# Project 1 Report: Moving Objects Detection in DAS Recordings

Authors:
Maria Musiał 156062
Martyna Stasiak 156071

## Introduction

The goal of this project was to detect moving objects in acoustic data collected from a Distributed Acoustic Sensing (DAS) system attached to a fiber optic cable at Jana Pawła II Street. Our primary objective was to identify the slanted lines in the data, representing moving vehicles like trams, trucks and cars moving along the street, and calculate their velocities.

## Dataset Description

The DAS data used in this project consists of time-series recordings, represented as 2D matrices, where rows denote time and columns represent spatial positions along the fiber optic cable.

Raw files characteristics:

- dx: 5.106500953873407 [m]
- dt: 0.0016 [s]
- file duration: 10s
- number of time samples in file: 6250
- file name format: HHMMSS
- files date: 2024-05-07

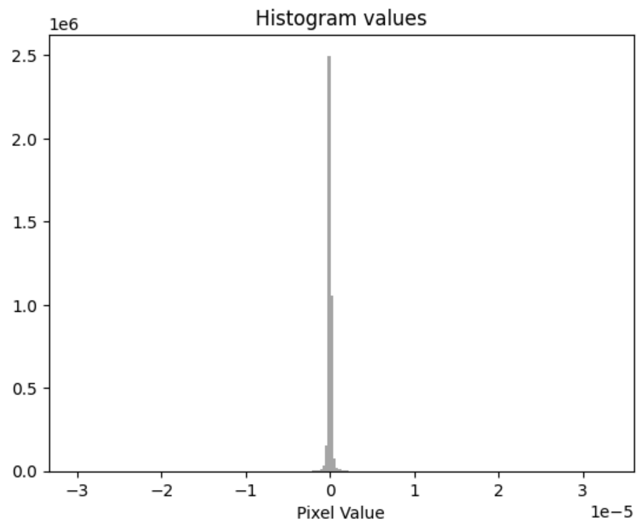
We handled the data in a 2m interval (12 files). Each column was around 5m and each row 0.0016s.

Our initial files

- 092022 - 092212 (Martyna)
- 090652 - 090842 (Maria)
- 091752 - 091942 (random)

# Analysis of the Dataset

15062:
Min value: -3.0223687645047903e-05,
max value: 3.285575803602114e-05,
Mean value: 1.2311088259941982e-10,
Standard deviation: 3.821909615453478e-07,



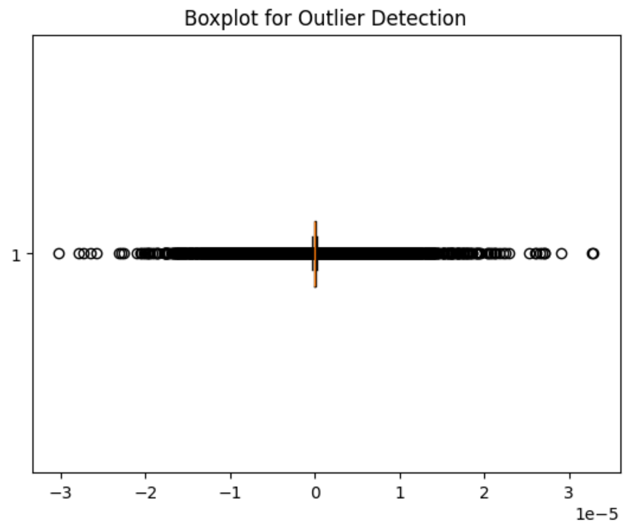
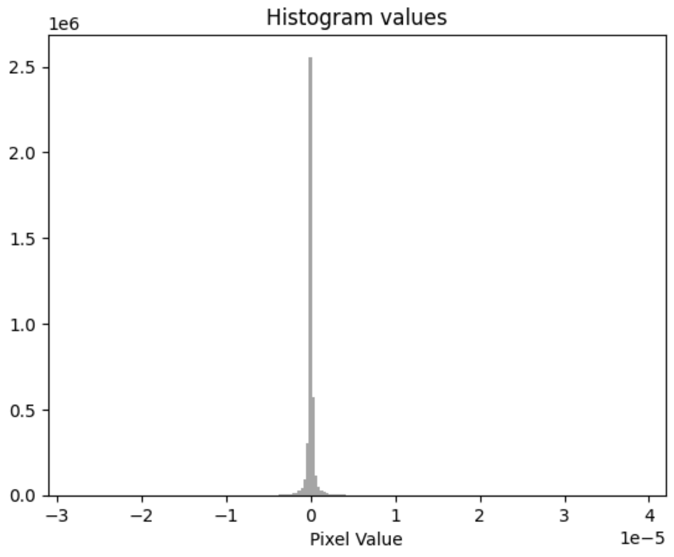


