ML RE1 - Reliability Test and improvement of Horizontal-Perpendicular FIUS Sensor for Distance and Object type Detection

Masters of Engineering
Information Technology
A S M Saiem Solimullah
Matriculation Number: 1427938
a.solimullah@stud.fra-uas.de

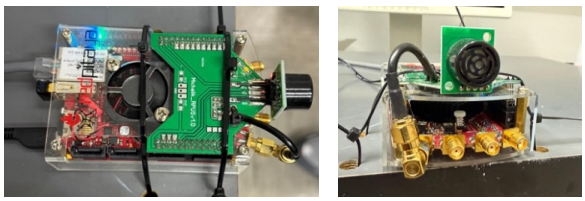
Masters of Engineering
Information Technology
Tanvir Ahmed
Matriculation Number: 1386435
tanvir.ahmed@stud.fra-uas.de

Abstract*- The ultrasonic sensors which use high frequencies sound wave can be used to identify people or objects along with measuring distance. This sensor uses sound waves that are emitted, bounce off surrounding objects, and return to the sensor, which uses the time it takes for the sound waves to return to determine the distance. Red Pitaya is an open-source, multifunctional tool designed for education, research, and prototyping. Using this hardware and software tool data has been collected and then preprocessed with several filtering method for train and test the data using machine learning model. The machine learning model designed to predict the object distance with calculated distance with better accuracy. Peak detection approach and filtering applied on the dataset for building the machine learning model. *Multilayer Perceptron, a machine learning model, applied on the dataset to detect and measure the distance of the object which gives a superb accuracy on the collected datasets.

Keywords—ADC, FFT, PSD, Savitzky-Golayn Filter and MLP

I. INTRODUCTION

Technology these days is doing some amazing things, especially when it comes to figuring out if someone's nearby or not, without requiring camera sensors or any kind of too expensive devices. This paper dives into the common feature-based work of ultrasound, which uses sound waves that are too high-pitched for humans to hear. Detection can be found by analyzing more smart systems when a person is nearby without needing to touch or watch them closely. Consider an IOT based, smart home that automatically detect when someone walks in, or a robot that moves around without bumping into anyone, or even a safety system that alerts others, like near heavy machinery. Ultrasound is perfect for these kinds of applications because of its discreet and unwanted video recordings. Also, it works well in all kinds of environments whether it's dark or messy. In medical research, using pulse ultrasound measurement different thing can be diagnosed like blood pressure [1]. This project is based on the FIUS Sonar Sensor device outputs. The distance (d_{FIUS}) to an object can be determined using the inbuilt feature or using machine learning techniques. For distance measuring feature, FIUS uses the time takes for the ultrasonic pulse to return. In this project the object type was person and static object(soft/hard).



(a) Top View (b) Front view

Figure 1: Top view and front view of FIUS sensor

The FIUS is mounted about 40-50 cm above the ground, keeping an eye out for movement directly across its path. It's already decent at measuring distances, but this analysis aims to make it even better particularly at distinguishing between a person and other static object, like a chair or a box. The main goal of this project is to measure the FIUS's ability to detect the object and measure the distance of that object accurately and reliably. The steps include collecting and labeling ultrasonic echo data for different distance of the object, developing algorithms for echo detection, after measuring the distance apply supervised machine learning algorithms for object type detection based on the collected data features. And finally, implementing real-time sensor output based on detection results.

II. LITARATURE REVIEW

In the scenario of insufficient or absent of visual information for human detection, machine learning and ultrasonic sensing have become very important instruments. This literature review is based on the study of current research on machine learning classifiers, ultrasonic based detection systems and signal processing methods which are relevant to the development of the perpendicular person detection system based on FIUS.

In [2], Sonia et al. proposed a person detection framework based on one-class classifiers and ultrasonic sensors. Fuzzy rule-based classification was used in their model. In their model features were extracted from time and frequency domain. This approach outperforms conventional SVMs in real time scenarios with very few false positives. This research supports the viability of ultrasonic-based sensing in our project.

