Ultrasound Imaging and Delay-and-Sum Reconstruction Algorithm

SAIGANESH S - [EE23B183]

June 8, 2025

Abstract

This report discusses the delay-and-sum reconstruction algorithm used in ultrasound imaging. We explore the impact of various parameters like speed of sound (C), number of microphones (Nmics), and number of samples (Nsamp) on the reconstructed image. The code simulates the behavior of ultrasound signals in the presence of obstacles and reconstructs the image based on the signals received by the microphones.

1 Introduction

The code presented in this report simulates the behavior of ultrasound waves as they interact with obstacles in a medium. The goal of the algorithm is to reconstruct an image of the environment using a delay-and-sum approach. This method relies on signals recorded by an array of microphones after reflecting off obstacles. By applying time delay calculations, the algorithm can generate a spatial map that represents the location of obstacles.

2 Motto of the Code

The motto of the code is to simulate and reconstruct the location of obstacles by analyzing the signals received by microphones placed in a specific pattern. The reconstruction is based on the delay and summation of signals, which are modeled using simple geometric distance calculations between the microphones, the source, and obstacles.

3 Code Structure and Parameters

The main function of the code is defined as:

```
def f (path=None,x_min=0,x_max=8,x_margin=1,x_resolution=200,y_min=-8,y_max=8,
y_margin=1,y_resolution=200,path_mode=False,
obstacles=[(3, -1), (2, 2), (1, -3)]):
```

The function accepts various parameters that define the simulation, including:

- `path` – The path to the data file, used when running the code in `path_mode`.
- `x_min, x_max` – The minimum and maximum x-coordinates for the grid used in the image reconstruction.
- `y_min, y_max` – The minimum and maximum y-coordinates for the grid.
- `x_resolution, y_resolution` – The resolution of the grid in the x and y directions.
- `obstacles` – A list of obstacles with coordinates, used to simulate the interaction with the ultrasound waves.
- `Nmics` – The number of microphones used to record the signals.
- `Nsamp` – The number of samples (time steps) used in the simulation.
- `C` – The speed of sound in the medium.

4 Procedure for Running the Code

To run the code, follow these steps:

1. The code can be run in two modes:
   - `path_mode == True`: In this mode, the obstacle list does not matter, as the data is loaded from a text file. Provide a valid path to the text file that contains the data.
   - `path_mode == False`: In this mode, you must provide a list of obstacles. Set the `path` parameter to `None`.
2. To execute the code, use the function `f()` with appropriate parameters. Example:

   ```
   f(path="path_to_data", x_min=0, x_max=8, x_resolution=200, y_min=-8, y_m
   ```

3. The code will generate a reconstructed image based on the signals recorded by the microphones and their time delays.

5 Answers to Key Questions

5.1 1. Maximum Amplitude and Obstacle Position

The maximum amplitude (yellow color) in the reconstructed image corresponds to the location where the sum of the delayed signals from the microphones aligns with the position of the obstacle. In this case, the approximate coordinates

of the maximum amplitude are $(30, 22)$. This is where the algorithm detects the presence of an obstacle based on the delays from the microphones. The signals are aligned constructively at this point, creating the maximum observed amplitude.

5.2 2. Maximum Obstacle Coordinates for Image Reconstruction

The obstacle's coordinates should lie within the grid defined by the parameters `x_min`, `x_max`, `y_min`, and `y_max`. In this code, the maximum values for the obstacle coordinates are:

$$x_{\max} = 8, \quad y_{\max} = 8$$

If the obstacle's coordinates exceed these values, the reconstruction may become inaccurate or fail to produce an image.

5.3 3. Effect of Speed of Sound (C) on Image Sharpness

The speed of sound C plays a crucial role in the time delay calculation for each microphone. If C is decreased, the time delays between the microphones become more pronounced. This results in sharper distinctions between obstacle positions, leading to a clearer image. Intuitively, a slower speed of sound allows for finer time-based resolution, helping to localize obstacles more precisely.

5.3.1 Effect of Different Speed of Sound Values (C)

Below are the reconstructions of the image with varying values for the speed of sound C. These changes affect the clarity and sharpness of the image.

