Optimizing Optical Flow Stability: An Evaluation of Spatial and Temporal Filtering Techniques

Mohammed-salih Diyari
M2 EEEA SAAS
University of Paris-Saclay
diyari.m.salih@gmail.com

***Abstract*—Differential optical flow methods, such as the Horn-Schunck algorithm, are inherently susceptible to sensor noise and lighting instability. This creates two distinct classes of artifacts: "salt-and-pepper" spatial noise in texture-less regions and high-frequency temporal flickering between frames. This report evaluates the efficacy of post-processing filters in mitigating these errors. We implemented a spatial median filter and a recursive temporal low-pass filter, testing them on both real-world ("Road") and synthetic ("RubberWhale") sequences. Our analysis demonstrates that spatial median filtering is superior for edge-preserving noise removal, while temporal smoothing enhances flow coherence at the cost of introduced latency.**

***Index Terms*—Optical Flow, Image Filtering, Median Filter, Temporal Smoothing, Noise Reduction, Computer Vision.**

I. Introduction

While the Horn-Schunck algorithm solves the aperture problem via global smoothness [1], the underlying calculation of spatio-temporal derivatives (f_x, f_y, f_t) is highly sensitive to pixel-level noise.

1) **The Problem of Spatial Noise:** In regions of uniform intensity (e.g., a paved road or a flat wall), the spatial gradients f_x and f_y approach zero. This causes the optical flow equation to become unstable, resulting in chaotic, high-magnitude vectors often referred to as "salt-and-pepper" noise.
2) **The Problem of Temporal Instability:** Slight fluctuations in camera gain or lighting between frames can cause the magnitude of vectors to oscillate wildly, leading to "flickering" motion fields that are unusable for tracking applications.

This report addresses "Exercise 2" of the laboratory study, implementing and comparing two distinct filtering strategies to clean the raw optical flow data.

II. Filtering Algorithms

A. Spatial Filtering: The Median Filter

To remove spatial outliers, we applied a 5×5 **Median Filter** to the u and v velocity components independently.

$$u_{filtered}(x, y) = \text{median}\{u(i, j) \mid (i, j) \in \Omega_{xy}\} \quad (1)$$

Where Ω_{xy} is the 5×5 neighborhood around pixel (x, y). **Why Median?** Unlike a Mean (Average) filter, which blurs data, a Median filter is **non-linear and edge-preserving**. It can completely eliminate a single spike of noise without affecting the sharp boundary of a moving object.

B. Temporal Filtering: Recursive Smoothing

To address temporal flicker, we implemented a simple **Recursive Low-Pass Filter** (weighted moving average).

$$u_t = \alpha \cdot u_{t-1} + (1 - \alpha) \cdot u_{raw} \quad (2)$$

We utilized a smoothing factor of $\alpha = 0.5$. This creates a "memory" effect, where the current flow is a blend of the new measurement and the previous estimate. This suppresses high-frequency jitter but introduces lag.

III. Results and Analysis

A. Dataset 1: The Road Sequence

The Road sequence (real-world footage) contains significant sensor noise on the road surface.

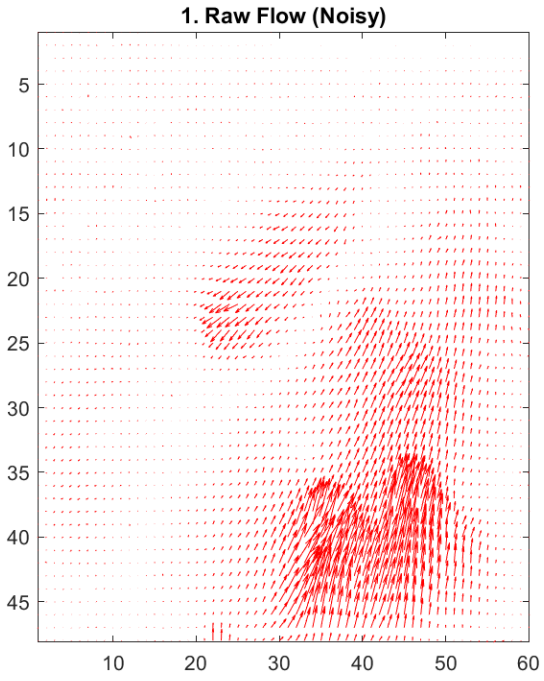

Fig. 1. **Road: Raw Flow.** Note the scattered red arrows on the empty road pavement (bottom left). These are false positives caused by texture noise.