For Ultrasonic signals to remain intact, noise reduction is very important, particularly in dynamic settings. Mohamed et al. [3] proposed a CNN based method with a focus on suppression of noise emphasizing low distortion and high signal fidelity. Their proposed approach allows for a more accurate extraction of the signal of interest (SOI) than traditional low-pass filters and wavelet-based methods. This emphasizes how crucial it is to incorporate clever filtering techniques into our setup prior to categorization.

Tripathi et al. [4] demonstrates that ultrasonic sensor-based detection systems can maintain categorizing people in different garment types and under variety of environmental setup. Also, they show efficient categorization using fuzzy rule-based systems, SVM, and FFT features, as well as machine learning classifiers. This ensures the feasibility of the preprocessing techniques (ACF, Hamming, and FFT) used in our research for categorization and conditioning of signals.

Zhang et al. [5] compared Support Vector Machines (SVM) and Multilayer Perceptron (MLP) neural networks for

sensor fusion applications. According to their findings, it indicated that MLPs performance are sometimes better than SVMs based on the initialization and architecture. Also, MLPs were identified to be more adaptable but also architecturally sensitive. This aligns with our projects method which uses MLPs as a Keu classifier for signal-based detection tasks.

Galvao et al. [7] used MLP, SVM and KNN classifiers for anomaly detection in smart homes, combining accelerometer data and image processing to identify falls in the elderly. Among the models, MLP showed that it's performance in terms of accuracy and low false negative rate was outperforming. This highlights it robustness in human-related event detection. This applicability of MLP in situations where sensor data needs to be mapped to human presence or activity is further reinforces this study.

III. Theoretical Background

In this project, we have used several components to assess the reliability and improve an object detecting sensor system. Here in this section, those element descriptions will be given over the necessary hardware, the programming language, and the machine learning technique that have been employed to create the model.

A. Red Pitaya Based FIUS sonar sensor:

Red pitaya is a stem lab board that can measure analog and digital signal. This has the ADC features which enables it to convert the analog signal into digital signal and vice versa. Red pitaya's open-source hardware and software can be coupled with an easy-to-use web interface, which enhance data visualization and remote control. The sensor used for this project to measure the horizontal and perpendicular data has 16 general input and output physical ports along with an RJ45 socket connector for the Ethernet and an USB port. The red pitaya sensor can measure high range frequency signal. For this project, only 30 KHz to 50 KHz signal taken into account.

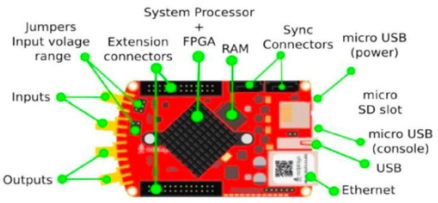


Figure 2: Red pitaya steam board configuration

The fully integrated ultrasonic sensor (FIUS) is a compact, high performance ultrasonic signal sensing device.

Ultrasonic sensor basically uses ultrasonic waves to calculate the distance to an object. This device is commonly used in robotics, automation and proximity detection applications. This device worked based on the echolocation of the signal. The sensor emits high frequency signals and then receive the same signal, which then calculate the distance of the sensor received signal using the time to flight formula.

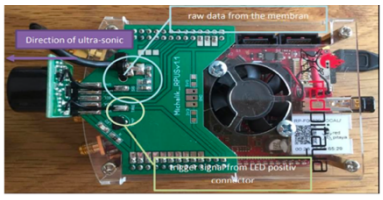


Figure 3: Red pitaya board's hardware and sensors

After calculating the signal data, first eco can be easily detected for the object and then the distance of that object can be measured.

B. Eco Detection

Eco detection is an important process to find the distance from the ultrasonic sonar sensor. It identifies the reflected sound waves that returns after hitting an object. From this reflected wave we can find the approximate distance of the object.

Using this first eco detection process we have identified the distance of the object in our project.

