Doppler Effect Based Sensor Fusion for Speed Estimation (April 2025)

Cameron Ritchie, Christopher Nikolik, Connor Johansson, Shivraj Singh Vishnoi, Shourrya Guha

***Abstract*—This paper presents a sensor fusion approach for vehicle speed estimation based on the acoustic Doppler effect. Synchronized smartphone recordings of a 2000 Hz tone emitted from a moving vehicle were collected by five microphones arranged along a 100-m trajectory. The Doppler-shifted frequency measurements and time delta data were fused using an Unscented Kalman Filter, whose parameters were finely tuned to track the nonlinear dynamics inherent in the Doppler signal. Due to limited sensor coverage in the direction perpendicular to the sensors, the process model was refined to focus on 1 dimensional motion. In addition, a Convolutional Neural Network was developed to process the spectrogram data and further enhance speed estimation accuracy. Experimental results, validated against in-vehicle speedometer readings, demonstrated that the proposed sensor fusion framework reliably estimates vehicle velocity under diverse operating conditions. This approach offers a practical alternative to traditional high-expense systems such as radar and LIDAR.**

***Index Terms*—Doppler effect, sensor fusion, Unscented Kalman Filter, Convolutional Neural Network, vehicle speed estimation, acoustic sensing, nonlinear filtering.**

I. Introduction

THE estimation of vehicle speeds plays a crucial role in modern traffic systems, autonomous navigation, and safety enforcement. Traditional techniques such as radar, LIDAR, and GPS, while effective, often have a high cost associated with them. These technologies also require specialized hardware, or face limitations in obstructed or signal degraded environments. This research paper aims at investigating a low cost and passive alternative to these traditionally used systems. This alternative utilizes the Doppler effect in acoustic signals to estimate the speed of vehicles using synchronized smartphones.

The experiment involves recording a high frequency tone emitted from a moving vehicle and aims to capture the frequency shift experienced at multiple fixed positions. These frequency shifts are then processed using sensor fusion technologies such as an Unscented Kalman Filter (UKF) and a Convolutional Neural Network (CNN), to estimate the vehicles' velocity. This approach provides an accessible and scalable solution for speed estimation in scenarios where traditional technologies are impractical. Some examples of these scenarios include urban canyons, tunnels or emergency vehicle detection.

This paper outlines the experimental setup, underlying theory, and signal processing pipelines. It also aims to evaluate the accuracy and feasibility of the proposed solution. The results demonstrate that acoustic Doppler based estimation, when combined with machine learning and sensor fusion, offers a promising direction for non-invasive and affordable speed measurement systems.

Researchers have done work for similar applications using other methods as below:

1) Speed Estimation On Moving Vehicle Based On Digital Image Processing [1]

Researchers in this paper have developed a method to utilize video and image data to estimate vehicle speeds. By extracting footage from CCTVs and applying methods like Gaussian Mixture Models (GMMs) for background subtraction, vehicles were detected across video frames. The Euclidean distance was used between vehicle positions in consecutive frames to estimate speed and gave pretty accurate results.

2) A Data-Driven Method for Vehicle Speed Estimation [2]

This research explores a data-driven method that utilizes fuzzy logic algorithms to estimate vehicle speed. By acquiring measurements from multiple sensors and applying fuzzy logic principles, the system can handle uncertainties and provide accurate speed estimations.

II. Background

A. Doppler Effect [3]

The Doppler Effect refers to the change in frequency or wavelength of a sound wave with respect to an observer that moves relative to the wave source.

When the source moves towards an observer:

- The sound waves are compressed.
- The observer perceives a higher pitch than emitted by the source.

When the source moves away from the observer:

- The sound waves are stretched.
- The observer perceives a lower pitch than emitted by the source.

Figure 1 below shows a visual representation of the phenomenon.

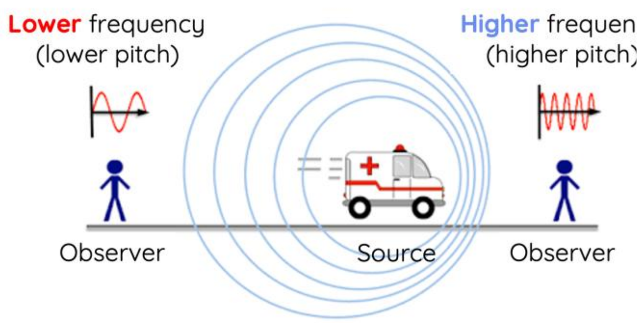

Figure 1: Doppler Effect [4]

Mathematical Expression for the doppler effect is as follows:

$$f_o = \left(\frac{v + v_o}{v + v_s \cos \theta}\right) f_s \quad (1)$$

Where:

- f_o: Observed frequency (what the receiver hears)
- f_s: Source frequency (what the speaker emits)
- v: Speed of sound in the medium
- v_o: Velocity of the observer
- v_s: Velocity of the source
- θ: Angle between the source's velocity vector and the line connecting the source to the observer

B. *Unscented Kalman Filter (UKF) [5]*

The Unscented Kalman filter is a variant of the Kalman filter to improve state estimation for non-linear systems.

The UKF does not suffer from the same issues with highly non-linear systems as the other alternative for non-linear systems, the Extended Kalman Filter (EKF) does.

The EKF, which linearizes non-linear models about a certain point, can have significant errors when modelling gaussian noise for highly non-linear dynamics. The first-order Taylor series linearization in EKF may produce large errors in the mean and covariance of the state estimate, potentially leading to divergence or suboptimal performance.

The UKF overcomes this by using Unscented Transform, a method that better captures the mean and covariance of a random variable undergoing a nonlinear transformation.

1) *Unscented Transform [6]*

The unscented transform approximates a probability distribution with a set of carefully chosen sample points, known as sigma points.

