Computer Vision
Project 1

BENGRICHE Tidiane (ER-2037)
ZVONIMIR Pipic (ER-2059)

Contents

1 Introduction

Distributed Acoustic Sensing (DAS) is an advanced technology that transforms standard optical fiber cables into highly sensitive vibration sensors. By analyzing the phase shifts in backscattered laser light, DAS systems can detect subtle strain variations along the fiber caused by external factors such as traffic or seismic activity.

The objective of this project is to develop an algorithm to process DAS recordings collected from a fiber optic cable located at Jana Pawła II Street. The input data represents the strain rate of the fiber, where moving objects appear as slanted lines in a time-space 2D matrix.

This report presents the methodology and results for detecting and tracking these moving objects. The implementation utilizes Python-based image processing techniques, including frequency analysis and noise filtering, to isolate these signals and calculate the velocity of the detected objects.

2 Loading Dataset

This section loads the provided DAS (Distributed Acoustic Sensing) data for analysis. The dataset consists of '.npy' files containing acoustic measurements captured along a fiber optic cable.

Data Loading Process:

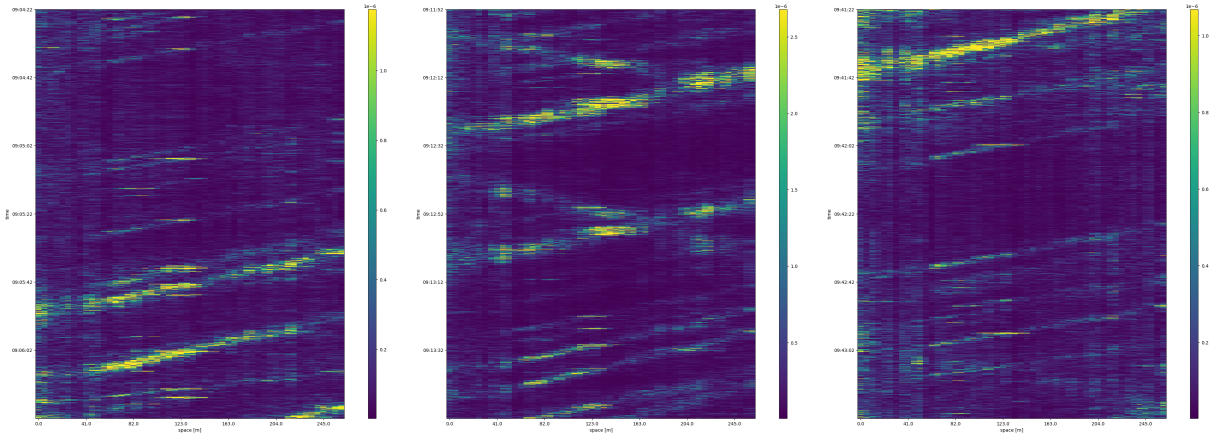
- **Spatial resolution** : $dx = 5.106$ meters between measurement points.
- **Temporal resolution** : $dt = 0.0016$ seconds between time samples.
- **Three time ranges** are analyzed:
  - Range 1: 09:04:22 - 09:06:12
  - Range 2: 09:11:52 - 09:13:42
  - Range 3: 09:41:22 - 09:43:12

Each dataset is loaded from multiple '.npy' files, concatenated, and organized into a pandas DataFrame with:

- **Rows** (index) : timestamps at dt intervals.
- **Columns** : spatial positions at dx intervals (in meters).
- **Values** : acoustic strain measurements

The data is then preprocessed by removing the mean and taking absolute values to enhance vehicle signatures. A space-time diagram is generated for each range, where vehicle trajectories appear as diagonal lines (slope = velocity).

The space-time diagrams clearly reveal vehicle trajectories as diagonal patterns. To better analyze these signals and prepare for trajectory extraction, we now examine their frequency content through FFT analysis.



(a) Loading Range 1 (b) Loading Range 2 (c) Loading Range 3

Figure 2.1: Comparison of Data Loading Ranges

3 Data Analysis

This section provides a statistical and visual analysis of the selected DAS time segments. For each analysis time, we compare the results across them. For clarity, the following names are used through the analysis:

- **Segment 1**: 09:04:22 – 09:06:12
- **Segment 2**: 09:11:52 – 09:13:42
- **Segment 3**: 09:41:22 – 09:43:12

3.1 Dataset Description

Each data set represents a two-minute DAS recording consisting of 75,000 time samples in 52 spatial channels. Table **??** summarizes the global statistics (mean, standard deviation, minimum, and maximum) for all three segments.

Table 1 presents the global statistics for all three segments, including the mean, standard deviation, minimum, and maximum strain-rate values. These statistics provide an initial indication of the noise level, the presence of vibration events, and the overall activity in each time segment.

Segment	Mean	Std	Min	Max
Segment 1	1.36×10^{-7}	3.82×10^{-7}	2.04×10^{-11}	7.18×10^{-5}
Segment 2	2.91×10^{-7}	6.37×10^{-7}	4.53×10^{-11}	4.24×10^{-5}
Segment 3	1.36×10^{-7}	2.82×10^{-7}	8.32×10^{-12}	3.25×10^{-5}

Table 1: Global statistics for the three DAS segments.

The **mean** strain-rate for all three segments is close to zero, which is expected for DAS systems.

The **standard deviation** reflects the overall vibration activity and noise level. Larger values mean that there was more noise detected. Segment 2 seems to have the highest noise levels, while Segment 3 seems to be the quietest.

The **minimum** values for all segments are extremely close to zero (10^{-11} to 10^{-12}), which corresponds to the noise floor where nothing measurable was recorded.

The **maximum** values reveal the strength of the largest vibration within each segment. Segment 1 contains the strongest peak (7.18×10^{-5}), indicating that the vehicle passed close by or was very loud.

Overall, the statistical differences suggest varying level of traffic or background noise between segments.

3.2 Histogram Analysis

The following figures show the logarithmic histograms of strain-rate values for the three analyzed segments. They represent the distribution of vibration amplitudes, where the x–axis represents the value and y-axis represents the number of occurrences. The number of occurrences is shown on the log scale because the majority of vibrations are just some small noises and do not represent the interesting part of the data for us.