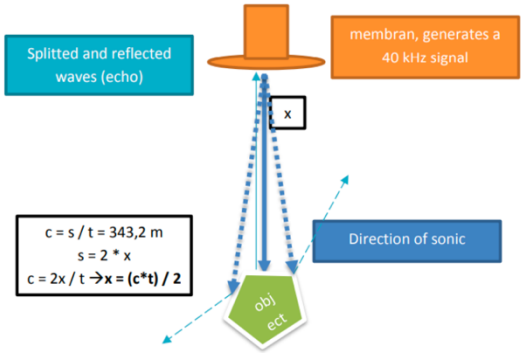


Figure 4: Ultrasonic pulse eco measuring process

C. Fast Fourier Transform:

FFT transform a signal from time domain to frequency domain which allows the analysis of the frequency component of the ultrasonic sound waves. We have applied autocorrelation function to the raw ADC signal which then transformed with FFT after applying a window to the data portion of the signal. This gives an output of power spectral density which helped to classify the signal later applied on the machine learning model.

D. ADC Data

The data we have found from the ultrasonic sensor is analog raw data. ADC samples this high frequency rate data and can convert it into digital form through FFT. These digital data frames then can be analyzed or classified using algorithm like MLP or SVM.

E. Savitzky-Golay filter

The Savitzky-Golayn (savgol) filter is a signal smoothing technique that reduces low signal noise while maintaining the key characteristics of the signal like peak and edges. It is useful for applications like ultrasonic sensing where clean signals are required without distorting unexpected shifts since it uses least squares to fit a polynomial to a changing window of data points [8]. It is useful for echo detection, feature extraction, and signal preprocessing in both scientific and technical applications. In respect to simple moving averages, it preserves the signal's default structure.

F. Multilayer Perceptron (MLP) Classifier

MLP is a type of feed forward neural network consisting of inputs, hidden and output layers. It is trained via backpropagation algorithm and useful for complex pattern learning, signal analysis and data classification. We have used MLP classifier for finding object type from different pattern of the data. We have used two different types of MLP

classifier for finding the object type and distance from the raw signal data.

G. Confusion Matrix

A confusion matrix is a table that shows us how well a classification model is performing is called a confusion matrix. It compares the predicted results of the model with the actual results. In our project we have analyzed the result of the performed model using this confusion matrix. This confusion matrix consists of four different elements as True Positive, True Negative, False Positive and False Negative.

- True Positive: When the expected and actual values are equal, we have a true positive case. This implies that even if the model predicted a positive outcome, the actual outcome is similarly positive.
- True Negative: When the expected value and the actual value are equal, the situation is said to be truly negative. However, in this case, the model's predicted and the actual outcome are both negative.
- False Positive: When the expected value matches an inaccurate value, this is known as a false positive. Type 1 prediction errors are the types that fall within this category.
- False Negative: When the expected value and the incorrect value match this is known as a false negative case. The actual value was positive, against the model's prediction that it would be negative. This is known as type 2 error.

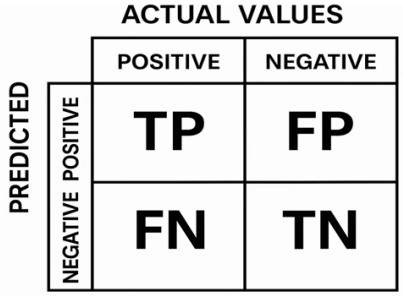


Figure 5: Confusion Matrix

Also using this confusion matrix, we have analyzed the result of different other measuring parameters like accuracy, precision, recall and F1 Score.

- Accuracy: How many predictions were correct in the dataset.
- Precision: How many predictions was accurate can be found through this matrix.
- Recall: How many true results we catch using the models.
- F1 Score: It defines a balance between precision and recall.

IV. Methodology

FIUS sensor which used in this project mounted with the red pitaya at a height of 40-50cm above from the ground for measuring the distance of the object along with the object type. It is based on a field-programmable gate array (FPGA) and a dual-core ARM Cortex-A9 processor, which allows real-time signal processing and control.