Given a state vector $x \in R^n$ with mean μ and covariance σ^2, the UKF generates $2n + 1$ sigma points $\tilde{x}^i$ as follows:

$$\tilde{x}^0 = \mu \quad (2)$$

$$\tilde{x}^i = \mu + \sqrt{1+\lambda} \cdot \sigma_i\,, \qquad i = 1, \dots, n \quad (3)$$

$$\tilde{x}^{n+i} = \mu - \sqrt{1+\lambda} \cdot \sigma_i\,, \qquad i = 1, \dots, n \quad (4)$$

where the scaling parameter λ is given by

$$\lambda = \alpha^2(n + \kappa) - n \quad (5)$$

These sigma points are assigned weights for the mean and covariance:

- Mean Weights:

$$w^0 = \frac{\lambda}{n+\lambda}\,, \qquad w^i = \frac{1}{2(n+\lambda)}, \quad i = 1, \dots, 2n \quad (6)$$

- Covariance Weights:

$$w_c^0 = w^0 + (1 - \alpha^2 + \beta)\,, \qquad w_c^i = w^i, \quad i = 1, \dots, 2n \quad (7)$$

C. *Convolutional Neural Network [7], [8]*

Convolutional Neural Networks (CNNs) are a class of deep learning models primarily used for 2D grid like data types like images or spectrograms. CNNs use convolutional layers to learn about visual features by applying various kinds of filters. These filters act as specialized neurons. This allows CNNs to detect patterns such as edges, shapes or textures. CNNs have become the foundation for many applications in image recognition and analysis due to their ability to extract high level features. Some of the common layers used in CNNs and used in this project are given below.

- Convolution
  - Putting the data through filters.
  - Activation functions such as ReLU (rectified linear unit) are used.
- Pooling
  - Down sampling to improve speed and emphasize features.
- Dense
  - Fully connected layer that acts similar to those in regular artificial neural networks.
- Dropout
  - Randomly sets some layer outputs to 0 to prevent over-reliance.
- Flatten
  - Turns multi-dimensional data into 1D. Prepares data for a decision.

III. METHODS AND MATERIALS

A. *Sensors Used*

In this project, smartphones were positioned along a 100-meter straight path at equal intervals to record the high-frequency tony outputted by a speaker held by the driver on a moving car. As the car travelled past each smartphone, the

Doppler effect caused a shift in the frequency of sound recorded. This shift was fused between all the sensors which were then analyzed to estimate the vehicle's velocity. Additionally, the time delay in the sound arrival time at the locations was captured, providing another way to validate the movement of the car.

The car's speedometer readings can be used as ground truth to evaluate the accuracy of the sensor fusion for velocity estimation. By comparing estimated velocities with the speedometer values, the performance of the project can be validated. Overall, the approach used readily available phones which is low-cost and scalable.

B. *Experimental Setup*

Please refer to Figure 2 for a visual understanding of the setup. The experimental setup was as follows:

- Five audio receivers (phones) are evenly positioned from Points1 through 5 along a straight 100m path.
- A loud noise in the form of a human shout is emitted to align the time series data readings for the sensors.
- The car begins accelerating from the 'Start' position.
- It aims to reach the constant speed from the 'Pickup Speed' marker before entering the measured section between points 1 to 5.
- For each trial, the car maintains a constant speed of 10, 20, 30, 40 and 50 kmph.
- A 2000 Hz continuous tone gets played from a speaker held outside the driver side window to ensure it gets picked up by the receivers.
- A high frequency tone is chosen to ensure that it can be differentiated from the low frequency engine noise during data pre-processing.

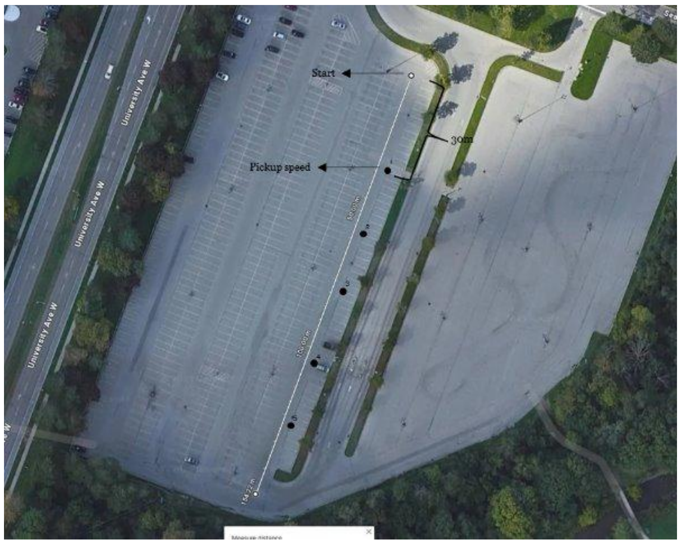

Figure 2: Experimental Setup

C. *Unscented Kalman Filter (UKF) Formulation*

1) *Initial Process Model and State Vector*

The initial state vector can be seen in equation (8). It considers the kinematic qualities for an object moving in two dimensions.

$$x_k = \begin{bmatrix} x_k \\ y_k \\ v_{x_k} \\ v_{y_k} \\ a_{x_k} \\ a_{y_k} \end{bmatrix} \tag{8}$$

The state vector used alongside the initial process model, which can be seen in equation (9). It is a 2D kinematics process prediction, which was the initial attempt, later reformulated for reasons seen in X-Y Attempt.

$$f(x) = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

2) *Initial Measurement Model and Measurement Vector*

There were five microphones collecting data, as seen in Experimental Setup. Frequency data, as seen in the Spectrogram Results spectrograms were the inputs. This was used in the sensor vector, seen in equation (10).

$$z_k = \begin{bmatrix} f_{o,1} \\ f_{o,2} \\ f_{o,3} \\ f_{o,4} \\ f_{o,5} \end{bmatrix} \tag{10}$$

