detected contours against a size threshold, it flags only significant movements. With this approach, you can eliminate noise like small objects and find unauthorized motion accurately.