Figure 1: Histogram of values in raw file 62

Figure 2: Outlier detection 62

156071:

Min value: -2.7772102839662693e-05,
max value: 3.863127494696528e-05,
Mean value: -3.591482111708899e-10,
Standard deviation: 6.315133873613377e-07,



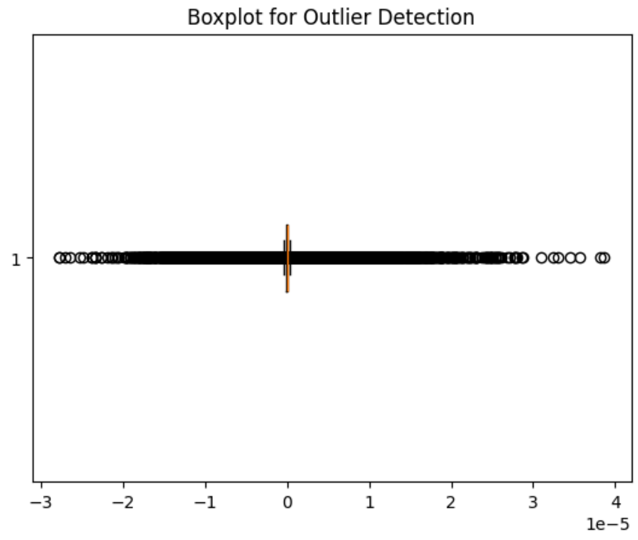


Figure 3: Histogram of values in raw file 71

Figure 4: Outlier detection 71

156072:

Min value: -6.239761569304392e-05,
max value: 4.1787250665947795e-05
Mean value: -1.0904931124766648e-10
Standard deviation: 6.565107923961477e-07