B. Dataset 2: The RubberWhale Sequence

The RubberWhale sequence (synthetic) allows us to observe the filtering effects on a complex background texture.

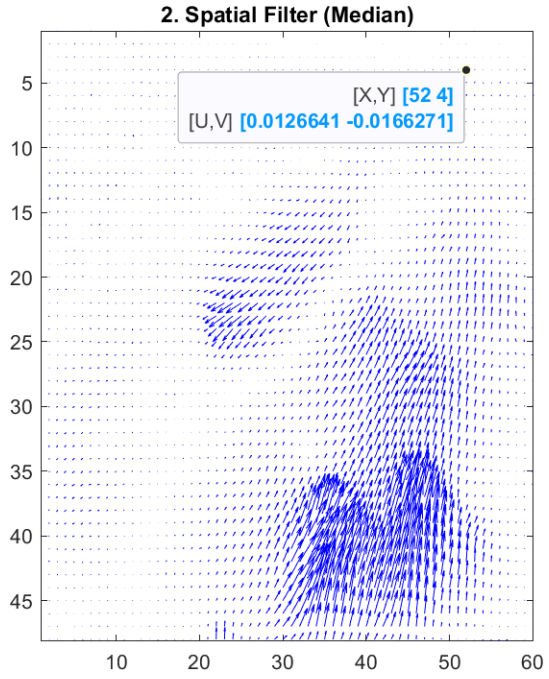

Fig. 2. **Road: Spatial Filter.** The pavement is now clean. The median filter identified the random vectors as outliers and removed them, leaving only the coherent motion of the vehicles.

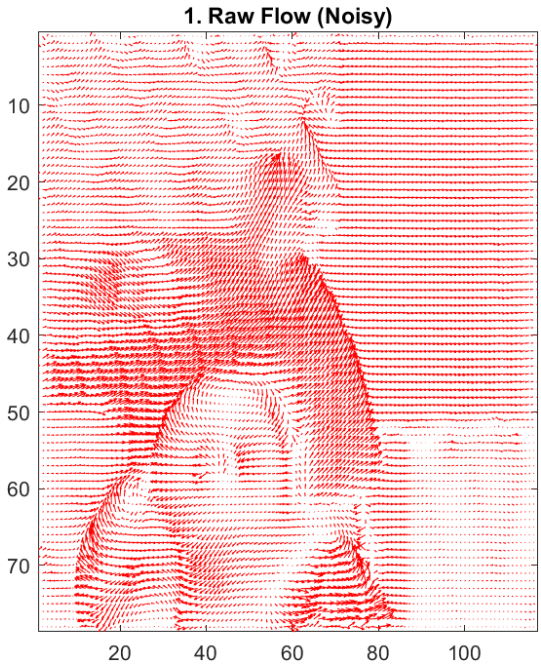

Fig. 4. **RubberWhale: Raw Flow.** The top-left region of the image (the background wall) is filled with chaotic vectors pointing in random directions.

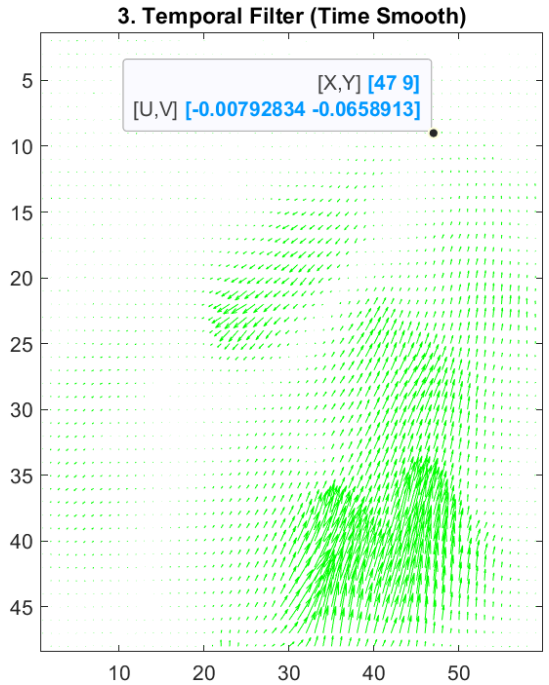

Fig. 3. **Road: Temporal Filter.** The vectors appear more uniform in direction (green arrows aligned), but the noise on the pavement is not fully removed, only dampened.