From here, these frequency values are used with a Doppler model setup, as seen in Doppler Effect Doppler Effect Doppler Effect . For a given state, the following results are observed, seen in equation (11).

$$\begin{bmatrix} h(x_k)_1 \\ h(x_k)_2 \\ h(x_k)_3 \\ h(x_k)_4 \\ h(x_k)_5 \end{bmatrix} = \begin{bmatrix} f_s \cdot \dfrac{v}{v - \left([v_{x_1} v_{y_1}] \cdot \dfrac{\begin{bmatrix} x_1' \\ y_1' \end{bmatrix}}{\sqrt{x_1^{r2} + y_1^{r2}}} \right)} \\ f_s \cdot \dfrac{v}{v - \left([v_{x_2} v_{y_2}] \cdot \dfrac{\begin{bmatrix} x_2' \\ y_2' \end{bmatrix}}{\sqrt{x_2^{r2} + y_2^{r2}}} \right)} \\ f_s \cdot \dfrac{v}{v - \left([v_{x_3} v_{y_3}] \cdot \dfrac{\begin{bmatrix} x_3' \\ y_3' \end{bmatrix}}{\sqrt{x_3^{r2} + y_3^{r2}}} \right)} \\ f_s \cdot \dfrac{v}{v - \left([v_{x_4} v_{y_4}] \cdot \dfrac{\begin{bmatrix} x_4' \\ y_4' \end{bmatrix}}{\sqrt{x_4^{r2} + y_4^{r2}}} \right)} \\ f_s \cdot \dfrac{v}{v - \left([v_{x_5} v_{y_5}] \cdot \dfrac{\begin{bmatrix} x_5' \\ y_5' \end{bmatrix}}{\sqrt{x_5^{r2} + y_5^{r2}}} \right)} \end{bmatrix} \tag{11}$$

3) *Modified Process Model and Measurement Vector*

The process model and state vector were later modified to only use the y-dimension, for reasons discussed in X-Y Attempt. The resultant state vector is very similar to the initial attempt, with the x-dimension removed, as seen in equation (12).

$$x_k = \begin{bmatrix} y_k \\ v_{y_k} \\ a_{y_k} \end{bmatrix} \tag{12}$$

The following process model also uses basic kinematic equations, only using the y-axis this time, as seen in equation (13).

$$F = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

4) *Modified Measurement Model and Measurement Vector*

Given that the process model and state vector have changed, the measurement model must change slightly, by keeping the x-axis values constant at 0. The measurement vector remains unchanged, however, seen in equation (10), as there are still five sensors reading five frequencies. The modified measurement model can be seen in equation (14).

$$\begin{bmatrix} h(x_k)_1 \\ h(x_k)_2 \\ h(x_k)_3 \\ h(x_k)_4 \\ h(x_k)_5 \end{bmatrix} = \begin{bmatrix} f_s \cdot \dfrac{v}{v - \left([0\ v_{y_1}] \cdot \dfrac{\begin{bmatrix} 0 \\ y_1' \end{bmatrix}}{\sqrt{0 + y_1^{r2}}} \right)} \\ f_s \cdot \dfrac{v}{v - \left([0\ v_{y_2}] \cdot \dfrac{\begin{bmatrix} 0 \\ y_2' \end{bmatrix}}{\sqrt{0 + y_2^{r2}}} \right)} \\ f_s \cdot \dfrac{v}{v - \left([0\ v_{y_3}] \cdot \dfrac{\begin{bmatrix} 0 \\ y_3' \end{bmatrix}}{\sqrt{0 + y_3^{r2}}} \right)} \\ f_s \cdot \dfrac{v}{v - \left([0\ v_{y_4}] \cdot \dfrac{\begin{bmatrix} 0 \\ y_4' \end{bmatrix}}{\sqrt{0 + y_4^{r2}}} \right)} \\ f_s \cdot \dfrac{v}{v - \left([0\ v_{y_5}] \cdot \dfrac{\begin{bmatrix} 0 \\ y_5' \end{bmatrix}}{\sqrt{0 + y_5^{r2}}} \right)} \end{bmatrix} \tag{14}$$

D. UKF Tuning

The Unscented Kalman Filter (UKF) was used to fuse the Doppler-derived frequency shifts to estimate the vehicle's velocity.

Tuning the UKF parameters is essential for achieving accurate and stable performance, especially for nonlinear systems where model uncertainties and measurement noise can significantly impact filter convergence.

1) *Measurement Noise Covariance (R)*

This parameter represents the expected noise in the observed measurements. A high value of *R* makes the filter overly trust the model prediction, which leads to slow responsiveness. On the other hand, a low value of *R* makes the filter highly sensitive to noisy measurements. This leads to erratic and incorrect estimations. For this experiment, the value of *R* was tuned iteratively based on the variance observed in the spectrogram processed frequency data.

2) *Process Noise Covariance (Q)*

The matrix *Q* represents the uncertainties in the system dynamics, including acceleration variability and model imperfections. A large value for *Q* causes the UKF to react too aggressively to small changes. Whereas a small value of *Q* can cause the UKF to ignore true variations in the systems. The *Q* matrix tuning was performed through repeated runs using ground truth from the speedometer as a reference

3) Sigma Point Parameters (α, β, κ)

α controls the spread of the sigma points. Smaller values of α reduce nonlinearity but at the same time they risk underestimating state uncertainty, while larger values increase sensitivity but can destabilize the filter. A value of α near 0.001 was used as a starting point.

β incorporates prior knowledge about the distribution of the state. For Gaussian distribution, which are commonly assumed in Kalman Filtering frameworks, a value of β = 2 is used.

κ is used as the secondary scaling parameter and was tuned to balance the influence of higher order moments without excessively widening the sigma spread.

4) Number of Sigma Points

For a state vector of dimension 'n', the UKF requires 2n+1 sigma points. These points are deterministically calculated to capture the mean and covariance of the system more accurately than linear approximations.