A. Data Acquisition

In order to detect the object type, two different types of object data have been taken from the sensor in different distance. In dataset 1, 2 and 3, Almost 9000 data have been collected every time for several distance with different objects. The object type was person and a static box. This whole experiment was monitored in the FIUS Ultrasonic Sensor UI interfaces. For different distance reading 1000 signal data has been observed. For individual time series data, 50000 data point have been found from the time series signal along with 17 header information. Also, before started to take the data measurement of the sensor for the object, different distance has been marked for manual measurement of the distance data.

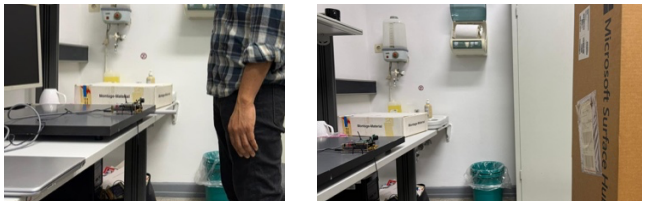


Figure 6: Collecting data from different types of object

These figures show the data collection setup for the person and static object. The signal found from the sensor has been collected through the red pitaya interface as shown in below figure.

Figure 7:UI interface for collecting raw signal

B. Darta Preprocessing

In order to find the information and features from the ADC signal found from the sensor, data need to be preprocessed. Later those preprocessed data were feed to the machine learning model for better accuracy.

a) Data Labelling: After collecting the data from the sensor, raw data has been added with two different level columns as “Distance” and “Object”. On the distance column, the measurement of manual distance has been added and on the object column, object types of the data have been defined. For visualsing the proper data in the dataframe we added header title in the dataset which has been used in the s

b) Removing Header: The data found from the ultrasonic sensor has 50017 columns, in which 17 column is header information and 50000 column is time series data

information. For finding the valid signal those headers were removed when preprocessed the signal data.

H2	H3	H4	H5	H6	H7	H8	H9	H10	...	D49993	D49994	D49995	D49996	D49997	D49998	D49999	D50000	Object	Distance
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-97.0	-97.0	-95.0	-96.0	-99.0	-98.0	-100.0	-104.0	person	30
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-100.0	-101.0	-103.0	-103.0	-105.0	-109.0	-109.0	-109.0	person	30
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-109.0	-111.0	-111.0	-102.0	-99.0	-99.0	-102.0	-107.0	person	30
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-95.0	-97.0	-98.0	-101.0	-103.0	-103.0	-96.0	-93.0	person	30
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-113.0	-112.0	-102.0	-97.0	-92.0	-97.0	-100.0	-105.0	person	30
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-86.0	-90.0	-104.0	-105.0	-94.0	-91.0	-94.0	-79.0	Box	100
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-94.0	-91.0	-89.0	-92.0	-105.0	-107.0	-98.0	-97.0	Box	100
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-114.0	-111.0	-105.0	-105.0	-96.0	-101.0	-104.0	-102.0	Box	100
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-105.0	-105.0	-93.0	-95.0	-95.0	-77.0	-91.0	-107.0	Box	100
000.0	0.0	1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-100.0	-98.0	-102.0	-104.0	-105.0	-107.0	-110.0	-111.0	Box	100