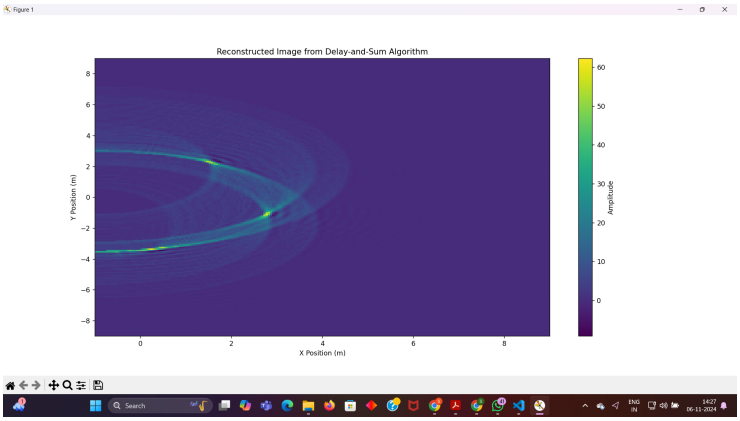

Figure 1: When speed is 1

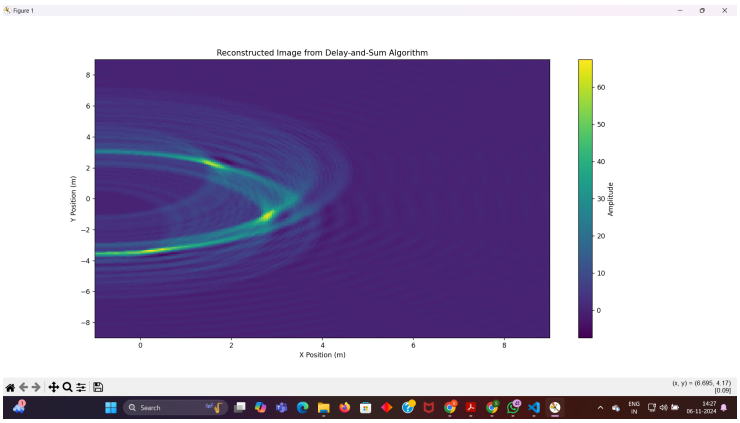

Figure 2: When speed is 2

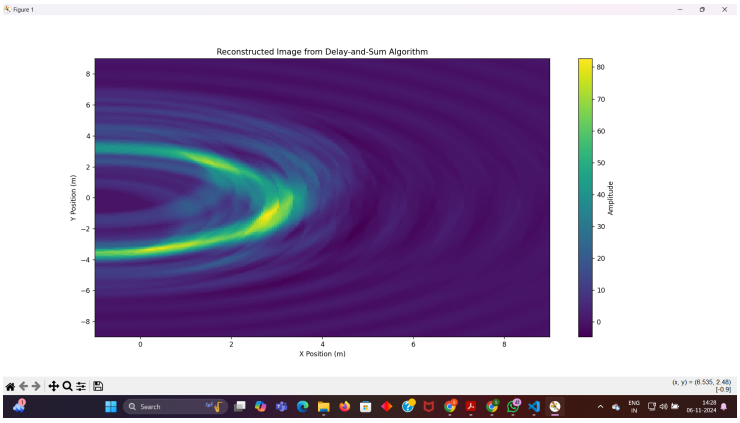

Figure 3: When speed is 5

5.4 4. Effect of Changing Number of Microphones (Nmics) and Samples (Nsamp)

The number of microphones (`Nmics`) and samples (`Nsamp`) significantly affect the resolution and accuracy of the reconstructed image:

- **Increasing `Nmics`** improves the resolution of the image by providing more data points and better localization of obstacles.
- **Decreasing `Nmics`** reduces the image resolution, potentially leading to less precise obstacle localization.
- **Increasing `Nsamp`** provides finer time resolution, allowing for more detailed calculations and sharper images.
- **Decreasing `Nsamp`** results in a coarser time resolution, reducing the sharpness and accuracy of the reconstructed image.

5.4.1 Effect of Different Number of Microphones (Nmics) and Samples (Nsamp)

Below are the images generated for different values of `Nmics` and `Nsamp`.