Tuning Process

The tuning was done empirically by minimizing the error between the UKF predicted speeds and the actual speed of the vehicle obtained from the speedometer. Multiple combinations of Q and R were tested to balance noise responsiveness with model stability. Additionally, the UKF performance was evaluated on its ability to track the ground truth, which was a variety of constant speeds.

The sigma point parameters were set as follows: $\alpha = 0.5, \beta = 2, \kappa = 2$.

The final process noise covariance was set as

$$Q = \begin{bmatrix} 1 \times 10^{-5} & 0 & 0 \\ 0 & 1 \times 10^{-5} & 0 \\ 0 & 0 & 1 \times 10^{-5} \end{bmatrix}$$

The final measurement noise covariance was set as

$$R = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}$$

These choses were validated by comparing the UKF-predicted speeds against the ground truth.

E. CNN Training

A convolutional neural network was used on the 2-dimensional spectrogram data. The neural network was built in Python using the NumPy, Sklearn, and TensorFlow libraries. Numerous models were tested using different parameters, data pre-processing, and models. The general process for the neural network can be broken down into the following four steps.

1) Processing of Data and Conversion to Spectrogram

This step involves loading the data into the Python script and then performing various processing methods before being input into the convolutional neural network. Note that much of the data processing for the convolutional neural network was done entirely separately from that of the unscented Kalman Filter.

To be most useful and applicable, minimal processing is desired. This ensures that, if this method were to be used in real-world scenarios, the raw audio files could be used directly without requiring processing time. Hence, much of the testing was done using spectrograms computed directly from the raw data using the Python library Librosa. When computing the spectrogram there are two main parameters: the number of bins, and the frequency range. The bins represent the number of divisions of the frequency range. Thus, more bins will divide the scale into more ranges. However, the audio equipment that was used in the process limits the number of bins by how precise it can acquire frequency data. Throughout the process, bin sizes of 64, 128, 256, and 512 were all used. Various frequency ranges within 0 and 8000 Hz were also tried. Through visual inspection of the data, it was decided that a bin number of 128 and a frequency range of 1000 – 2400 Hz was best. These values were used for most of the testing.

After converting the data into spectrograms, the power values are converted to dB since this is common when working with audio data.

Next, various methods of filtering and cropping were tried. Some examples include a median filter with window size of 3x3, cropping to a smaller frequency range of 1800 – 2200 Hz, time-cropping to only slightly longer than the effect is present. The latter two resulted in significantly lower training times. However, even without filtering, all individual model training took less than 10 minutes, likely due to the small amount of data used.

Thresholding is used to cut out low-energy noise by replacing any values below the threshold with the threshold itself. Different thresholds from -40 dB to -80 dB were attempted and compared.

2) Preparing Data for Training

To prepare the data for the model, various methods were used to ensure functionality. These methods are outlined below.

Firstly, all the data needs to be the same size to work with the model. Thus, padding needed to be done. This involved augmenting the data with empty data at the end. Then, the data is cropped to a specified time width to minimize the size and training time. This size varied from 6 – 60 seconds depending on the extent of cropping in data pre-processing.

To help with training speed and ensure a consistent input scale, the data is then normalized. Two different types of normalization were tried: [0, 1] and [-1, 1]. The equations used for each are given below.

$$norm_{[0,1]} = \frac{data - minimum}{0 - minimum}$$

$$norm_{[-1,1]} = \left(2 * \frac{data - minimum}{0 - minimum}\right) - 1$$

Due to the application of thresholding, the minimum is equivalent to the chosen threshold.

In the chosen setup, the first sensor was closest to the start point. Thus, the vehicle was often not yet up to speed by the time it reached the first sensor. Therefore, the inclusion of this data was tested. The results of including and ignoring this data are compared.

Finally, the data is prepared to be used in the convolutional neural network. The Python library numpy is used to put the data into the proper sized arrays. The data is split into training and test data. Initially, an 80/20 training/test split was used. However, it was later decided to use 5-fold cross validation. This is because of the limited amount of data. By utilizing the cross validation, the model can be trained and tested on various sets of the same limited data. Additionally, this provides a more robust estimate of the performance since it averages the performance across different train/test sets.

3) *Building the CNN*

Upon pre-processing and preparing the data, the convolutional neural network architecture is defined, and the model is built. Numerous different combinations of layers, depths, training times, activation functions, and other variables were tried. However, all networks followed the same general structure which is defined below.

- Multiple layers of:
  - Convolution
  - Pooling
- Flatten
- Dense
- Dropout
- Dense

Throughout the process, the main components of this architecture that were altered were the number of convolution and pooling layers and the number of filters used in each layer. Deep and shallow networks were compared. One example of the depth and filters used is 3 layers deep with 8-16-32 filters. Another is 2 layers deep with 8-32 filters.

In addition to this, the number of epochs, optimizer, and activation function were all changed and tested. The epochs ranged from 30-300. The optimizer used most often was Adam. The activation functions tried were ReLU and leaky ReLU. The only loss that was used was mean squared error. A stop condition was often used and was based on the loss failing to improve for a set number of epochs. Upon this stop condition occurring, the weights are restored to the best seen in training. The number of epochs was altered from 10 to 150 for this stop condition. Throughout testing, the batch size, dropout rate, and learning rate were kept consistent at 4, 30%, and 0.001, respectively.

4) *Testing the CNN*

Finally, the network is tested. This step involved training and testing the network using numerous different variations that have been discussed previously. After each training and testing, the results were noted, changes were made, and a new model was trained. The models were evaluated on the mean average error. As discussed previously, this eventually became an average across the 5 folds of the mean average error. This process repeated until a reasonable effort had been made to achieve the best possible model given the quality and quantity of data and the time limitations.