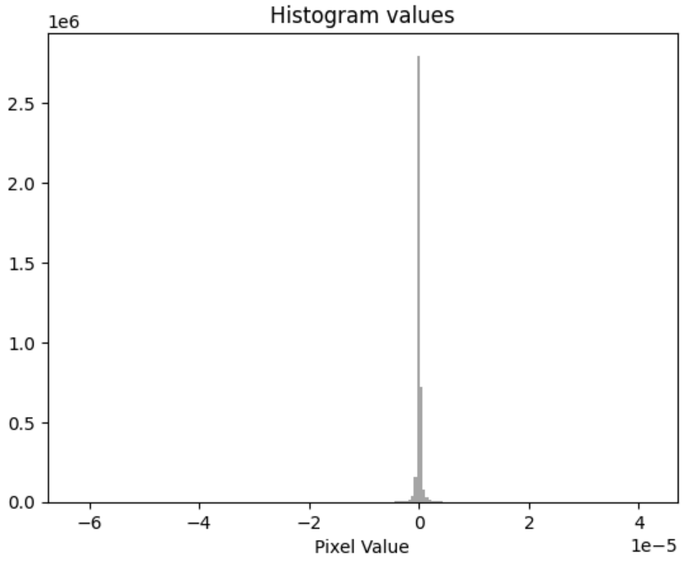


Figure 5: Histogram of values in raw file 72

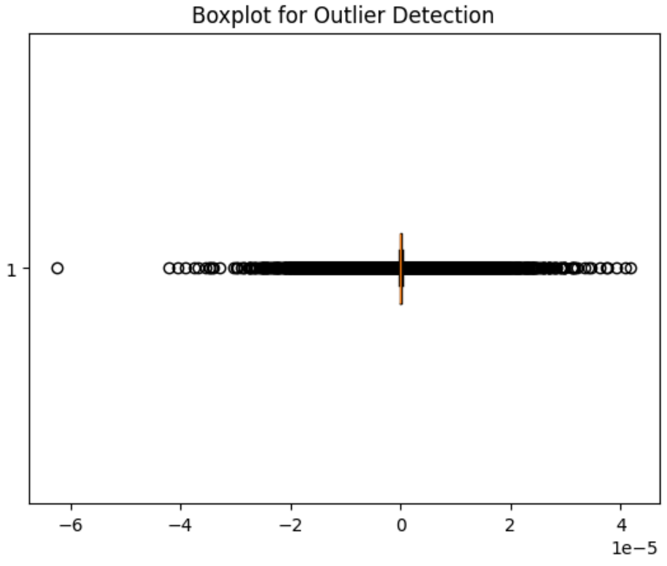


Figure6: Outliers in 72

First visualisation after taking out outliers, taking absolute value of the data and minmax normalization:

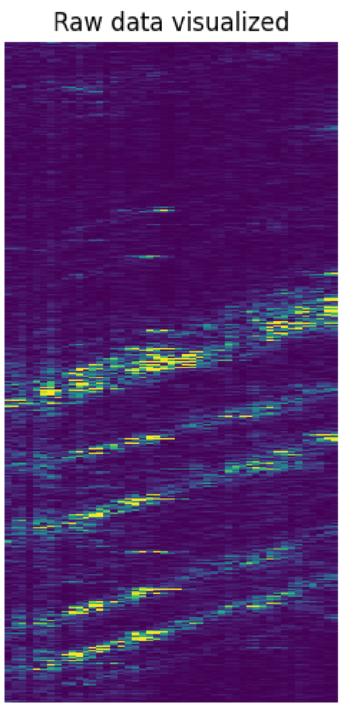


Figure 7: Visualisation of data

The initial analysis of the dataset containing our files revealed that it contained noise patterns, background signal variations, and prominent slanted lines corresponding to moving objects. These observations underscored the need for effective preprocessing to enhance the signal and isolate meaningful features.

## Algorithm

Preprocessing was a critical and most time consuming part of this project to ensure that the slanted lines representing moving objects could be detected accurately. It was a particularly challenging task since the files 090652 - 090842 contained a lot of noise; the noise in one part of the image was represented with the same frequency as the part of the lines in other parts of the image. Data also contained noise in specific columns, indicating zebra crossing or something of this nature. Because of that we needed some local denoising techniques.