Figure 8: Labelled data with object type and distance

c) Remove Noisy Signal: For removing noisy signal we have applied an initial filter after collect the dataset for more clean signal using savgol filter.

```
def is_proper_signal(sensor_data, peak_height_factor=1.5, min_amplitude=50, std_threshold=5, straightness_threshold=50, max_variance=3000):
    # Apply Savitzky-Golay filter to smooth the data
    window_length = 101  # Must be an odd number
    poly_order = 3
    filtered_data = savgol_filter(sensor_data, window_length, poly_order)

    # Compute amplitude (max - min)
    amplitude = np.max(sensor_data) - np.min(sensor_data)

    # Compute standard deviation of the raw data and filtered data
    std_dev = np.std(sensor_data)
    filtered_std_dev = np.std(filtered_data)

    # Compute variance of the filtered data
    variance = np.var(filtered_data)

    # Compute dynamic peak threshold
    peak_threshold = np.mean(sensor_data) + peak_height_factor * np.std(sensor_data)

    # Detect peaks
    peaks, _ = find_peaks(sensor_data, height=peak_threshold, distance=50)

    # Check if the signal is "straight" based on filtered signal's standard deviation
    is_straight = filtered_std_dev < straightness_threshold

    # Check for large fluctuations in the filtered signal (variance)
    is_low_variance = variance < max_variance

    # Apply filtering conditions:
    return (amplitude > min_amplitude) and (std_dev > std_threshold) and (len(peaks) > 0) and is_straight and is_low_variance
```

Figure 9: Raw Data filtering process for removing noise

This makes a little easier for the model to detect the signal after reducing the noise. The output of the filtered signal after applied 'savgol' filter shows below.

```
Original number of rows in df_new: 12999
Number of rows with proper signals in df_filtered_new: 10905
Number of noisy rows removed: 2094
```

Figure 10: Output of filtered data from main data frame

d) First Eco Detection: For finding the distance of the object from the sensor data, first we need to find the first echo of the signal. After finding the first echo of the signal distance of the data can be calculated and these calculated data has been later stored in the data frame as a separate column leveled as "First Echo Index".

```
def detect_first_echo(sensor_data, threshold_factor=2.0, min_distance=50):
    """
    Detects the first significant peak (first echo) in the sensor data.

    Parameters:
    - sensor_data: 1D numpy array of the signal.
    - threshold_factor: Peak detection threshold (mean + factor * std).
    - min_distance: Minimum distance between peaks.

    Returns:
    - First peak index (or None if no peak found).
    """
    threshold = np.mean(sensor_data) + threshold_factor * np.std(sensor_data)
    peaks, _ = find_peaks(sensor_data, height=threshold, distance=min_distance)
    return peaks[0] if len(peaks) > 0 else None  # Return first peak index

# Add 'First_Echo_Index' column if missing
if 'First_Echo_Index' not in df_data.columns:
    df_data['First_Echo_Index'] = df_data.iloc[:, 17:50017].apply(
        lambda row: detect_first_echo(row.values), axis=1
    )
```

Figure 11: First eco detection procedure

From this "First Echo Index" column we have analyzed the distance of the first echo of the object and then measured the distance of the object.

H4	H5	H6	H7	H8	H9	H10	...	D49994	D49995	D49996	D49997	D49998	D49999	D50000	Object	Distance	First_Echo_Index
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-101.0	-100.0	-100.0	-101.0	-98.0	-101.0	-99.0	Person	60	2007
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-106.0	-105.0	-99.0	-96.0	-99.0	-98.0	-97.0	Person	60	2063
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-100.0	-102.0	-99.0	-102.0	-101.0	-103.0	-99.0	Person	60	2054
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-101.0	-98.0	-102.0	-101.0	-101.0	-100.0	-101.0	Person	60	2035
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-103.0	-105.0	-107.0	-106.0	-104.0	-105.0	-105.0	Person	60	2099
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-100.0	-102.0	-105.0	-105.0	-107.0	-108.0	-110.0	Box	140	11171
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-97.0	-104.0	-103.0	-100.0	-98.0	-97.0	-98.0	Box	140	11156
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-107.0	-108.0	-110.0	-113.0	-114.0	-113.0	-112.0	Box	140	11160
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-84.0	-85.0	-91.0	-91.0	-90.0	-88.0	-95.0	Box	140	11171
1.0	512.0	1.0	1953125.0	14.0	15.0	2622.0	...	-104.0	-105.0	-103.0	-103.0	-100.0	-103.0	-104.0	Box	140	11076

