Project Proposal: Real-Time Object Tracking with Adaptive Correlation Filters (MOSSE)

Motivation (2-4 sentences): Tracking objects in video streams has many applications in surveillance, robotics, and augmented reality. The MOSSE (Minimum Output Sum of Squared Error) tracker, introduced by Bolme et al. (2010), offers a fast and robust correlation-filter approach that can be used for real-time performance. By implementing and evaluating MOSSE, we aim to deepen our understanding of correlation filters and deliver a working system that reliably tracks a selected object across challenging video sequences.

Milestones:

- **Week 1:** Environment setup, gather sample videos, implement video I/O and preprocessing pipeline.
- **Week 2:** Implement MOSSE filter initialization (training on the first frame) and basic tracking loop.
- **Week 3:** Add bounding-box overlay, implement filter update strategy, and logging of tracking results.
- **Week 4:** Optimize code for real-time performance (target ≥30 FPS), handle occlusions and scale changes.
- **Week 5:** Design evaluation suite: compute tracking accuracy metrics (IoU, center-error) on test videos.
- **Week 6:** Prepare demonstration video, write final report, and finalize code documentation.

Evaluation (2-4 sentences): We will test the tracker on a set of benchmarks and custom videos, measuring Intersection-over-Union (IoU) and center-location error for each frame. Success is defined as maintaining IoU ≥ 0.5 for at least 80% of frames in each sequence and achieving real-time performance (≥30 FPS) on standard hardware.

Resources:

- **Software:** Python, OpenCV, NumPy, Matplotlib.
- **Data:** OTB-2013 benchmark videos and custom recordings of a moving ball.
- **Hardware:** Google collab, laptops
- **References:** Bolme et al. "Visual Object Tracking using Adaptive Correlation Filters" (2010).

Group Contributions (3 members):

- **Alex Marzban (Preprocessing & Core Filter Implementation + Report - Motivation/Approach):**
    - Set up development environment and dependencies (Python, OpenCV, NumPy).
    - Implement video I/O and frame preprocessing pipeline (grayscale conversion, window cropping, cosine window).
    - Code MOSSE filter initialization (training on first frame) and implement online filter update (Equations 10–12).
    - Test preprocessing functions and filter update logic.
    - Draft the **Motivation**, **Approach** and **Implementation Details** sections of the project report.
- **Ayush Sharma (Visualization & Optimization + Report - Implementation Details/Performance):**
    - Develop bounding-box overlay and live visualization module (drawing tracked box, PSR display).
    - Integrate occlusion detection using PSR threshold and implement scale/illumination adaptation.
    - Profile and optimize code for real-time performance (FFT acceleration, vectorization, target ≥30 FPS).
    - Collect performance data (frame rates vs. window sizes) and generate plots.
    - Draft the **Real-Time Performance**, and **Results** subsections of the report.
- **Fahad Khan (Evaluation Suite & Reporting + Report - Results/Challenges):**
    - Design and implement evaluation suite: load benchmark videos (OTB-2013), run tracker, compute IoU and center-error per frame.
    - Aggregate results, produce confusion plots (PSR over time, success plots) and compile demonstration video.
    - Integrate all modules into a single runnable script and prepare code documentation and README.
    - Draft the **Discussion**, **Conclusions**, and **Challenges/Innovation** sections of the report.

Collaborative Tasks:

- All members participate in code reviews, integration testing, and final editing of the report to ensure consistency and completeness.
- Coordinate on milestone checkpoints and merge code branches through version control (Git).