opencv-week2

1 Sources Consulted

- **OpenCV Documentation:**
  - Motion detection using background subtraction: `https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html`
  - Laplacian operator for blur detection: `https://docs.opencv.org/4.x/d5/db5/tutorial_laplace_operator.html`
  - Histogram analysis: `https://docs.opencv.org/4.x/d1/db7/tutorial_py_histogram_begins.html`
- **Community Resources:**
  - Stack Overflow threads on RTSP stream handling and tamper detection.
  - YouTube tutorials on real-time motion detection with OpenCV.
  - Blog posts on camera integrity checks (blur detection, laser interference).

2 Key Learnings and Insights

- **RTSP Stream Stability:** RTSP streams are prone to connection issues and require retries or buffering logic.
- **Motion Detection:**
  - Background subtraction (MOG2) works effectively for detecting moving objects.
  - Frame differencing with `cv2.absdiff` is simpler but sensitive to noise.
- **Real-Time Optimization:** Lowering frame resolution and skipping frames help maintain FPS close to real-time streaming.
- **Camera Tamper Detection:**
  - Laplacian variance thresholding helps detect blur.
  - Histogram uniformity can identify covered or laser-affected cameras.
  - A rule-based system (>75% pixels affected) ensures reliable integrity checks.

3 Challenges Faced

- **RTSP Handling:** Frequent "cannot open stream" errors required adding timeout checks.
- **Motion Detection Accuracy:** Small lighting changes triggered false positives.
- **Camera Integrity:** Differentiating between intentional covering vs. natural low-light conditions was non-trivial.
- **Performance:** Balancing detection accuracy and real-time FPS required experimentation with parameters.

4 Practice Attempts and Exercises

- Implemented a small script to test RTSP connectivity using OpenCV `VideoCapture`.
- Experimented with background subtraction and frame differencing for motion detection.
- Wrote test code for blur detection using Laplacian variance and verified with blurred images.
- Used color histograms to detect uniform frames (covered/laser conditions).
- Combined integrity checks into a real-time loop and displayed overlay warnings with `cv2.putText`.

5 Conclusions

- Real-time motion detection is achievable with optimized OpenCV pipelines.
- Camera integrity checks require combining multiple methods (blur + histogram).
- RTSP streams need robust error handling for practical deployment.
- This week's tasks improved understanding of OpenCV's `imgproc` and `videoio` modules, bridging theoretical learning with hands-on practice.