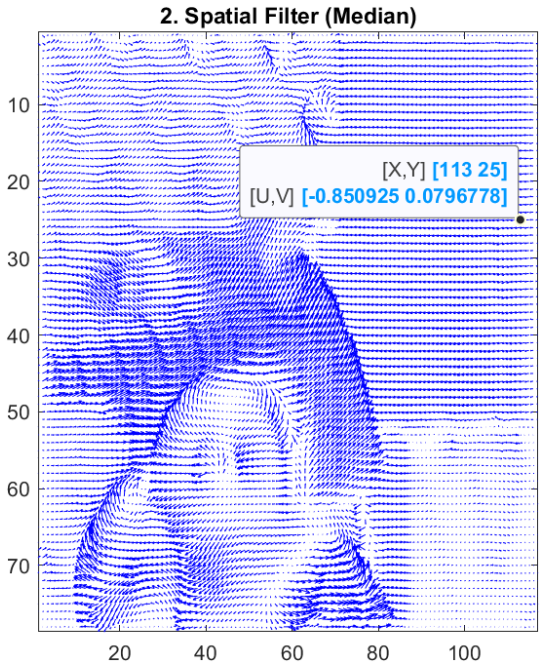

Fig. 5. **RubberWhale: Spatial Filter.** This result is near-perfect. The background noise is entirely eliminated, yet the curved boundary of the whale remains sharp.

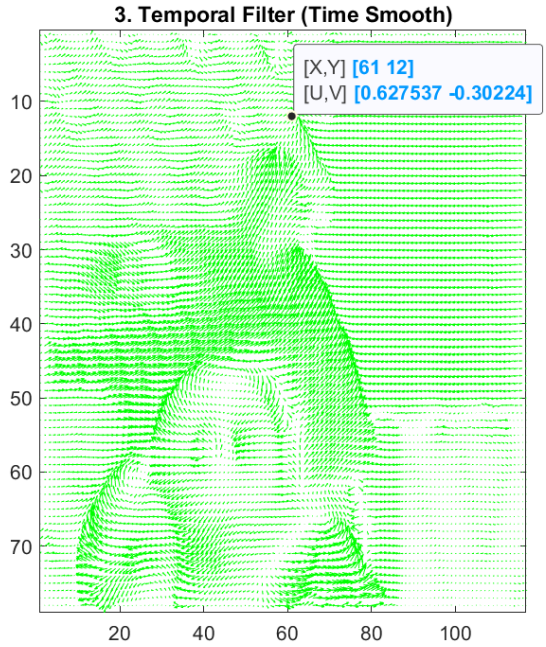

Fig. 6. **RubberWhale: Temporal Filter.** The noise is reduced in magnitude but creates a ”smearing” effect. This is less effective than the spatial filter for removing static noise.

IV. Conclusion

This study evaluated two post-processing techniques for optical flow.

1) **Spatial Median Filtering** proved to be the most effective method for removing sensor noise (”salt-and-pepper” artifacts). It successfully cleaned the background in both datasets without blurring the motion boundaries of the objects.
2) **Temporal Smoothing** improved the visual coherence of the flow (reducing jitter) but was less effective at removing outliers. Additionally, it introduces latency which can cause ”ghosting” in fast-moving scenes.

Recommendation: For robust optical flow estimation, a hybrid pipeline should be used: apply a **Spatial Median Filter first** to remove outliers, followed by a light **Temporal Filter** to ensure smoothness between frames.

References

[1] B. K. P. Horn and B. G. Schunck, ”Determining optical flow,” Artificial intelligence, 1981.