IV. Results

A. *Spectrogram Results*

In this section, the spectrogram analysis of the recorded acoustic signals is presented and discussed. A spectrogram, which provides a time-frequency representation of the signal, was used to verify that the speaker's 2000 Hz tone was captured reliably. The raw spectrogram readings from four of the microphone sensors can be seen in Figure 3.

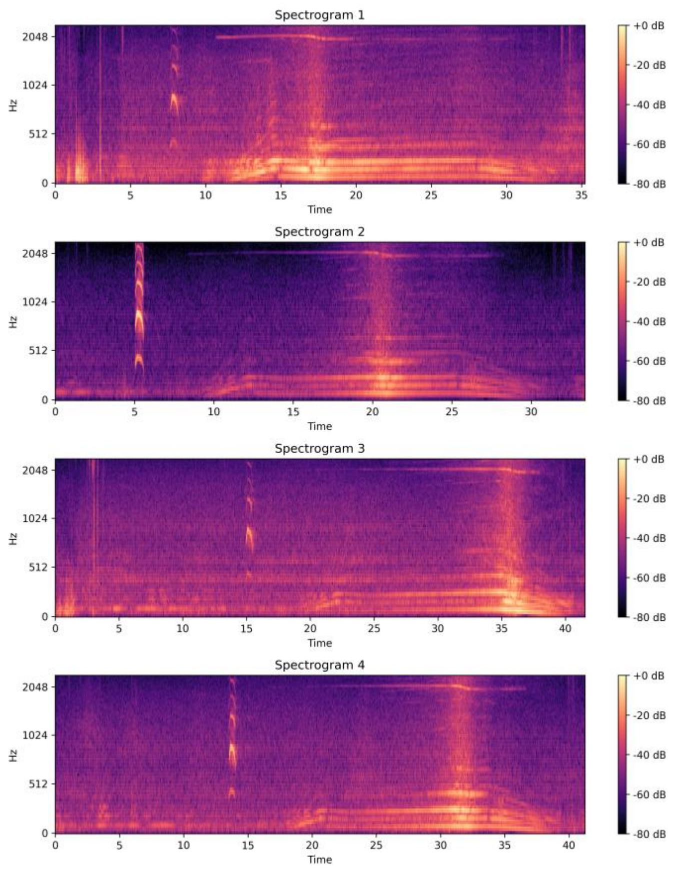

Figure 3: Raw spectrograms

Figure 4 shows how the different regions of one of the sensors spectrograms can be interpreted. The loud start reference appears as a multi-frequency sound at a specific timestamp. The Doppler effect curve can be seen soon after occurring at high frequencies of near 2000 Hz in the form of a sinusoidal curve. The low frequency engine noise from the moving vehicle is also visible.

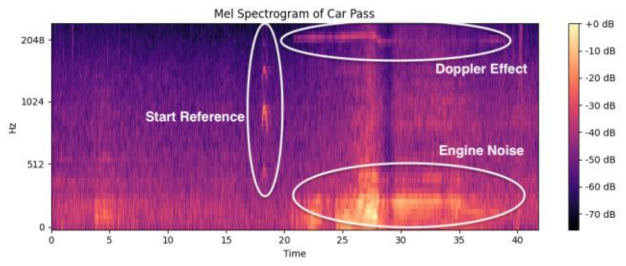

Figure 4: Interpreted spectrogram for car passing along sensor

A band-pass filter, set between 1900 and 2100 Hz, was then applied to the raw audio data to isolate the frequency band corresponding to the target 2000 Hz tone. Furthermore, only values within a specific decibel range were accepted to aid in highlighting the Doppler effect curve. As illustrated in Figure 5, the filtered spectrogram clearly reveals a persistent, dominant frequency band near 2000 Hz, along with a curve. The overlay green highlight in the figure emphasizes the high-intensity regions, confirming that the filtering process effectively suppresses unwanted frequencies and isolates the signal of interest.

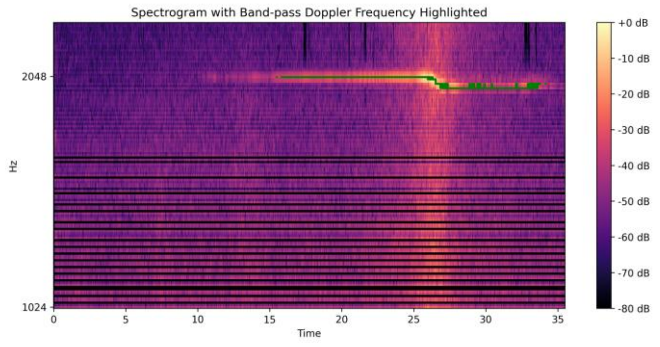

Figure 5: Sample spectrogram with band-pass Doppler highlighted

Further analysis involved comparing spectrograms from the multiple sensors placed along the test with a synced time series. As shown in Figure 6, the time-shifted peak intensities across different sensors provide evidence of the vehicle’s progression along the sensor array. Although the tone itself originates from a stationary speaker held outside the moving vehicle, the sequential detection of the signal due to the vehicle’s movement past the sensors, allows us to capture the time delays needed for estimating vehicle speed via Doppler shift.