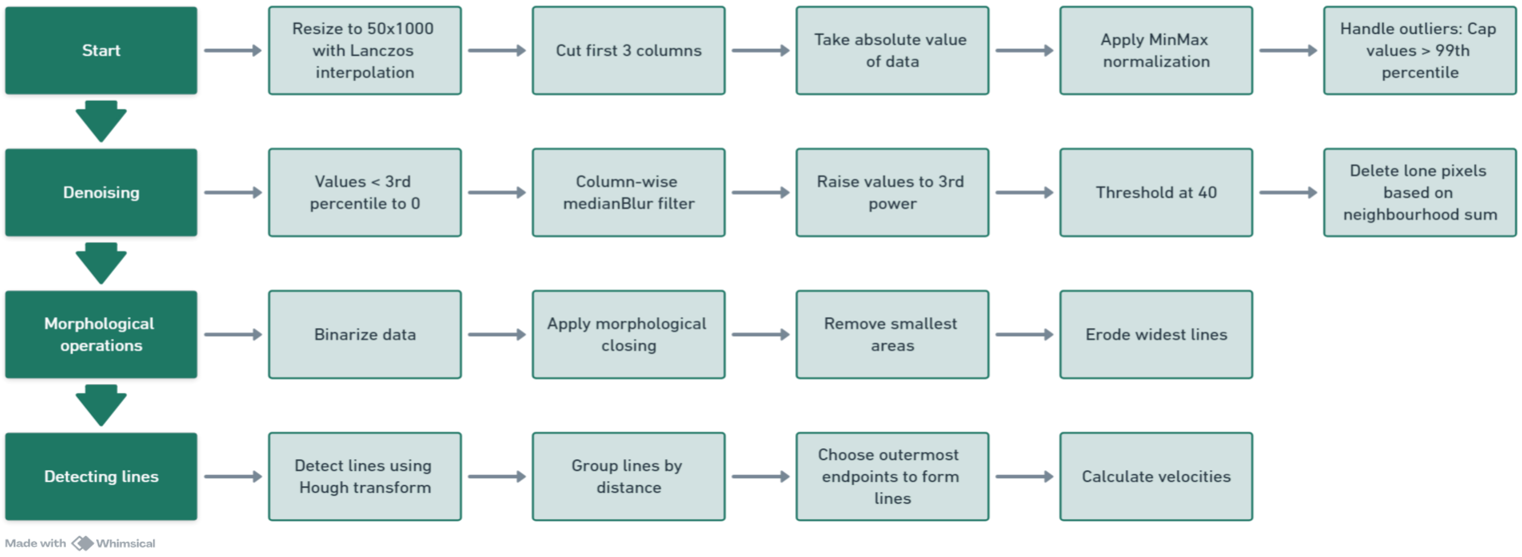

Start
Resize to 50x1000 with Lanczos interpolation
Cut first 3 columns
Take absolute value of data
Apply MinMax normalization
Handle outliers: Cap values > 99th percentile
Denoising
Values < 3rd percentile to 0
Column-wise medianBlur filter
Raise values to 3rd power
Threshold at 40
Delete lone pixels based on neighbourhood sum
Morphological operations
Binarize data
Apply morphological closing
Remove smallest areas
Erode widest lines
Detecting lines
Detect lines using Hough transform
Group lines by distance
Choose outermost endpoints to form lines
Calculate velocities
Made with Whimsical

Figure 8: Pipeline

Preprocessing:

1. Resizing to 50x1000 with lanczos interpolation. It preserved most of the significant features, while cutting out some of the noise. We needed the resizing to handle data better.
2. Cutting first 3 columns, as it had a lot of noise and data there wasn't significant for line detection
3. Absolute value of data, indicated by the nature of frequencies
4. MinMax normalization
5. Handling outliers: changing values over 99th percentile to value at 99th percentile

Denoising:

6. Changing values below 3rd percentile to 0
7. Column-wise medianBlur filter
8. Taking 3rd power of values (as our noise had value below 1) to suppress noise and enhance lines
9. Taking threshold at 40
10. Deleting lone pixels based on sum of pixels in neighbourhood

Morphological processing:

11. Binarization
12. Morphological closing on binarized image
13. Removing smallest areas (noise)
14. Eroding widest lines (for hough transform performance)

Detecting lines:

15. hough transform algorithm for detection of lines
16. merging into groups based on distance between lines. Choosing outermost right and outermost left endpoints from group to form final line
17. recalculating dx and dt based on initial resizing
18. calculating velocities

## Line selection

Once the data was preprocessed, we used the Hough Line Transform to detect the slanted lines. This technique identified line segments as (x1, y1, x2, y2), corresponding to the movement of objects. The lines were then grouped and merged using a custom algorithm. The merging logic considered the proximity and angle of the detected lines to group those that likely represented the same moving object. Within each group, the longest line closest to the group's midpoint was selected as the representative line. This ensured that the detected line accurately captured the extent and direction of the moving object.