Figure 12: First eco index found using the applied process

e) One hat encoding of object type: The label "Object" which we added to the dataframe was in string values. To convert this data for machine learning use we have applied one hat encoding, which have then regenarate this object column in two different column value as '0' and '1'. This helps the model to classify the signal from the raw data.

f) ACF and Windowed Signal: Autocorrelation function and window has been applied to the data to find the power spectral density of the signal. For finding this power spectral density, sampling frequency assumed to be 195317 Hz and the applied window size was 300 on the signal data. This PSD calculation has been done using welch method which is an inbuilt method for finding the PSD in the python. These PSD features then feed to the model for train and testing the data to find the object type and distance from the signal.

C. MLP Classifier & SVM model:

As defined in the instructed document, we have used two different types of classifiers for building the model. One is multi-level-perceptron classifier and another one is SVM classifier. For finding the object type MLP classifier provide more accuracy than SVM model for this reason the classification process only applied with MLP.

```
# 2. Extract features (sensor data and First_Echo_Index)
sensor_data = df.iloc[:, 17:50017].values  # Sensor data columns (17 to 50017)
first_echo_index = df['First_Echo_Index'].values.reshape(-1, 1)  # First Echo Index feature

# 3. Compute the PSD for each row of sensor data
psd_features = []
for idx in range(len(df)):
    psd = compute_psd(sensor_data[idx])  # Compute the PSD for this row
    psd_features.append(psd)  # Collect the PSD features

# Convert PSD features to a numpy array and ensure the shape is consistent
psd_features = np.array(psd_features)

# 4. Combine the sensor data, First_Echo_Index, and PSD features
# If PSD has the same length as sensor data, you can concatenate the features like this:
X = np.concatenate((sensor_data, first_echo_index, psd_features), axis=1)  # Adding PSD features

# Target variables:
y_object = df['Object_Box'].values  # Object type (Box = 1, Person = 0)
y_distance = df['Distance'].values  # Distance to the sensor (continuous)

# 5. Train-test split
X_train, X_test, y_train_object, y_test_object, y_train_distance, y_test_distance = train_test_split(
    X, y_object, y_distance, test_size=0.2, random_state=42
)

# 6. Feature scaling: Standardizing the sensor data, PSD, and First_Echo_Index
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 7. Model: MLPClassifier for object classification and MLPRegressor for distance prediction
mlp_object = MLPClassifier(hidden_layer_sizes=(128, 64), activation='relu', solver='adam', max_iter=1000, random_state=42)
mlp_distance = MLPRegressor(hidden_layer_sizes=(128, 64), activation='relu', solver='adam', max_iter=1000, random_state=42)
```

Figure 13: MLP Classifier for object type and distance

Before classifying the data, several steps have been followed with the data as described below.

1. *Feature Extraction:* For model data input from a raw data the input data must be processed before sending it to the model. The feature we have extracted from the raw signal is PSD. This PSD calculation has been done for every single row and then used as an input of the classifier.
2. *Combine eco index and PSD:* After this PSD calculation we have taken this analyzed data's and first eco index as input variables for the classifier.

3. *Target Variable:* We have defined the target variable for the raw data from object type and distance for the machine learning model.
4. *Train/Test Data:* For the machine learning model we have to split the dataset for training and testing. For training we took 80% and for test we keep 20% of the data from the raw filtered data frame.

V. Result Analysis

This project consisting of three phases after collecting the data: Data Labelling, Eco detection for distance measuring and then detecting the object through machine learning models. It aims to enhance the accuracy of object detection and distance estimation. Throughout the dataset more than 11000 labelled ultrasonic signal sampled along with various distance and object type.

1. *Data Pre-processing Output Analysis:*

For measuring the distance of the object, first we have optimized and analyzed the sensor signal which has been found from the sensor. As previously mentioned about the 'Savgol' filter, the output of the sampled signal found as below with detecting the first eco position of the signal.

This below figure shows an eco-position detection from the signal with distance 40 cm (marked as 'X'), measured using analog tools (folding stick). The blue signal is the output of the raw signal which found after filtering as shown, in the background the original signal can be viewed.