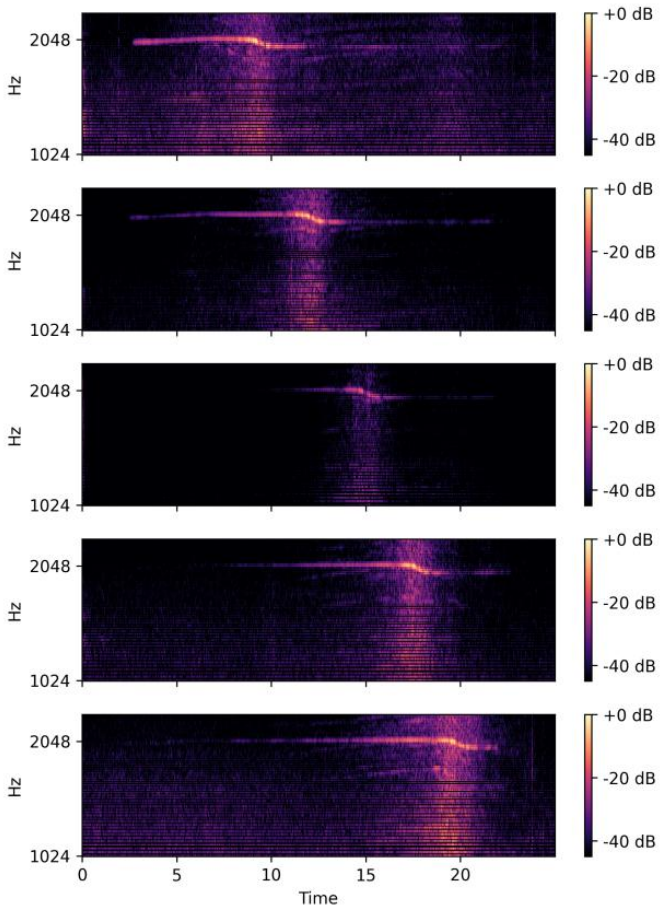

Figure 6: Time-synced spectrogram results

The frequency values and the corresponding timestamps extracted during the spectrogram processing as seen in Figure 5 are logged and formatted into a CSV file to serve as input for the subsequent UKF and CNN sensor fusion stages.

To further improve the quality of the frequency readings, a low-pass filter with outlier rejection is applied to the readings from Figure 6, as seen in **Error! Reference source not found.**.

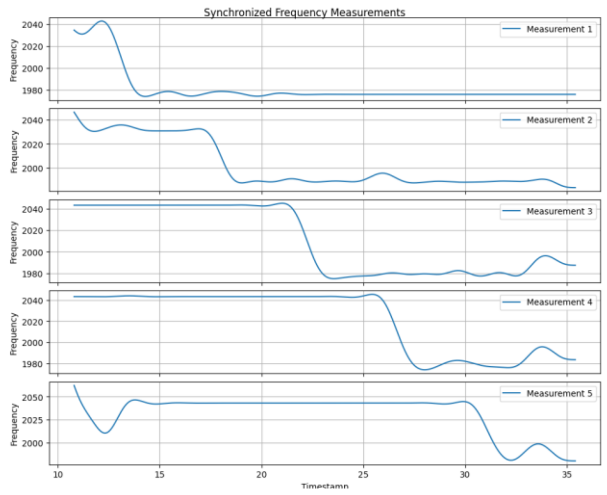

Figure 7: Low-pass-filter with outlier rejection for cleaner frequency readings

B. Sensor Fusion UKF Results

1) X-Y Attempt

Initial efforts, as seen in Initial Process Model and Initial Measurement Model, used a full 2D UKF to estimate the object’s position, velocity and acceleration in both the x and y directions. However, consistent instability and divergence were observed in the x-axis. Straight line motion, where x-axis velocity should be 0, had the UKF predicting non-zero velocities in the x-axis, as seen in Figure 8.

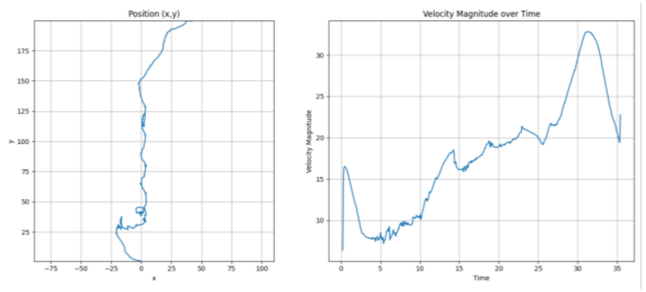

Figure 8: Ground truth is straight line 10 km/h motion, unlike predicted results

The issues seen in Figure 8 can be attributed to poor sensor placement. All five microphones were placed in a straight line along the y-axis only. Consequently, the systems had strong observability in the y-axis, but poor sensitivity in the x-axis. The line-of-sight vectors were mainly aligned in the y-direction. The Doppler shift model relies on the projection of the object's velocity vector onto the line-of-sight direction to extract radial velocity measurements. When an object moves along the x-axis, the component of its velocity projected onto these vertically oriented line-of-sight vectors is negligible, causing minimal Doppler shift sensitivity.

2) *Simplified Y Only*

By only considering the motion in the y-direction, the previous issues seen in X-Y Attempt with x-axis velocity prediction can removed.

By reducing the state vector to include only the y position, velocity and acceleration, the filter avoids unobservable states.

The results for straight line, constant velocity along the y-axis for the speeds of 10, 20 and 40 km/h can be seen in Figure 9, Figure 10, and Figure 11. The constant speed was determined using a modern car speedometer and was kept with 2 km/h of the target constant speed during operation.