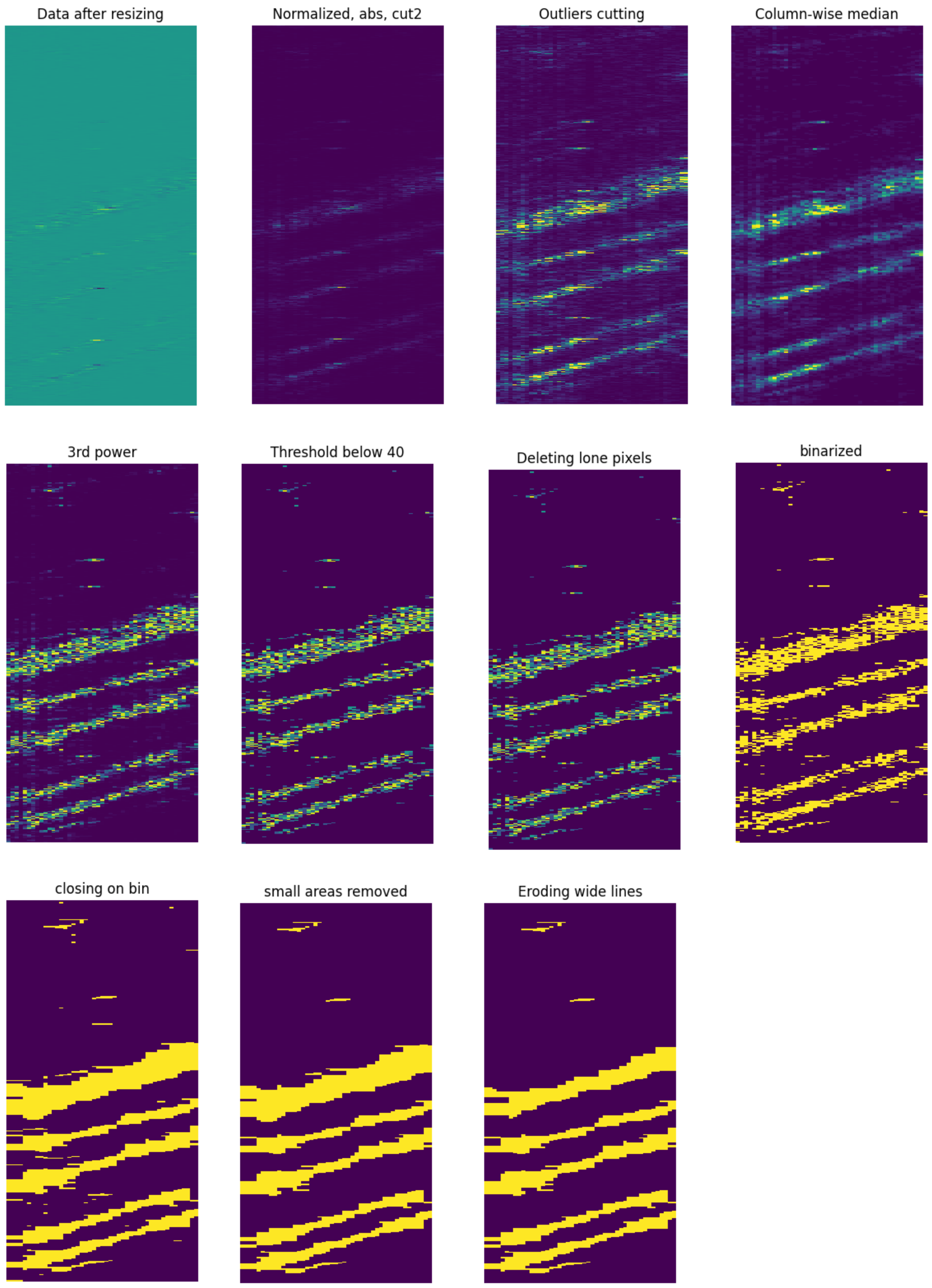
## Velocity

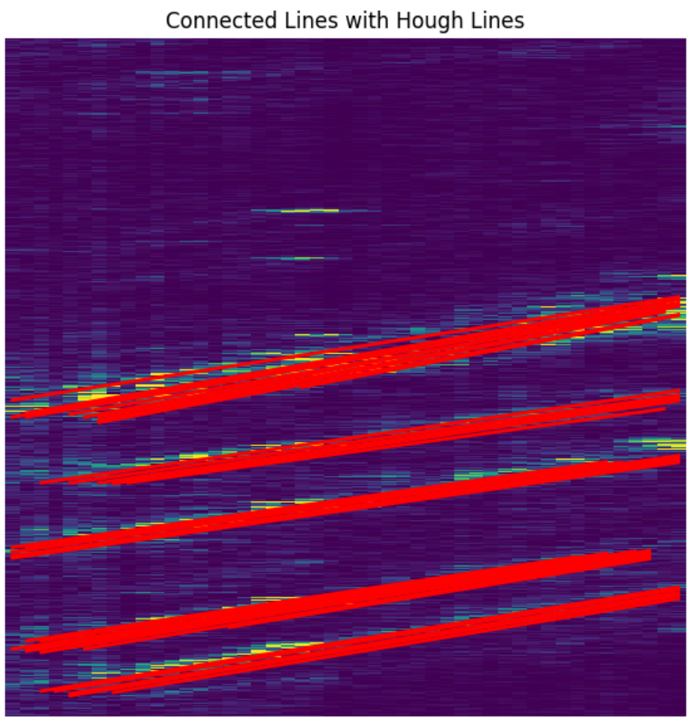
To calculate the velocity of the objects, the slope of the selected lines in the time-space domain was computed using the provided metadata. The velocity was derived as