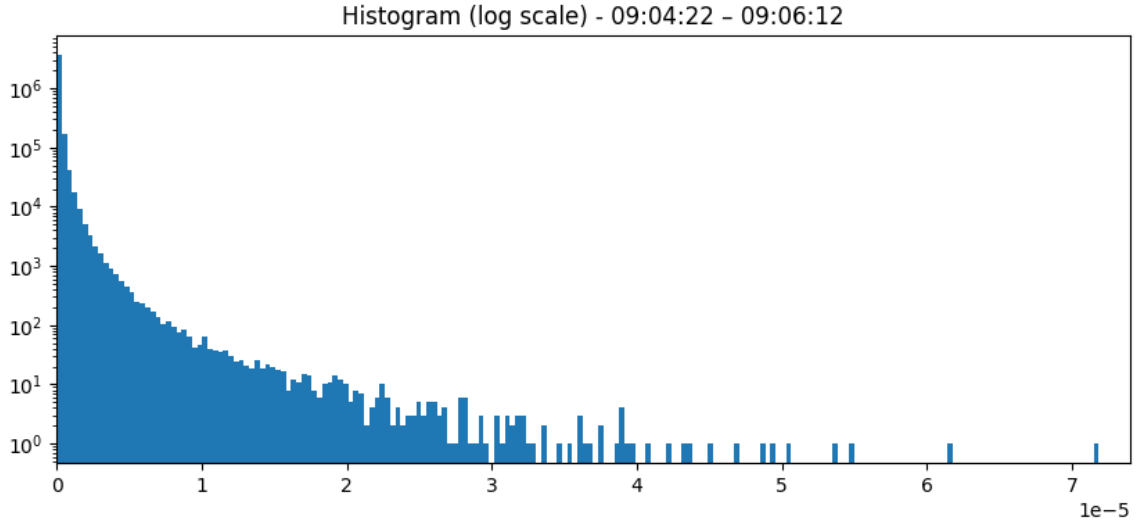


Figure 3.1: Log-scale histogram of strain-rate values for Segment 1.

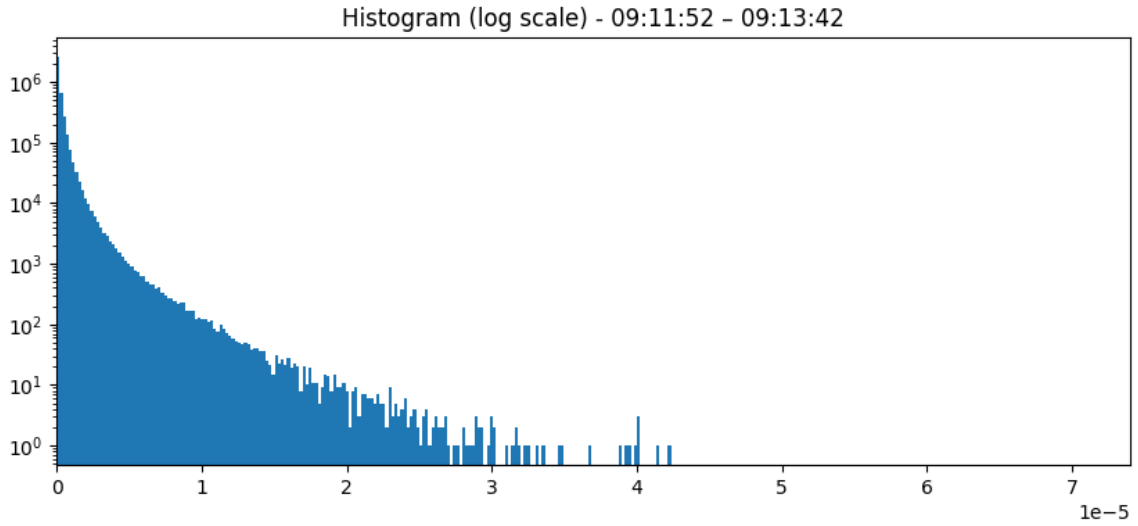


Figure 3.2: Log-scale histogram of strain-rate values for Segment 2.

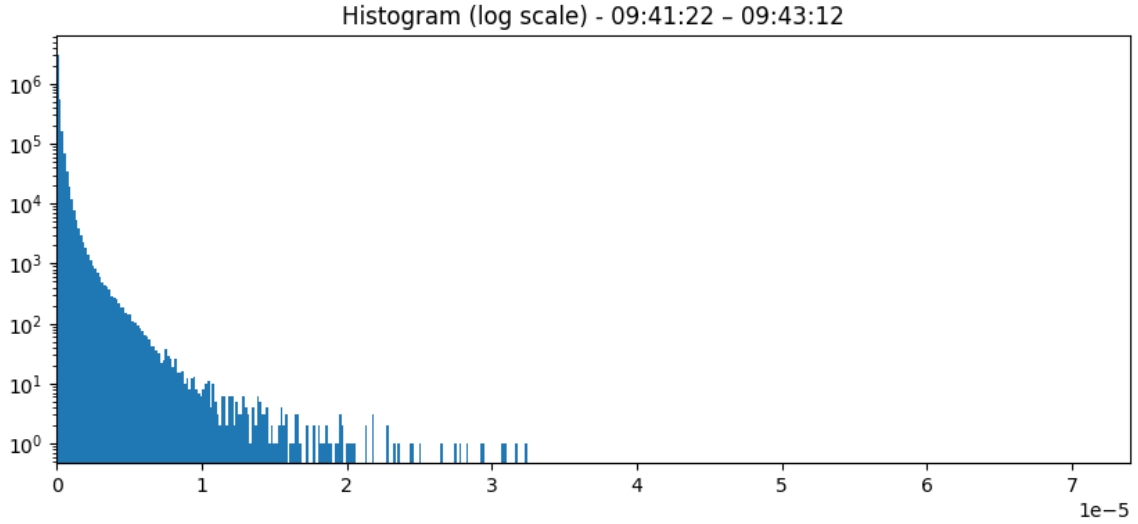


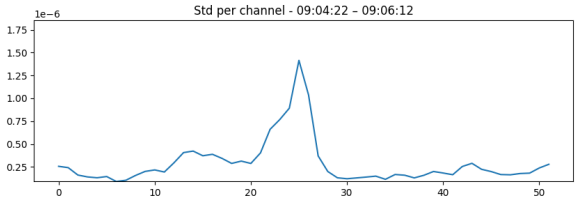
Figure 3.3: Log-scale histogram of strain-rate values for Segment 3.

In conclusion, Segment 1 contains several of the strongest vibration events, as indicated by its long histogram tail and the highest peak amplitude among all segments. Segment 3 exhibits the fewest high-amplitude events and shows the steepest decay in its histogram, confirming that it is the quietest segment. The Segment 2 lies between these two cases: it does not contain the strongest event, but has a broader distribution of mid-amplitude values, indicating a moderate level of vibration activity.