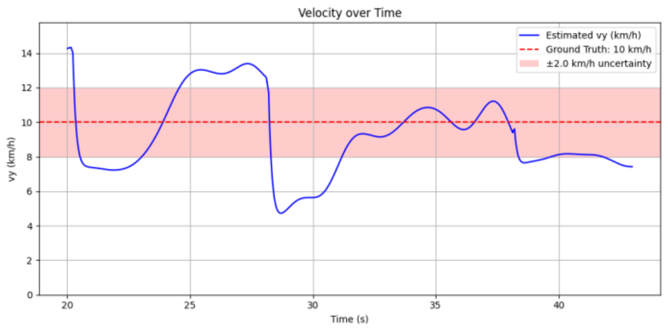

Figure 9: 10 kph constant straight-line motion

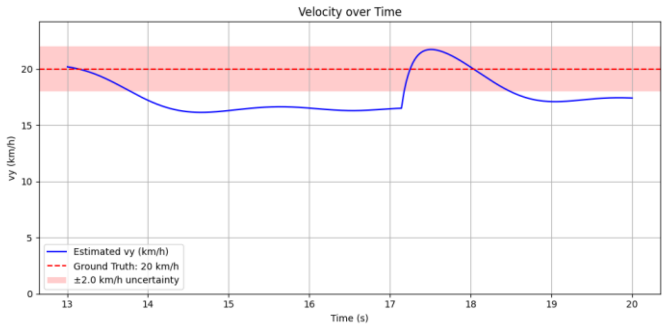

Figure 10: 20 kph constant straight-line motion

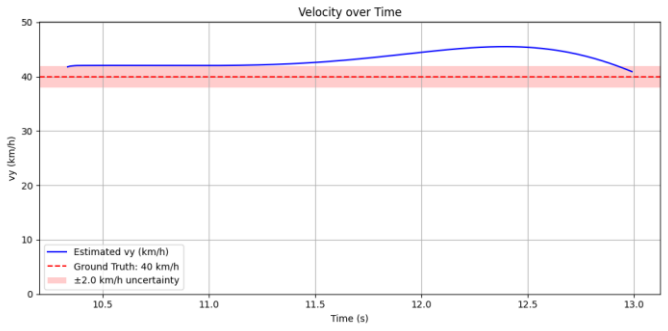

Figure 11: 40 kph constant straight-line motion

Some of the microphone data for 30 km/h and 50 km/h was poor, so it was excluded from the results.

Despite the improvements resulting from only considering the y-dimension results, there were still issues with accuracy. Some of these issues can be attributed to the fact that the frequency shift saturates at low relative velocities, making it difficult to distinguish between small changes in speed. Additionally, sensor and environmental noise add significant variations in the collected data.

C. *Convolution Neural Network Results*

The convolutional neural network was trained and tested on different variations of data pre-processing, preparation, neural network architecture, and other variations. Notable results that lead to the final, best model are discussed below.

Regarding the pre-processing, it was eventually determined that a bin size of 128 and a frequency range of 1024 – 2400 Hz resulted in the best model. This is likely because more bins exceed the accuracy limits of the microphones used to record the data. Additionally, somewhat surprisingly, filtering and cropping resulted in worse performance. Although the model trained significantly faster, there was clearly important data and relationships being lost when filtering and cropping. This could likely be fixed by changing the extent to which the cropping and filtering was done. However, given the desire to minimally process the data and the fact that the model works well without this, it was decided not to pursue such changes. With thresholding, the best value was found to be -70 dB. Testing with different thresholds did not significantly impact the results, however -70 dB consistently performed slightly better.

Next, in relation to data preparation, a width of 60 seconds outperformed any smaller widths. The increase in training time was significant, however the improvements were enough to justify such time. Using such a large window ensures that no data is missed in the doppler curve for any of the data. This is crucial to the performance of the model. The [0, 1] normalization performed much better than the [-1, 1] normalization even when the activation function was changed to account for such changes. This is likely related to the shape of the data and the architecture of the model, because it would be expected that [-1, 1] would perform better in many cases. The inclusion of the first sensor consistently decreased the performance of the model, as was expected. Therefore, much of the testing was done without it included. Lastly, using cross fold validation typically negatively impacted the model. This is expected and explainable by the increased robustness of the cross-fold validation. Thus, it was used for much of the testing.

Upon building the CNN, various conclusions about performance of different variations were made. A layer pattern of doubling the number of filters for each layer and using deeper networks with more layers tended to result in better performance. Further, longer training times with more epochs generally improved performance. However, there was risk of overtraining when using exceptionally high numbers of epochs. A sample of the loss function plot in an overfit model is given in Figure 12, below.

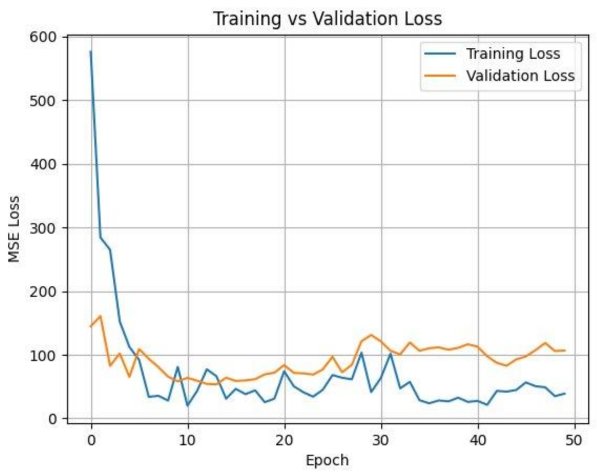

Figure 12: Training and Validation Loss in an Overfit Model

Notably, the validation loss is increasing while the training loss is flat or decreasing. This shows that the model is overfit. In this specific case, the model did not require many epochs to reach its optimal fit.

The activation function that performed the best was ReLU. Leaky ReLU was also tried when using the [-1, 1] normalization, however it performed much worse. The optimizer that was found to work best was the Adam optimizer. This combination of activation function and optimizer working best is sensible given that they are two of the most common and best performing. Lastly, a more restrictive stop condition, one that required more epochs without improvement, typically lead to lower error. This is sensible because it allows more training time while also not overfitting.

Some variations and their outcomes are presented in Table 1, below.

