Cp1

Murmani Akhaladze

TarnNished

1. Introduction

The goal of this Computational Project (CP1) was to extract and analyze the motion of objects in video sequences using numerical methods. The task involved identifying moving objects, tracking their trajectories, and computing the first four time derivatives: velocity, acceleration, jerk, and jounce. Additionally, clustering methods using different norms were applied to understand patterns in object motion.

To fully evaluate the method, two videos were selected: one where the method works well (a stationary-camera highway video), and one where it performs poorly (a dynamic F1 highlight video). The project was implemented twice: once using basic operations from scratch, and once using OpenCV and scikit-learn for more efficient and robust operations.

This report describes the model, approach, experiments, and conclusions.

2. Model and Approach

2.1 Pipeline Overview

The computational pipeline consists of several stages:

- **Preprocessing:** Each frame is converted to grayscale and smoothed using Gaussian blur to reduce noise.
- **Object Detection:** In the library version, background subtraction (MOG2) is used. In the scratch version, simple frame differencing is applied. Thresholding and morphology isolate moving objects.
- **Tracking:** Detected centroids are matched between consecutive frames using a nearest-neighbor rule. Lost tracks are removed after a set tolerance.
- **Trajectory Extraction:** Since the cars in the highway video move mostly vertically, motion is projected to the vertical axis:

$$s(t) = y(t).$$

- **Smoothing:** A Savitzky–Golay filter is applied to stabilize the trajectory before computing derivatives.
- **Derivative Computation:** Using finite differences:

$$v(t) = \frac{ds}{dt}, \quad a(t) = \frac{dv}{dt}, \quad j(t) = \frac{da}{dt}, \quad jo(t) = \frac{dj}{dt}.$$

- **Unit Conversion:** Pixel measurements can be converted to meters if the scene has a known reference. In the highway video, the lane width (3.7 m) corresponds

to approximately 120 pixels, giving:

$$scale = \frac{3.7}{120} m/px.$$

- **Clustering:** Each track is represented by a feature vector:

$$f = [\overline{v},\ \sigma(v),\ \overline{a},\ \sigma(a)],$$

and clustered using k-means to group objects by motion behavior.

3. Experiments

3.1 Video 1: Highway Traffic (Success Case)

The first video shows cars moving on a highway recorded by a stationary camera. This setting satisfies the assumptions of the model: the background is static, lighting is stable, and motion is mostly along a single dominant direction.

3.1.1 Object Tracking

The algorithm successfully detects and tracks multiple vehicles. One track near the selected Region of Interest (ROI) is used as the single-object analysis example.

3.1.2 Kinematic Analysis

The smoothed position, velocity, acceleration, jerk, and jounce are computed. Results show:

- Position increases smoothly over time.
- Velocity behaves realistically and correlates with car motion.
- Acceleration has moderate fluctuations, typical for real-world driving.
- Jerk and jounce are noisier due to the sensitivity of higher derivatives.

After applying pixel-to-meter scaling, the velocity and acceleration values match physically reasonable magnitudes for highway driving.

3.1.3 Clustering

The clustering step groups tracks into three distinct categories:

1. Fast cars close to the camera (high velocity).
2. Slower, distant cars (low pixel displacement).
3. Unstable or short tracks (higher variance).

This demonstrates meaningful separation based on motion patterns.

3.2 Video 2 (Failure Case)

The second video is captured on mobile phone. This video violates almost every assumption of the computational model:

- **Frequent cuts:** The scene changes every few seconds, so no continuous track can be maintained.
- **Camera motion:** Panning, zooming, and onboard camera views cause the background to move more than the cars.
- **Perspective changes:** Cars switch from close-up shots to far-away overhead angles.
- **Extremely high speed:** Cars move so fast that inter-frame displacement is large and unstable.

3.2.1 Observed Failure

As expected, the method performs poorly:

- Tracking resets constantly due to scene changes.
- Background subtraction identifies nearly the entire frame as "moving".
- Derivatives show extreme spikes and unrealistic values.
- Clustering becomes meaningless because each scene has different scale and motion.

3.2.2 Conclusion From Failure Case

This video effectively demonstrates the limitations of the method. Classical tracking and numerical derivatives require a static camera, stable viewpoint, and continuous trajectories. When these assumptions break, the results become unreliable.

4. Conclusion

This computational project successfully implemented a full pipeline for detecting, tracking, and analyzing moving objects. The highway video validated that the method works well under proper conditions: a stationary camera, consistent motion, and identifiable scale.

The failure-case F1 video showed that the approach breaks when assumptions are violated. Camera motion, rapid scene changes, and perspective variation create unstable tracks and meaningless kinematic estimates.

Overall, this project highlights both the power and limitations of classical computer vision combined with numerical differentiation. The approach is effective for controlled surveillance-like videos but unsuitable for dynamic, edited, or rapidly changing scenes.

presentation video is in readme.md