Task: Assignment 2 - Camera-Based Speedometer -Implementation and Analysis Documentation

Name: Kipchirchir Raphael

Neptun Code: LGL7CS

This project implements a camera-based speedometer pipeline. It utilizes **Homography** for Bird's-Eye View (BEV) transformation, **Dense Optical Flow** (Farneback) to calculate motion between frames in the BEV, and a **RANSAC-based method** to robustly estimate the ego-translation vector. This motion vector is then converted and smoothed to provide a stable vehicle speed in km/h.

Generated Videos:

The video results generated by the pipeline are hosted in my drive. Please find the links below for the required run modes:

Before Refinement:

- result_speedometer_half_1.mp4
- result_speedometer_part1.mp4

After Refinement:

- result_speedometer_test.mp4

GitHub Link:
https://github.com/Raphtildai/visual-speed-measurement

I. My Contributions (Kipchirchir Raphael)

I wrote the core implementation logic for the entire pipeline, including:

- **Pipeline Control (main.py):** Defined the execution modes (FULL, HALF1, TEST), parameter configuration (BEV size, scale, optical flow settings), and managed the checkpointing and final results saving (PKL, CSV).
- **Homography Calculation:** Implemented the functions for **Normalized DLT (Direct Linear Transformation)** and the normalization matrices (calculate_normalized_homography) for robust initial homography estimation.
- **Speed Estimation Core:**
  - Wrote the **RANSAC Translation function** (ransac_translation_improved) which uses the median for candidate estimation and the mean of final inliers

for refinement. This is crucial for filtering outliers caused by non-static objects (other cars, people).
    - Designed the **Exponential Moving Average (EMA)** smoothing classes for both the raw translation vector and the final speed value (speed_smoother, translation_smoother), which significantly enhances temporal stability.
- **Visualization and Utility:** Implemented the optimized frame stitching logic (stitch_two_images_optimized) and the final display composition (compose_visual_frame), including the drawing of the smoothed motion vector and speed diagnostics.
- **Optimization and Debugging:** Set the optimized optical flow parameters (flow_params), adjusted interpolation methods (Cubic for BEV), and implemented the color matching optimization (match_color_fast).

II. AI (ChatGPT and Grok AI) Contributions

Component	AI Assistance Provided
Code Structure & Refactoring	Simplified and optimized function structures for better readability and modularity, particularly within pipeline_with_world_coords.py.
Numerical Stability	Assisted in debugging and resolving subtle numerical issues, such as ensuring correct array slicing and scalar casting when interacting between NumPy and OpenCV (e.g., in match_color_fast).
Robustness & Estimation	Helped refine the RANSAC loop, specifically checking the logic for selecting the median as the initial estimate in ransac_translation_improved for outlier robustness.

Visualization Refinement	Suggested improvements for the compose_visual_frame function, particularly the logic for correctly scaling and centering the stitched image within the visualization panel.
Documentation	Provided assistance in generating comprehensive inline comments and docstrings.

AI was used as a pair-programming assistant to refine implementation details, ensure robustness, and improve code quality.

III. AI Prompts Used

The following are examples of prompts used during the development process:

- "Refactor my Python pipeline that combines image stitching and optical flow for better separation of concerns."
- "Help me debug an cv2.addWeighted error when trying to use NumPy arrays for color matching, why are scalar floats needed?"
- "Suggest a robust way to estimate a single translation vector from a dense optical flow field, considering background noise."
- "Implement an Exponential Moving Average filter class in Python and advise on suitable alpha values for strong and moderate smoothing."
- "Check my calculate_normalized_homography function for standard implementation errors."

IV. Conclusion

AI assistance was instrumental in accelerating the refinement, debugging, and quality assurance phases of the project. **All architectural decisions, parameter selection (flow, scale, EMA alphas), testing against the dataset, and core algorithmic implementations were performed and verified by me.**