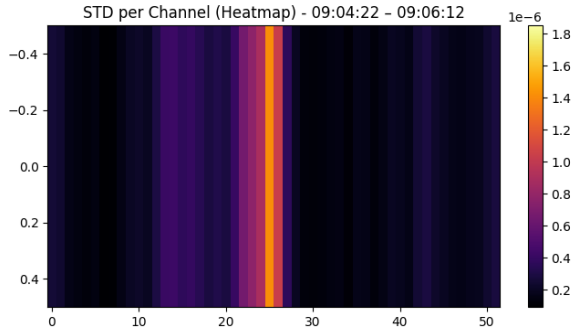
3.3 STD per Channel Analysis

The following figures show the standard deviation (STD) of the signal across the 52 fiber channels. The line plot shows the STD value for each channel, while the heatmap provides

a color-coded representation. Higher STD value indicates stronger vibration activity in the channel.

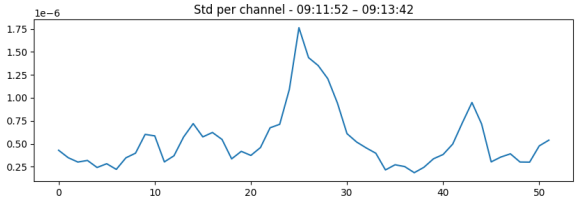


(a) STD per channel (Segment 1)

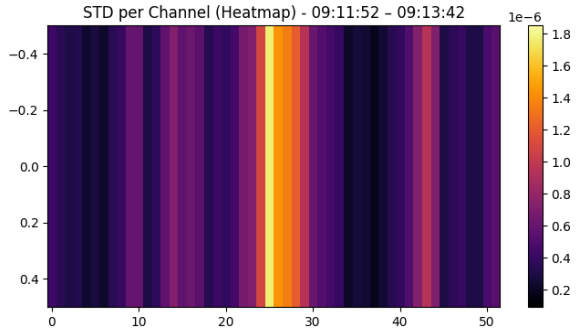


(b) STD heatmap (Segment 1)

Figure 3.4: Channel-wise STD for Segment 1.

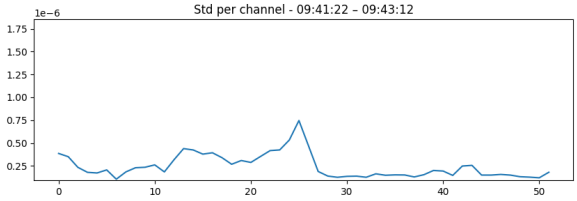


(a) STD per channel (Segment 2)

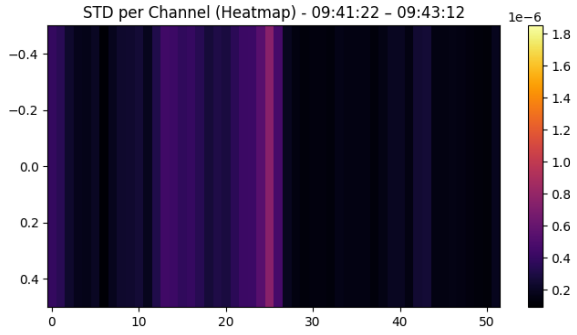


(b) STD heatmap (Segment 2)

Figure 3.5: Channel-wise STD for Segment 2.



(a) STD per channel (Segment 3)



(b) STD heatmap (Segment 3)

Figure 3.6: Channel-wise STD for Segment 3.

As shown in Figures 3.4–3.6, the middle channels in the approximate range 22–26 exhibit the highest STD values indicating that they are the most affected by the vibrations. Segment 2 has an additional peak near channels 42–45. Segment 1 has the highest peak, while Segment 3 once again shows as the quietest.

3.4 Raw Vibration Map

The figures below show the vibration maps. They show how the signal changes over time (x-axis) and across the fiber channels (y-axis,top to bottom). The colors show the strength: blue means little to no activity, and red means strong vibrations. Diagonal lines mean that something is moving across the fiber because vibrations appear in channels consecutively.

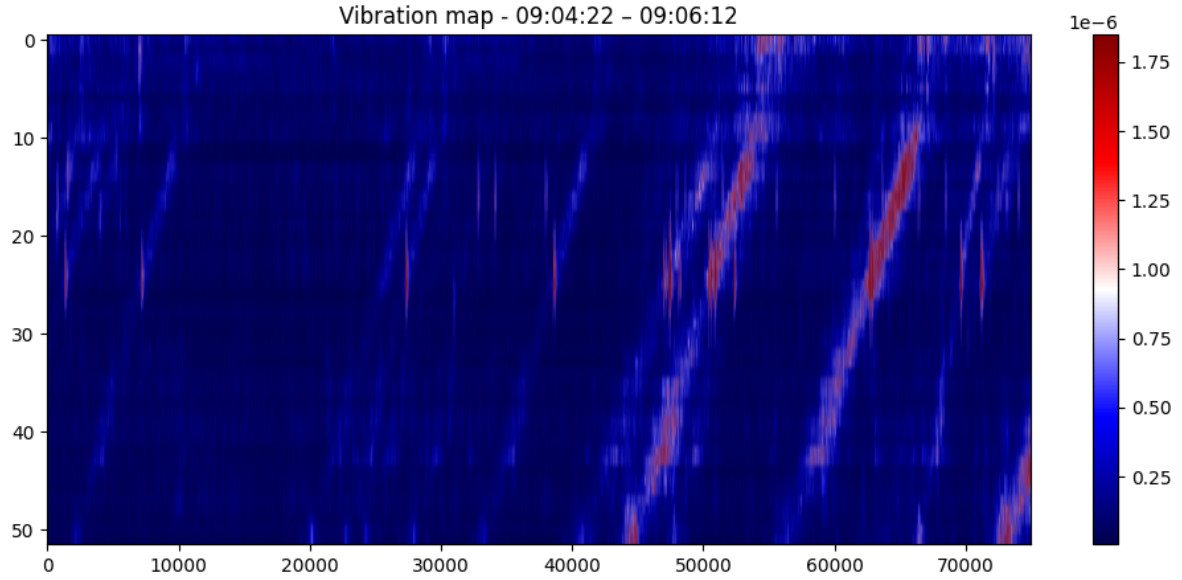


Figure 3.7: Vibration map for Segment 1.