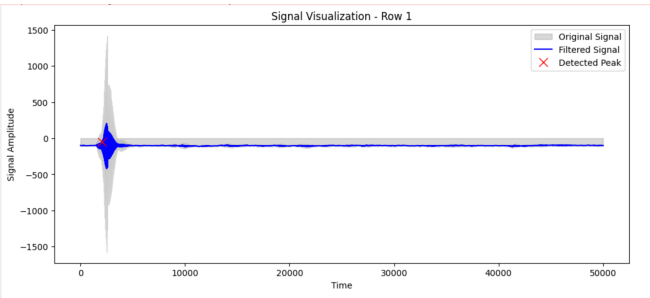


Figure 14: Peak detected signal after filtering

For representing the distance analysis, we have also visualized the whole data frame based on the distance found from first eco position.

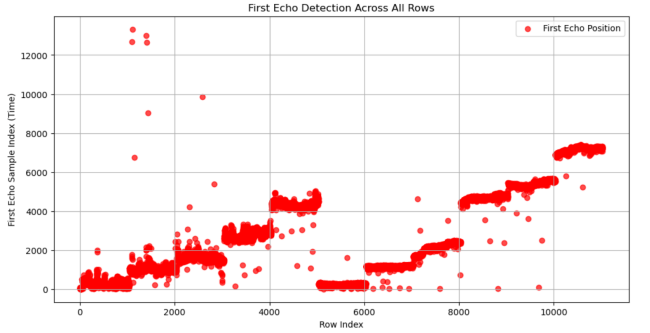


Figure 15: Data distance for the whole dataset

This figure shows the distance for the data both for person and static object. This data contains some error inputs which have been tried to reduce through the filtering.

2. Distance and Person detection Model Analysis:

For distance measuring we have calculated the first eco position of the signal and then from the first eco position index we have applied the regression algorithm from the raw data to train and test the distance using MLP model. The crated model has been applied in the sensor and then found the result for the detection of person from the data as shown in below figure.

Figure 16: Signal prediction for object type

In this figure, it is shown that even after noise the data peak has been detected by applied model and the person is detected also when the person is detected the sensor light got turned on.

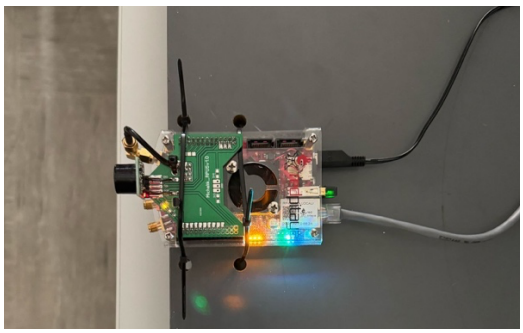


Figure 17: Turn on LED when person is detected

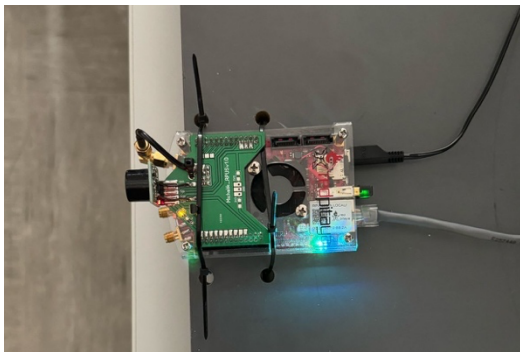


Figure 18: Turn off the LED when object is found

The MLP model demonstrated enhanced generalization and improved flexibility to intricate echo patterns owing to its multilayer construction.

