Sound Localization using Python

Siva Sundar, EE23B151

June 4, 2025

1 Objective

The aim of this project is to locate obstacles by analyzing sound wave data collected from multiple microphones. This is achieved through the Delay-and-Sum (DAS) algorithm, which utilizes time differences between the captured signals to estimate the position of obstacles.

2 Methodology

The methodology for this project is organized as follows:

1. **Single Obstacle Case Study:** Implement a sound wave plot for a known obstacle at (3, -1) and verify location detection using the DAS algorithm.
2. **Defining DAS Algorithm Function:** The DAS function takes in system parameters (e.g., speed of sound, sample spacing, source location) and outputs a delay-sum array and distance array for plotting.
3. **Maxima Detection Function:** This function identifies local maxima separated by a specified distance, enabling detection of multiple obstacles based on peak locations in the DAS result.
4. **Visualization and Results Display:** The final outputs, including obstacle locations, are plotted and the calculated coordinates are printed for clarity.

3 Results

The following table provides the calculated locations for obstacles using the DAS algorithm:

Test Cases	**Index**	**Obstacle Location**
Prof's system	(30, 22)	(3, -1)
rx2.txt	(30, 22), (21, 32)	(3, -1), (2.1, 0)
rx3.txt	(30, 22), (20, 32), (40, 39)	(3, -1), (2.1, 0), (4, 0.7)

Table 1: Calculated obstacle locations using DAS algorithm for test cases.

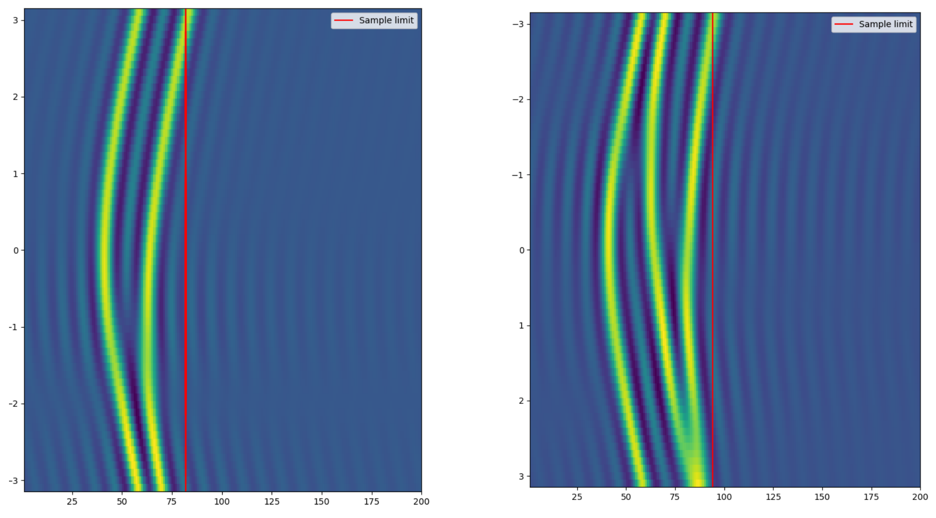
4 How to run the Code?

This **jupyter notebook** contains the python code which implements DAS algorithm to find obstacles for each case and also contains answers for **Q1** at the start and **Q5** at the end. To **run** the notebook, just press **run all**. Also, the text-files are **assumed** to be in the **same directory** as the notebook.

5 Answers to the Questions (Q2-6)

Q2: Does it make sense to reconstruct up to `Nsamp`? What value is more reasonable as an upper limit for the x-axis here?

Answer:
We only need to iterate till we could find an obstacle whose sound wave pattern resembles that of the **right-most wave**. The obstacle point would be to the left of this curve obviously, but we dont know this point. So iterating till (x, y) coordinate of the right-most wave's maxima (labelled in red) would be sufficient enough than iterating through '`Nsamp`'. But doing this causes the reconstructed image to appear blurred to the right of the rightmost obstacle.



(a) Data in 'rx2.txt' file

(b) Data in 'rx3.txt' file

Figure 1: The red line shows the 'x' iteration limit used in `DAS()`

Q3: The (x, y) coordinates corresponding to the maximum amplitude (yellow colour) is approximately (30, 22). Explain why this is the correct expected position for the given obstacle.

Answer:
In delay sum algorithm, we **assume** an obstacle bounded in the region defined by the conditions:

- $x \in [0, x\_max]$ (explained in **Q2**) and
- $y \in Y_{mics}$ (set of y-coordinates of mics).