Table 1: Various Model Results

Epochs	Convolution/ Pooling Depth and Filters	Notes	Results (Mean Average Error)
50	Layer 1: 16 Layer 2: 32	- Overfit	**9.34 km/h** (Not Cross-Fold)
75	Layer 1: 8 Layer 2: 32	- No Sensor 1	**3.43 km/h** (Not Cross-Fold)
150	Layer 1: 8 Layer 2: 16 Layer 3: 32	- N/A	**4.7 km/h** (Not Cross-Fold)
300	Layer 1: 8 Layer 2: 16	- No Sensor 1	**2.87 km/h** (Not Cross-Fold)
75	Layer 1: 8 Layer 2: 16	- No Sensor 1 - 3x3 Median Filter	**5.15 km/h** (Cross-Fold)
150	Layer 1: 8 Layer 2: 16 Layer 3: 32	- No Sensor 1 - [-1, 1] Normalization - Leaky ReLU	**8.43 km/h** (Cross-Fold)
300	Layer 1: 8 Layer 2: 16 Layer 3: 32 Layer 4: 64	- No Sensor 1 - Stop Condition: 30	**4.11 km/h** (Cross-Fold)
300	Layer 1: 8 Layer 2: 16 Layer 3: 32 Layer 4: 64	- No Sensor 1 - Stop Condition: 150 - Threshold -70 dB	**2.62 km/h** (Cross-Fold)

Note that the table above includes only certain tests that best show the results of varying parameters and processing. The bottom row contains the best performing model. This model is fully outlined below.

The chosen model converts the data into spectrograms with 128 bins from 1024 to 2400 Hz. Thresholding is applied with a threshold of -70 dB. No other pre-processing is done. Next, the data is padded to 60 seconds and [0, 1] normalized. The data from the first sensor is not included. 5-fold cross validation is used with an 80/20 training/test split. The model uses 4 convolution and pooling layers with 8-16-32-64 filters. Then there is a flatten layer, a dense layer with 64 neurons, a dropout layer, and another dense layer with 1 neuron. The training time is set to 300 epochs. The chosen activation function is ReLU. The Adam optimizer with a learning rate of 0.001 is used. Mean squared error is the loss function and there is a stop condition at 150 epochs. The batch size is 4 and the dropout rate is 30%. This model results in a mean error of 2.62 km/h when using 5-fold cross validation.

Given the limited data and the experimental setup, this is a very well-performing model. Even with limited runs the model can generalize well. Additionally, the vehicle was not traveling at the exact speed for the entire time. There were small variations due to driver error. Further, the vehicle speedometer is likely not perfectly accurate. Taken together, this means that the data being used to train the model itself is not perfect. Therefore, an error of 2.62 km/h is more than reasonable.

D. Limitations

1) UKF Limitations

A UKF is a suboptimal method for this application. The main drawback is the amount of preprocessing and prior knowledge that would not necessarily be known in the situations described in this paper.

Extracting the curves as described in Spectrogram Results is a tedious process, and often requires manual oversight, a non-optimal solution.

Additionally, the relative positions of the sensor and the starting point must be known in order to determine a line-of-sight vector.

These issues make it such that a CNN approach is superior, as much less preprocessing is required to get velocity estimations.

2) *CNN Limitations*

The convolutional neural network performance is largely limited by the lack of data. Convolutional neural networks thrive when there is a large amount of data, and thus, the limited amount available for this project impacts the performance.

The accuracy of the data collected is also a limiting factor for the neural network. Specifically, the potential for inaccurate driving speeds means that the model may be trained with incorrect labels. This can lead to a model that does not predict as well as it could.

Lastly, the large number of variations and changes that can be made to the CNN make it nearly impossible to optimize.

V. Conclusion

A. *Overall Results*

The project successfully measured vehicle velocity along the y-axis, that is, the straight path along which the car drove. By using the doppler effect through smartphone audio recordings, it demonstrated a low-cost and easy solution requiring no additional hardware. Results were verified against the car’s speedometer to validate the approach.

The system was less accurate for estimating velocity in the x-axis due to absence of sensor placements in that direction in the setup, for reasons discussed in X-Y Attempt. Additionally, background noise sensitivity remains an issue which needs to be addressed with better filtering. Overall, future work should be on improving robustness against noise and incorporate lateral sensor coverage for full 2D motion tracking.

In regard to the convolutional neural network, a mean prediction error of 2.62 km/h over 5-fold validation was achieved. This was done with very minimal data processing required, which is an advantage of this system. The model can take raw data and very quickly compute an accurate prediction. Numerous variations were tested when finding the best model, however more work could be done to even further optimize. Additionally, more data would be highly beneficial in improving the neural network even further.

VI. References

[1] D. W. B. Setiyono, "Speed Estimation On Moving Vehicle Based On Digital Image Processing," [Online]. Available: https://www.researchgate.net/publication/317312246_Speed_Estimation_On_Moving_Vehicle_Based_On_Digital_Image_Processing. [Accessed 13 4 2025].

[2] A. Bonfitto and S. Feraco, "A Data-Driven Method for Vehicle Speed Estimation," [Online]. Available: https://www.researchgate.net/publication/350709410_A_Data-Driven_Method_for_Vehicle_Speed_Estimation. [Accessed 13 4 2025].

[3] The Physics Classroom, "The Doppler Effect," [Online]. Available: https://www.physicsclassroom.com/class/waves/Lesson-3/The-Doppler-Effect. [Accessed 12 4 2025].

[4] Pinterest, "Doppler Effect Physics," 11 4 2025. [Online]. Available: https://ru.pinterest.com/pin/1894-me-gusta-3-comentarios-physics-formula-physics_formula-en-instagram-doppler-effect-foll--598626975455680200/.

[5] E. A. W. a. R. v. d. Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," [Online]. Available: https://groups.seas.harvard.edu/courses/cs281/papers/unscented.pdf. [Accessed 12 4 2025].

[6] A. Arami, "Lecture 13-Unscented Kalman Filter_W25_compact," [Online]. Available: https://learn.uwaterloo.ca/d2l/le/content/1111339/viewContent/5914720/View. [Accessed 13 4 2025].

[7] A. Biswal, "CNN in Deep Learning: Algorithm and Machine Learning Uses," [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network. [Accessed 13 4 2025].

[8] MathWorks, "What is a Convolutional Neural Network?," [Online]. Available: https://www.mathworks.com/discovery/convolutional-neural-network.html. [Accessed 10 April 2025].