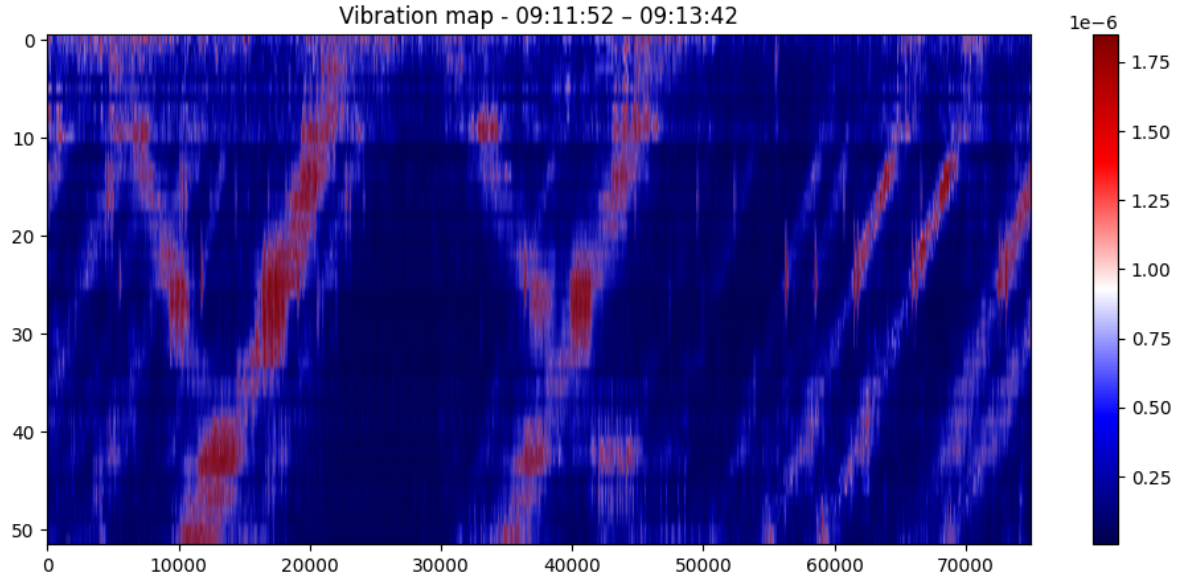


Figure 3.8: Vibration map for Segment 2.

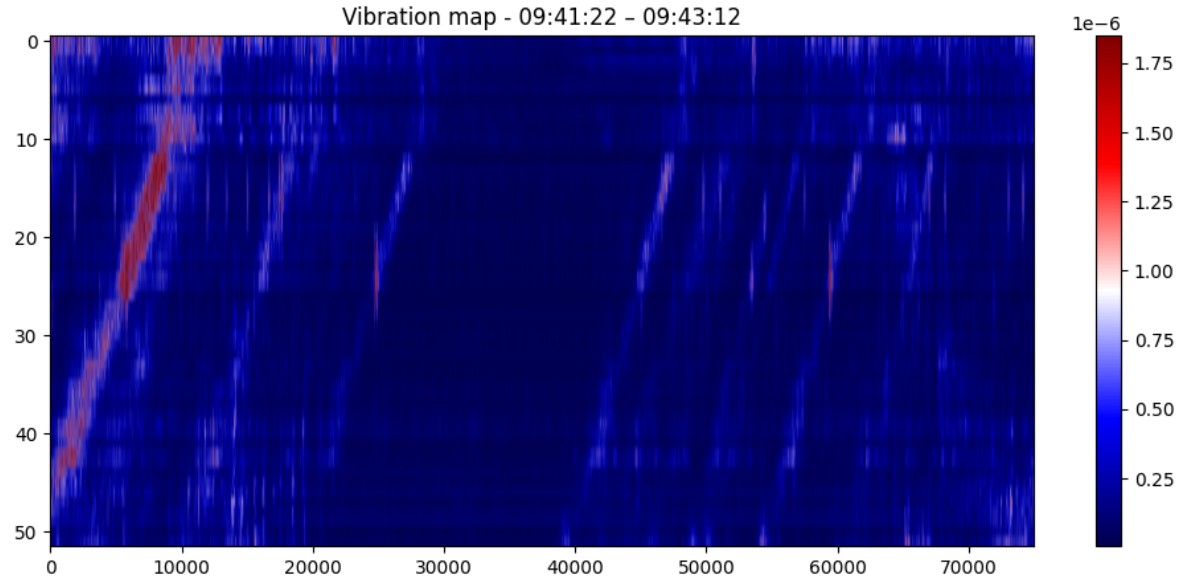


Figure 3.9: Vibration map for Segment 3.

Comparing the Figures 3.7–3.9, Segment 2 has the most activity, Segment 1 is second, and Segment 3 has the least, which corresponds to the previous data as well.

4 FTT

4.1 Overview of FFT

The Fast Fourier Transformation (FFT) is used to convert the time-domain DAS signal into the frequency domain. This allows us to see which frequencies are present in the vibration data. Low frequencies correspond to slow, smooth changes which are typical for vehicles, while high fequencies represnt sharp or noisy components.

Since each channel is independent, FFT is computed separately for each, and the average spectrum over all channels is used to summarize the frequency content of each segment.