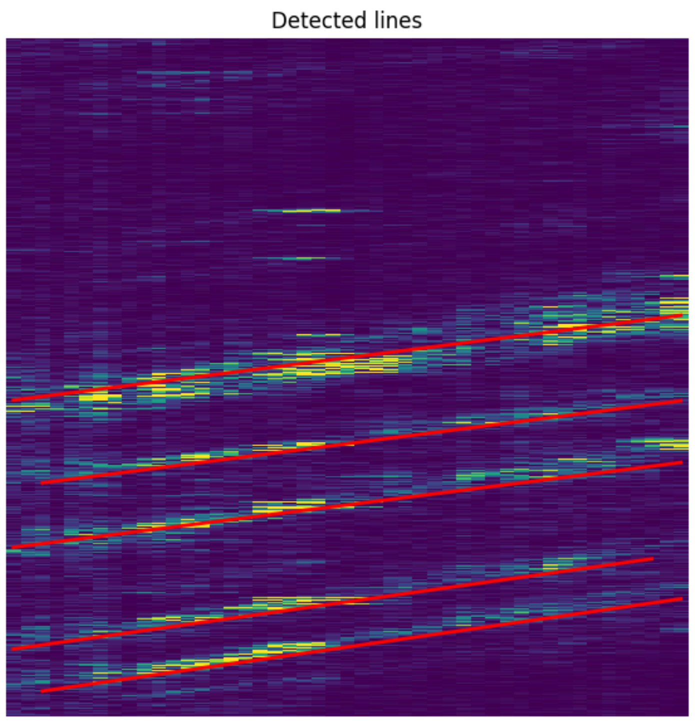
$v=\Delta x/ \Delta t$, where $\Delta x$ and $\Delta t$ are the spatial and temporal differences, respectively.

This representation was particularly useful for distinguishing between objects moving at a constant speed and those with varying speeds.