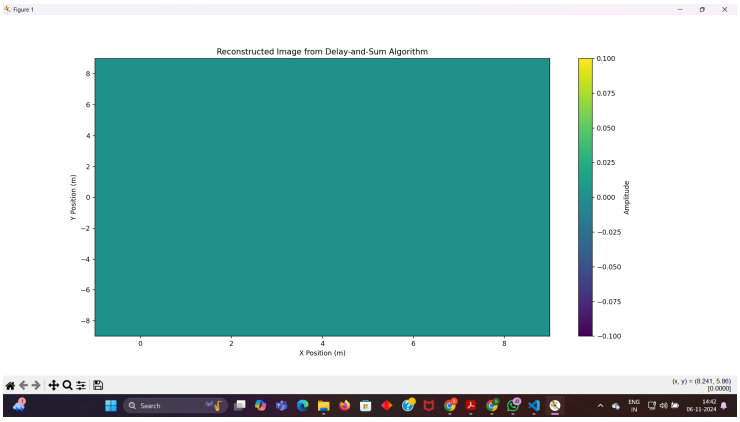

Figure 4: Nmics = 8 and Nsamp = 50

5.5 5. Generating Sinc Pulses and Their Effect on the Image

The sinc pulse is generated using the following function:

```
def wsrc(t, SincP=5):
    return np.sinc(SincP * t)
```