4.2 Algorithm

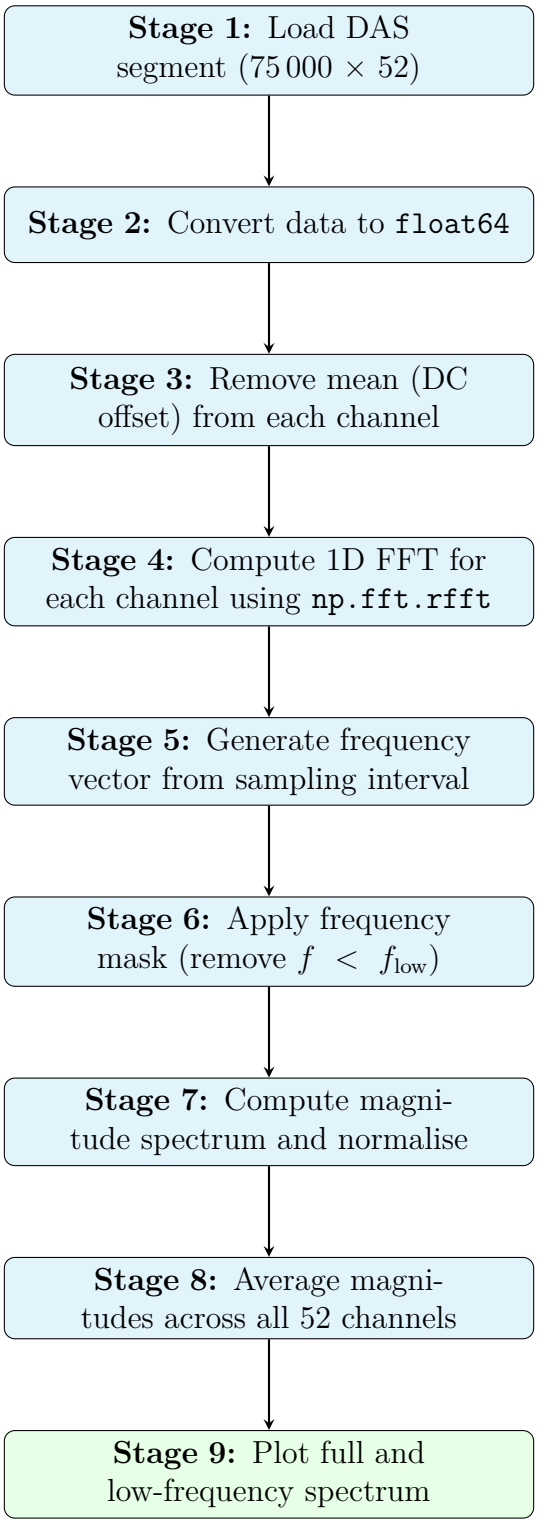


Figure 4.1: FFT processing pipeline used for analysing the DAS signal.

Stage 1: Load DAS Segment

The algorithm loads a two-minute DAS recording consisting of 75,000 time samples and 52 channels. Each channel contains an independent vibration signal.

Stage 2: Convert Data to Float64

The values are converted to float64 for precision.

Stage 3: Remove Mean (DC Offset)

Each channel has its own small offset. It is subtracted to align the signal around zero.

Stage 4: Compute FFT for Each Channel

A one-dimensional FFT (`np.fft.rfft`) is applied along the time axis for every channel. This transforms each time-domain signal into its corresponding frequency-domain representation.

The function `numpy.fft.rfft` computes the Fast Fourier Transform for real-valued signals. It takes a time-domain signal and breaks it down into the frequencies that make it up. Because the input contains only real numbers, the frequency spectrum is symmetric, so only the positive frequencies contain new information. The `rfft` function returns exactly this useful half of the spectrum, making the computation faster and reducing memory use.

The output of `rfft` is a set of complex values whose magnitudes show how strong each frequency is in the original signal.

Stage 5: Generate Frequency Vector

Based on the number of samples and the sampling interval dt, the algorithm computes the exact frequencies associated with each FFT bin.

Stage 6: Apply Frequency Mask

Frequencies below a chosen threshold f_{low} can be removed by setting their FFT coefficients to zero to eliminate unwanted frequencies in order to focus on the importat range.

Stage 7: Compute Magnitude Spectrum

The FFT output is complex-valued. The magnitude of each frequency component is computed using the absolute value of the complex spectrum and then normalized.

Stage 8: Average Across Channels

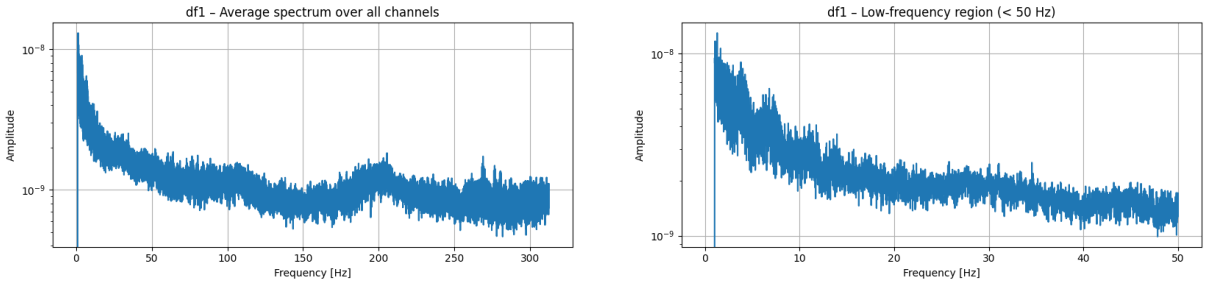
The magnitude spectra from all 52 channels are averaged to produce a single, clean spectrum. This reduces noise and highlights the dominant frequencies.

Stage 9: Plot the Spectrum

Two plots are generated: the full-frequency spectrum and a zoomed view of the low-frequency range below 50 Hz, where most vehicle-related vibrations occur.

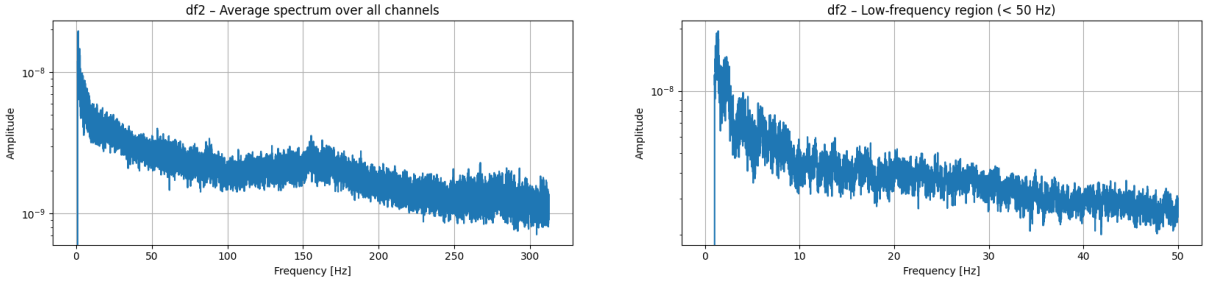
4.3 Graphs

The Figures bellow show FFT graphs displaying how much vibration energy is present at each frequency. The x-axis represents frequency, and the y-axis shows the amplitude of the vibration at that frequency. Higher amplitude means stronger vibrations. The second graph zooms in on frequencies lower than 50 Hz because vehicles fall under that range. Frequency of 0 Hz is also removed on both graph because it measures even the slowest, constant changes and is not needed.