## Intermediate results:







## Results

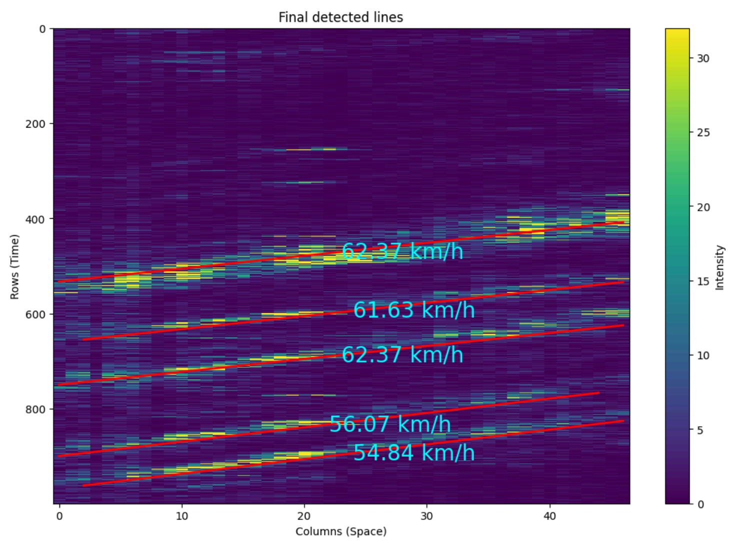


Figure 9: Results for 156071

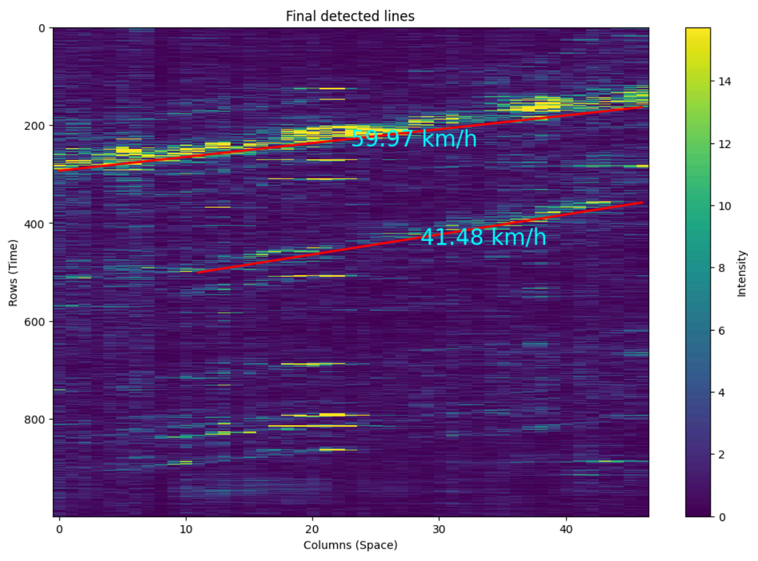


Figure 10: Results for 156062

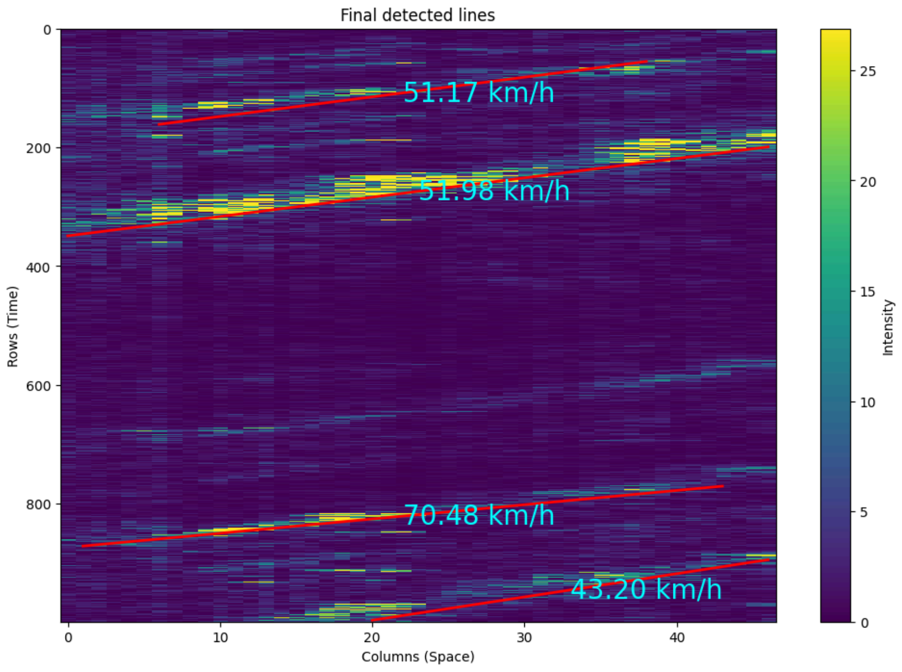


Figure 11: Results for 3rd set

## Tried solutions:

We tried multiple solutions of getting rid of noise, detecting lines and merging them. Algorithms tried include usage of:

- fourier transformation with low pass filtering,
- CLAHE,
- canny,
- nlm denoising
- pattern detection with convolution
- background subtraction using Gaussian blur
- skeletonization and thinning
- merging with DBSCAN
- using regions from skimage

All of the above were discarded in the end. NLM denoising did good job, same with Gaussian blur background subtraction. Yet, it was not the best solution, as we needed more contrast in our data. The range of values is narrow, so slight changes (eg. one value in threshold) gave big changes. It was challenging to make an algorithm that worked for all 3 datasets, as 156062 data could be processed more in focus on short lines with low contrast, but the solution we came up with didn't work with thick, close to each other lines in dataset 156072.

The merging was also done in different ways, with DBSCAN working well, but working on midpoints, which wasn't ideal, as we could have a line that was narrow in the middle, thicker in the ends, and DBSCAN didn't cluster them together.

## Conclusions

The project goal was achieved. The dimensions of the dataset were challenging, as morphology didn’t work well with the initial ~50x70000. Noise proved to be very problematic. Especially, because it was more of local nature. Using the frequency domain could yield even better results, but we couldn’t find a setting that would work better than what was achieved by morphology and arithmetics. In dataset 156062 the 2 lines on bottom are being detected with different settings in hough transformation, but merging part of the algorithm doesn’t work well with that. Possible improvements should be focused on better merging of lines and even better denoising.

### Sources of ideas:

- Previous laboratories
- OpenCV documentation
- skimage documentation
- StackOverflow
- Medium articles