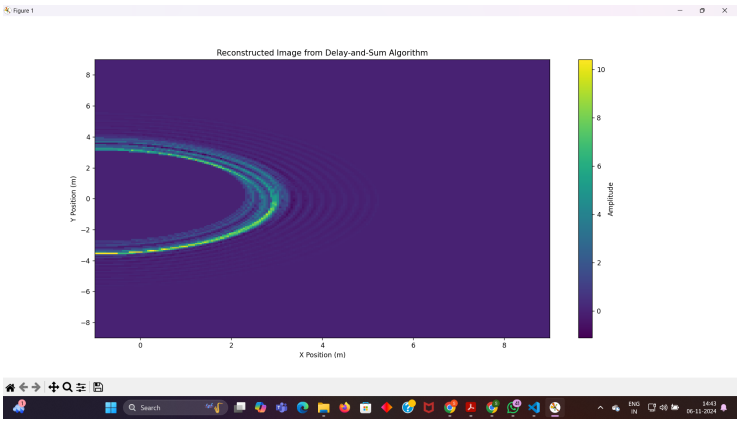

Figure 5: Nmics = 8 and Nsamp = 100

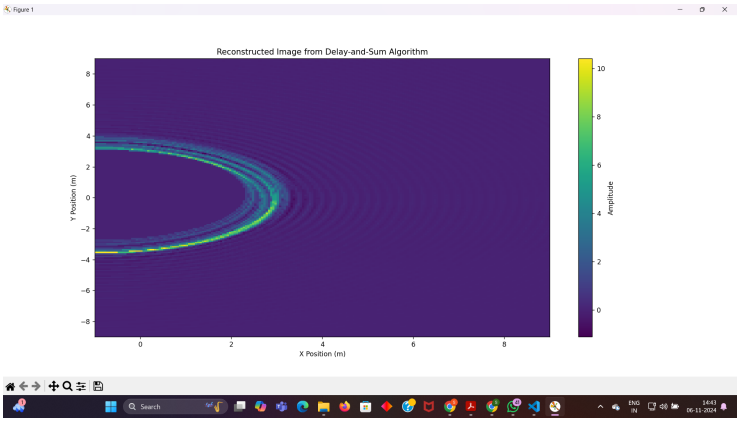

Figure 6: Nmics = 8 and Nsamp = 200

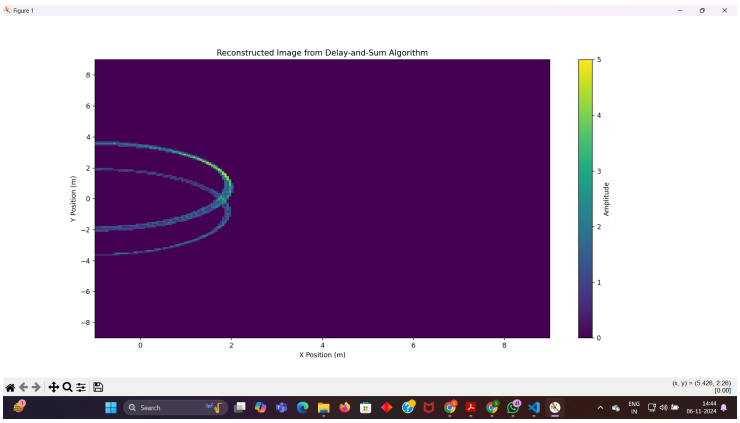

Figure 7: Nmics = 32 and Nsamp = 50

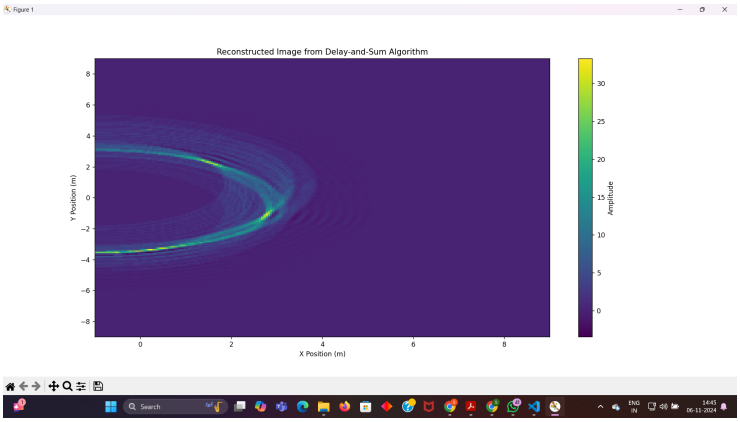

Figure 8: Nmics = 32 and Nsamp = 100

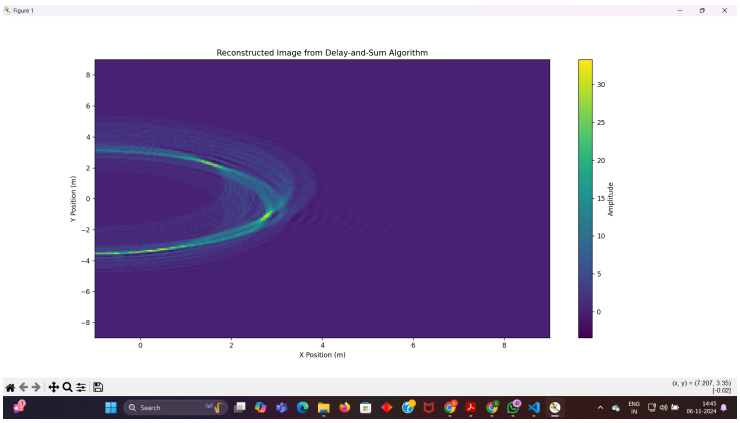

Figure 9: Nmics = 32 and Nsamp = 200

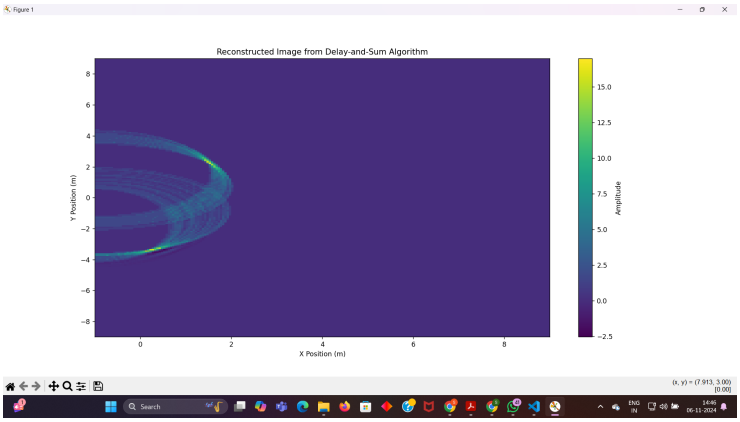

Figure 10: Nmics = 64 and Nsamp = 50

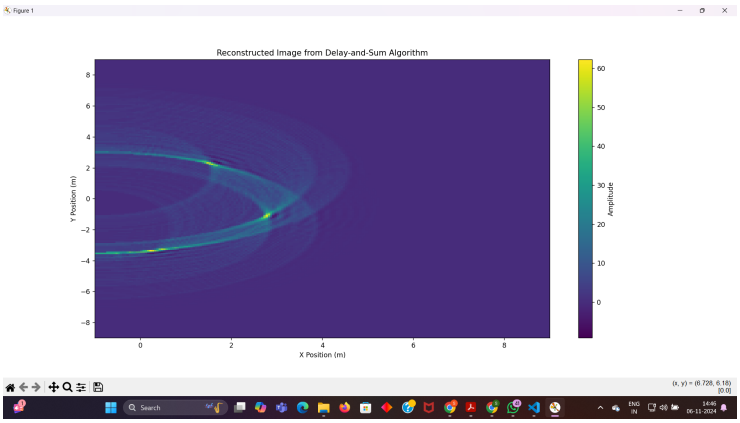

Figure 11: Nmics = 64 and Nsamp = 100

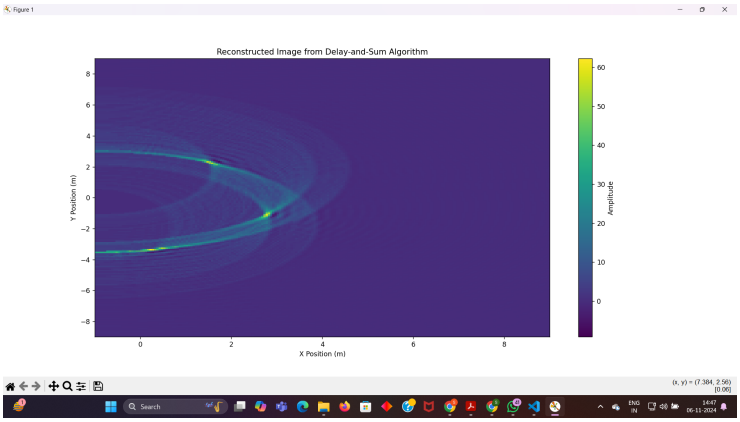

Figure 12: Nmics = 64 and Nsamp = 200

The parameter `SincP` controls the width of the sinc pulse. By changing the value of `SincP`, we can control how narrow or wide the pulse is:

- **Increasing `SincP`** narrows the pulse, making it more focused and sharper.
- **Decreasing `SincP`** broadens the pulse, making it more diffuse and resulting in a less sharp image.

The effect of this on the final image is that a narrower pulse (larger `SincP`) will result in a sharper image, allowing for better localization of obstacles, while a wider pulse (smaller `SincP`) may blur the image and reduce resolution.

6 my outputs when C=1(seems like It works better)