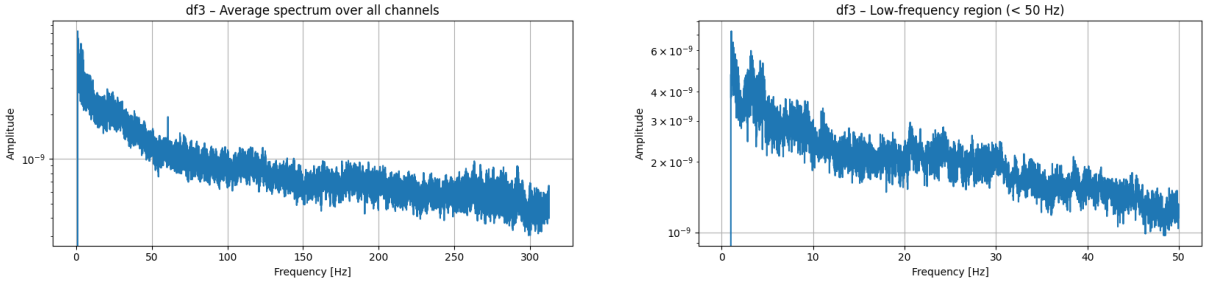
(a) Full spectrum (Segment 1) (b) Low-frequency region < 50 Hz (Segment 1)

Figure 4.2: Frequency spectrum of Segment 1.



(a) Full spectrum (Segment 2) (b) Low-frequency region < 50 Hz (Segment 2)

Figure 4.3: Frequency spectrum of Segment 2.



(a) Full spectrum (Segment 3) (b) Low-frequency region < 50 Hz (Segment 3)

Figure 4.4: Frequency spectrum of Segment 3.

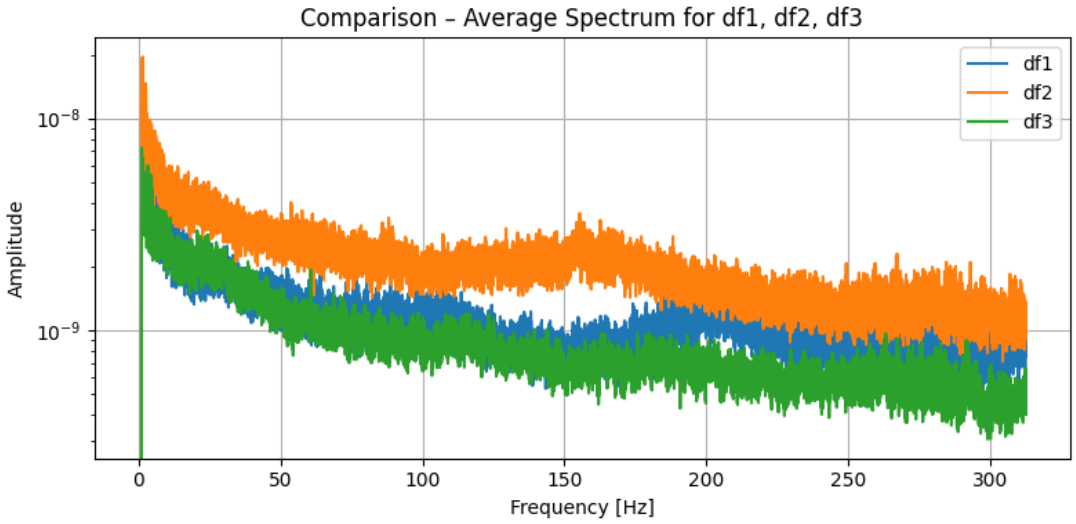


Figure 4.5: Comparison of average FFT spectra for all three segments.

The FFT analysis shows that most of the vibrations happen between 2–15 Hz range. Segment 2 has the highest amplitudes, while the Segments 1 and 3 have lower but similar amplitudes, suggesting that 2 has the most traffic.

5 Preprocessing

This section implements a comprehensive image processing pipeline to extract vehicle trajectories from space-time diagrams and calculate their velocities. The methodology consists of eight sequential stages:

5.1 Processing Pipeline Overview

The complete workflow processes each space-time diagram through the following stages:

5.2 Detailed Processing Steps

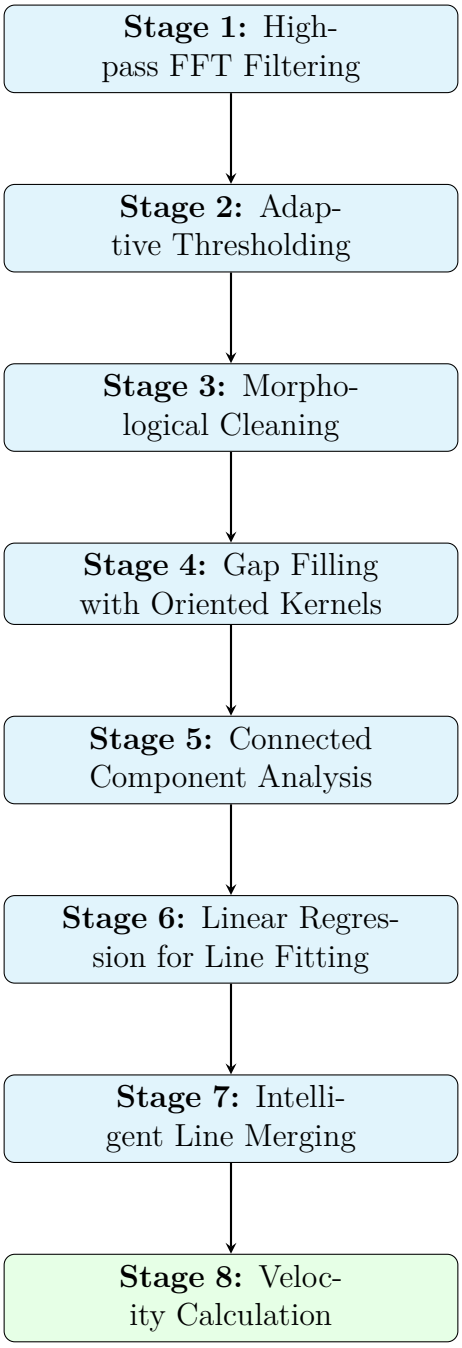


Figure 5.1: Algorithm Pipeline: From raw signal to velocity estimation.