3. Object Classification report and Confusion Matrix analysis:

After applying machine learning model with a labeled dataset, we also checked the classification report and the confusion matrix stats of that trained model. As per the classification report the accuracy found as below.

```
Object Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      1400
           1       0.97      1.00      0.98       781

    accuracy                           0.99      2181
   macro avg       0.98      0.99      0.99      2181
weighted avg       0.99      0.99      0.99      2181

Distance Mean Squared Error: 184.8787567458485
```

Figure 19: Classification report of the model

Also, the confusion matrix shows the status of the dataset after trained with machine learning model with all confusion matrix element where there was some type 1 and type 2 error present in the matrix as shown in below figure.

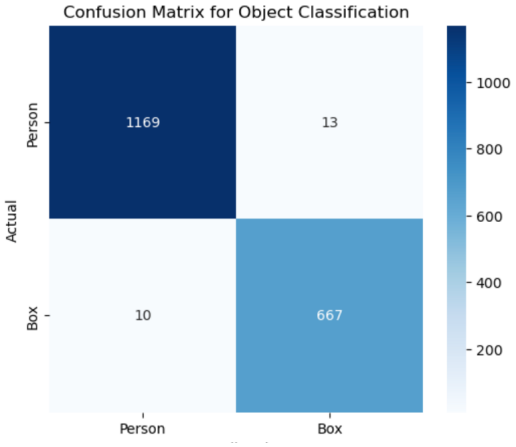


Figure 20: Confusion matrix visualization for the dataset

After analyzing the confusion matrix for the machine learning model, the model has been tasted with the input data and the accuracy is showing below for a single data signal.

```
Predicted Object: Person
Predicted Distance: 59.38 cm
True Object: Person
True Distance: 60.0 cm
```

Figure 19: Actual and predicted result analysis

This shows that for a signal data with 60cm distance (found using measuring tape) the model can predict the signal distance and the object type accurately.

VI. Future Scope

This current project is based on the person detection and distance measuring using ultrasonic sensor data, however, there has a significant potential for future development. One major scope is applying the convolutional neural network (CNN) model or Recurring Neural Network (RNN) model. Using CNN through PSD features can give a better output in this project as it can capture the whole spectrogram data from the signal. The mean square error we found for the distance model can also be improved by different noisy data filtering and apply feature engineering on the raw signal. This method with high performance runnable device can give the accurate outcome for this type of ultrasonic sensor signal. Also, the setup of the sensor can be applied differently for this model which can give more accurate precision for the model as in the current setup the sensor captures noisy data from its surroundings.

VII. Conclusion

This machine learning project based on horizontal perpendicular FIUS sensor detect the object type and the distance of that object by analyzing the signal data found from the ultrasonic sensor. Several preprocessing techniques have been applied on the data like filtering and PSD for feeding this signal into the machine learning model. For detecting real world object eco positions and type of objects, this project gives a proper accuracy on the ultrasonic sensor signal data.

References

[1] Hoctor, R. T., Dentinger, A. M., & Thomenius, K. E. (2004). Signal processing for ultrasound-based arterial pulse wave velocity estimation. In Proceedings of the IEEE Ultrasonics Symposium (Vol. 2, pp. 1492–1497).

[2] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[3] S. Sonia et al., “Ultrasonic Sensor-based Human Detector using One-class Classifiers,” IEEE EAIS, 2015.

[4] M. E. Mohamed et al., “A Convolution Neural Netowrk Based Machine Learning Approach for Ultrasonic Noise Suppression,” IEEE IUS, 2019.

[5] A.M. Tripathi et al., “Ultrasonic Sensor-Based Human Detection Using One-Class Classifiers,” IEEE EAIS, 2015.

[6] J. Zhang, L. Sun, and J. Cao., “SVM for Sensor Fusion – A Comparison with Multilayer perception Networks,” IEEE International Conference on Machine Learning and Cybernetics, 2006.

[7] Y. M. Galvao et al., “Anomaly Detection in Smart Houses: Monitoring Elderly Daily Behavior For Fall Detecting,” IEEE Conference on Smart Homes and IoT, 2017.

[8] Savitzky, A., & Golay, M. J. E. (1964). Smoothing and differentiation of data by simplified least squares procedures. Analytical Chemistry, 36(8), 1627–1639.