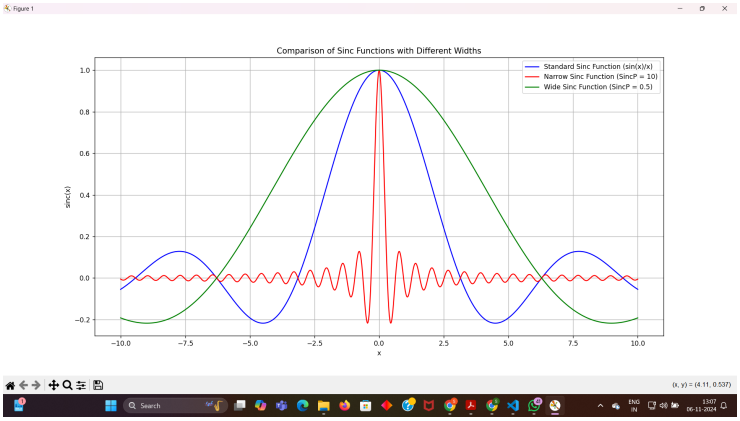

Figure 13: generating different sinc signals by varying the sinc-P parameter

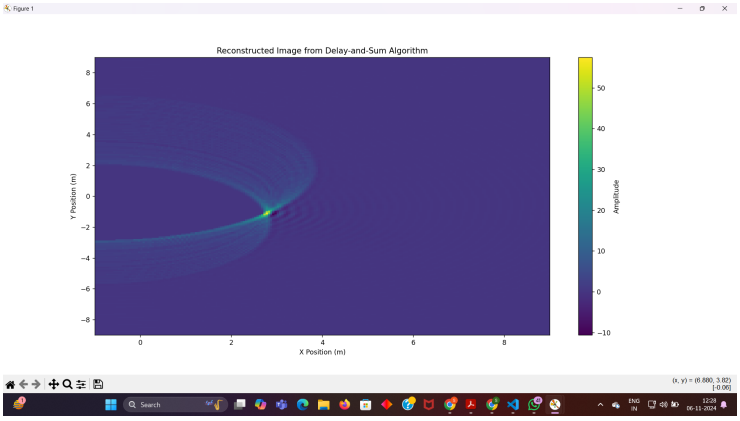

Figure 14: when a object is placed at a arbitrary position lets say (3,-1)

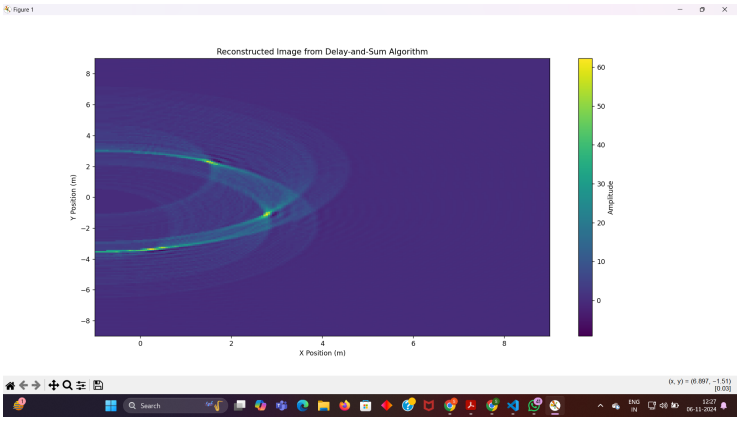

Figure 15: when a object is placed at a arbitrary positions lets say obstacles=[(3, -1), (2, 2), (1, -3)]

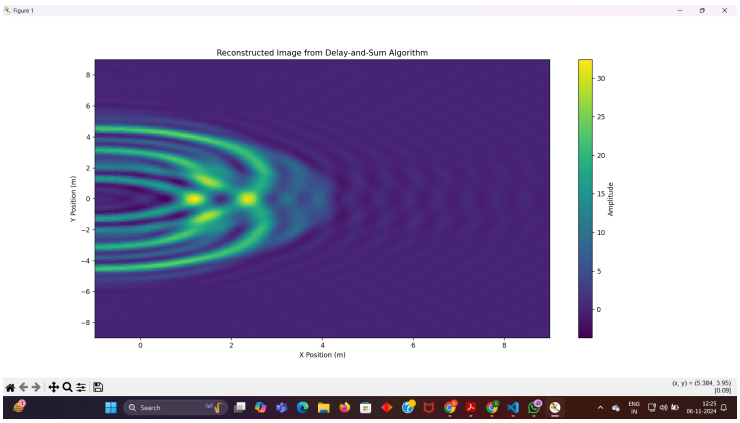

Figure 16: for mic outputs given by dataset RX2.txt

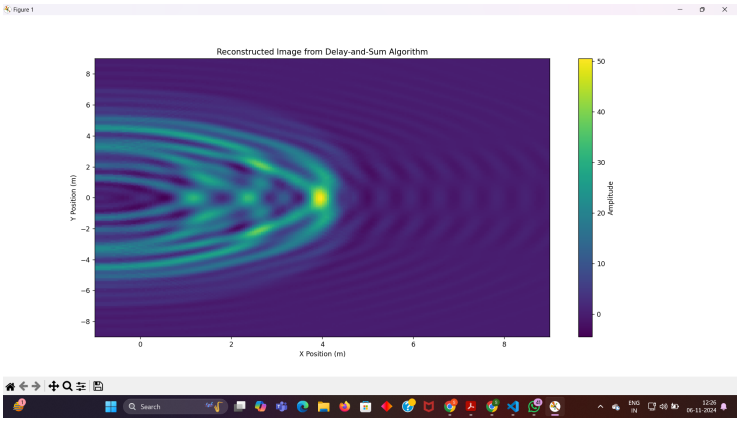

Figure 17: for mic outputs given by dataset RX3.txt