5.2.1 Stage 1: High-pass FFT Filtering

The first step applies a frequency domain filter to isolate trajectory information:

- Compute the 2D Discrete Fourier Transform (DFT) for each RGB channel
- Shift the zero-frequency component to the center
- Apply a circular mask with radius $r = \min(H, W)/8$ to suppress low-frequency components
- Preserve high-frequency edges corresponding to vehicle trajectories
- Reconstruct the filtered image using inverse FFT

5.2.2 Stage 2: Adaptive Thresholding

After converting to grayscale, intensity-based thresholding is applied:

- Calculate threshold at the 87th percentile of non-zero pixels
- Adapt to varying signal strengths across time ranges
- Extract clear trajectory lines while rejecting ambient noise

5.2.3 Stage 3: Morphological Cleaning

Sequential morphological operations remove artifacts:

- **Closing** with small elliptical kernel (3×3) to connect nearby pixels
- **Opening** with large kernel (7×7, 2 iterations) to eliminate noise blobs
- **Final closing** to smooth boundaries

5.2.4 Stage 4: Gap Filling with Oriented Kernels

Fragmented trajectories are reconnected using specialized kernels:

- Diagonal kernel (5×5) at $+45$ř for positive slope lines
- Diagonal kernel (5×5) at -45ř for negative slope lines
- Horizontal kernel (15×3) to bridge small horizontal gaps
- Combine results with bitwise OR operations
- Apply final smoothing with elliptical kernel (9×9)

You can see the results of these 4 first steps in Figure 5.2, Figure 5.3 and Figure 5.4.

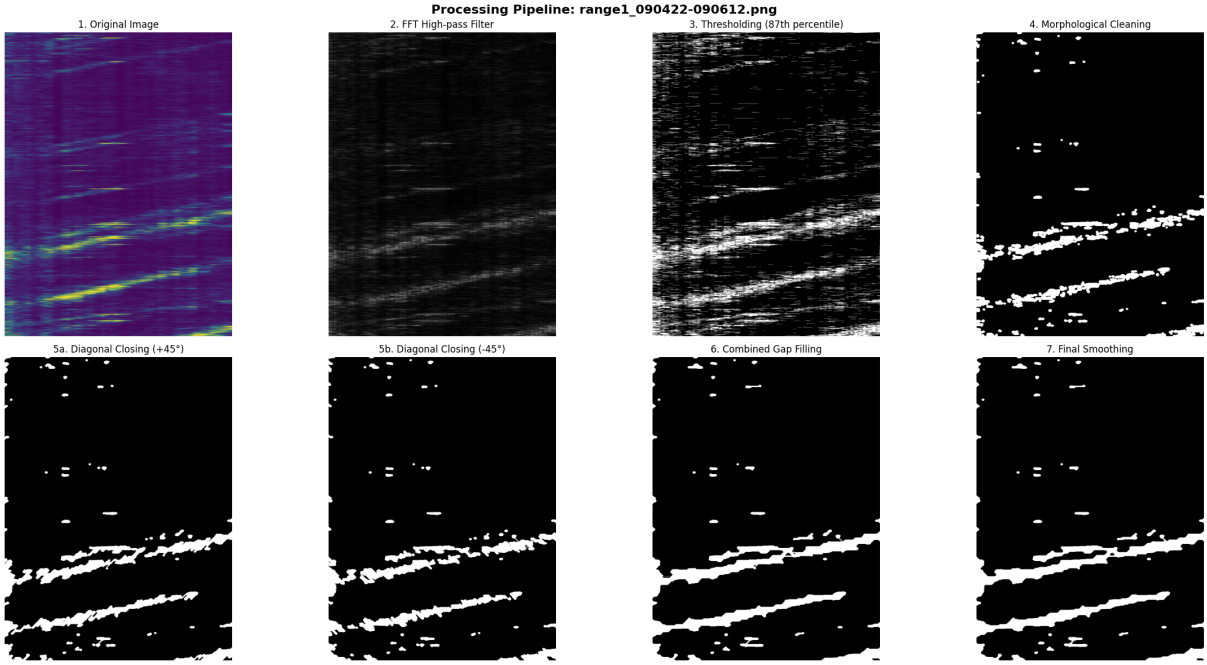


Figure 5.2: Steps for Range 1

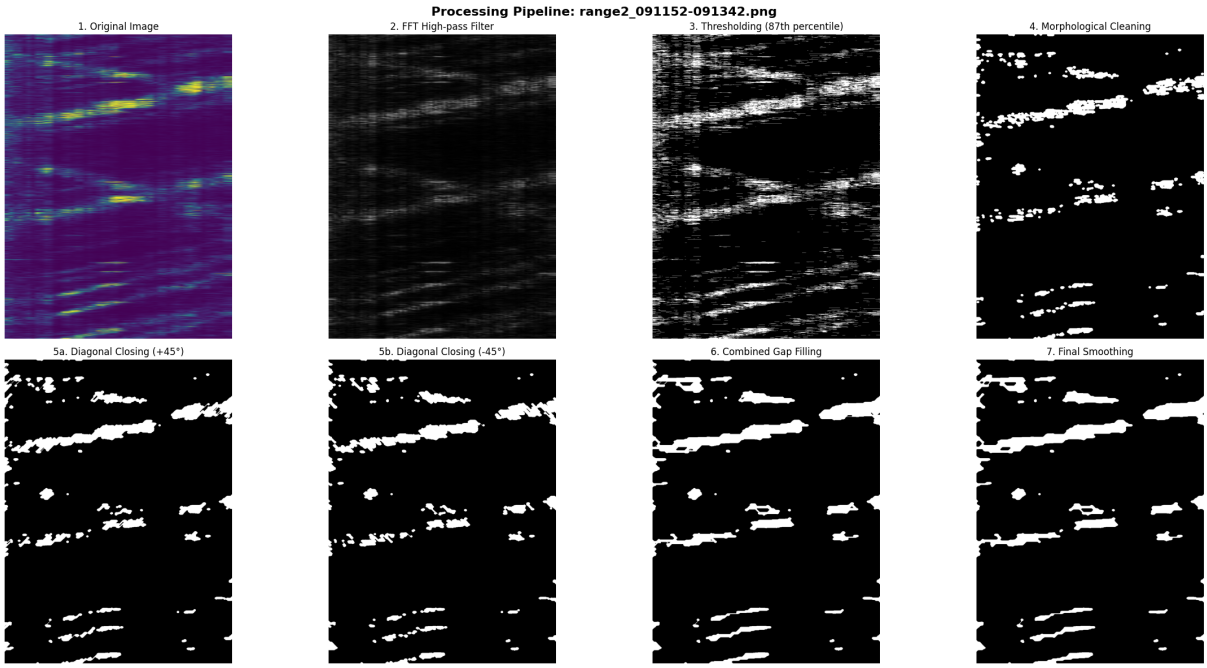


Figure 5.3: Steps for Range 2

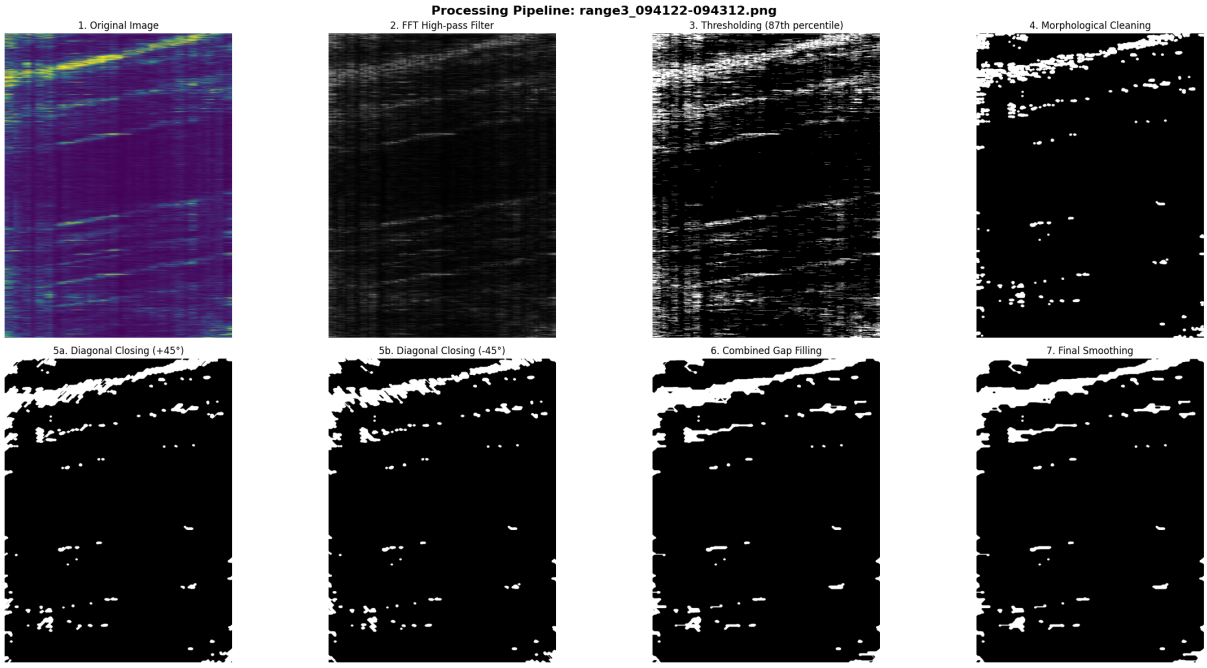


Figure 5.4: Steps for Range 3

5.2.5 Stage 5: Connected Component Analysis

Distinct regions are identified in the binary image:

- Apply 8-connectivity labeling
- Extract statistical properties: area, bounding box, centroid
- Filter components smaller than 1.2× median area as noise

5.2.6 Stage 6: Linear Regression for Line Fitting

Each valid component is modeled as a linear trajectory:

- Extract all pixel coordinates (x_i, y_i) from the component
- Fit linear model $y = mx + b$ using least-squares polynomial regression
- Obtain slope m and intercept b characterizing the trajectory

5.2.7 Stage 7: Intelligent Line Merging

Fragmented trajectories from the same vehicle are merged based on similarity:

- **Direction constraint**: Same direction with $m_1 \cdot m_2 > 0$
- **Slope similarity**: $|m_1 - m_2| < 0.3$
- **Horizontal proximity**: $\min(\Delta x) < 300$ pixels

- **Vertical proximity**: $|\Delta y| < 120$ pixels
- Iteratively merge closest matching pairs until convergence. We only merge big lines with smaller ones.

5.2.8 Stage 8: Velocity Calculation

Convert pixel-space trajectories to physical velocities:

- Use calibration: $dx = 5.106$ m/pixel (spatial), $dt = 0.0016$ s/pixel (temporal)
- Calculate velocity: $v = \frac{\Delta x \cdot dx}{\Delta y \cdot dt} \times 3.6$ [km/h]
- Filter outliers: reject $v < 0.1$ or $v > 200$ km/h
- Overlay detected lines and speeds on original images

6 Results

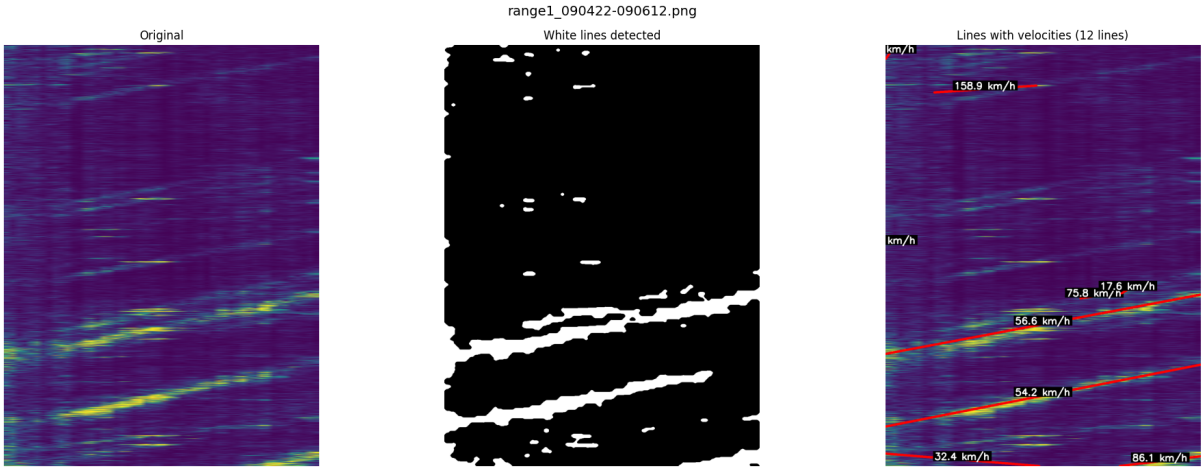


Figure 6.1: Results Range 1

Figure 6.2: Results Range 2

Figure 6.3: Results Range 3

7 Conclusion

In conclusion, Segment 2 has the most detected vehicles, while Segments 1 and 3 have similar levels of traffic, but smaller than Segment 2. The measured velocities are probably not 100% correct as some may be missing, some may be detect incorrectly, and most of them seem around 10–20 km/h higher than expected. Outliers can be created in multiple ways. The most probable one is noise form multiple vehicles merging together during gap filling or connecting lines. Noise fragment also have an impact and may be interpreted as very fast vehicles explaining some of the outliers.