The point is actually (29,21) which is the index (x, y) in the delay sum algorithm as Prof plotted the output using `imshow` which uses axes as array indices and inverted y-axis by **default**. (see docs).
As $x = 0^{th}$ index is at `dist_per_samp`, 29^{th} index is at 30 times this distance which is **3 x-units**.
For y coordinate, the number 21 represent the 22^{nd} mic (0 being the 1^{st} mic) and as it is given mic array is centered at origin with $\frac{1}{2}$`Nmics` mics above and below the origin, we can say 22^{nd} mic is at a location:

$$(22 - \frac{\texttt{Nmics}}{2}) \times \texttt{pitch} + \frac{\texttt{pitch}}{2} \approx -1 \text{ y-units.}$$

Hence, $(30, 22)$ index $\longrightarrow$ $(3, -1)$ xy-coordinate.

Q4: What is the maximum obstacle x- and y- coordinate that you can use and still have an image reconstructed?

Answer:
Each mic can detect up to `Nsamp` samples of the source wave (assuming an impulse wave). If an obstacle's peak arrives at a mic only after `Nsamp` samples, it won't be recorded. If this occurs for all mics, the obstacle **cannot be located** by the DAS algorithm, as its sound wave pattern cannot be captured with only `Nsamp` samples. Therefore, for DAS to function:

$$\frac{\text{Total Path Difference}}{\texttt{dist\_per\_samp}} < \texttt{Nsamp} \quad \text{for at least one mic}$$

$$\sqrt{x^2 + y^2} + \sqrt{x^2 + (y - y_{\text{mic}})^2} < \texttt{Nsamp} \times \texttt{dist\_per\_samp} \tag{1}$$

The DAS algorithm further assumes $y_{\text{obstacle}} \in Y_{\text{mics}}$, imposing an additional constraint. See Desmos visualization using `Nsamp` $= 64$ & `dist_per_samp` $= 0.1$.
Note: These arguments apply to finite waves like impulse waves. For sinc waves, which extend over $[-\infty, +\infty]$, mics can detect smaller peaks and still reconstruct an image, though it may be erroneous due to external noise. This is why results in Q5 are close even when the obstacle violates the condition.

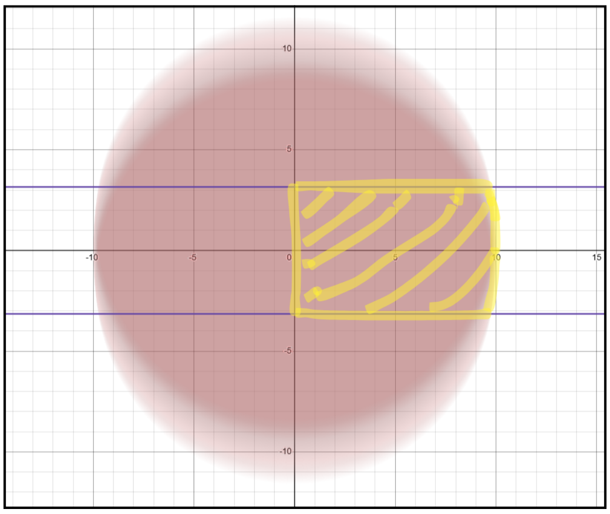


Figure 2: Obstacle can be found if it is placed in the yellow boundary

Q5: What happens if C is different - if C is decreased it looks like the image becomes sharper. Can you explain why intuitively?

Answer:
I'm using speed of sound C while creating the sound wave plot for a given obstacle. There, I defined a quantity called `time_per_samp` = `dist_per_samp`$/C$. As C decreases, time between samples increases which in turn **shrinks the sinc wave**. This causes the sampling to be **more precise**.
This can be explained using **Nyquist theorem of Sampling**:

- The sample rate is `time_per_samp`
- For the source, $\omega = 2\pi f$ = `SincP` (see definition of wsrc).

As C decreases, sample rate **increases** and hence, by Nyquist theorem, the image has **more resolution**. (See Fig. 6)

Q6: What happens if `Nmics` is increased or decreased? Do the experiments with `Nmics` $= [8, 32, 64]$ and `Nsamp` $= [50, 100, 200]$ (all combinations). Attach the resulting images.

Answer:
As discussed in **Q4**, sinc waves allow for image generation even if the obstacle lies beyond the limit set by inequality (1), though with reduced precision. For an obstacle at (3, -1), using different microphone and sample configurations, we generated the plots and predicted obstacle locations shown in Figs. 3-5 and Table 2. Two trends emerge:

- With fixed `Nmics`, increasing `Nsamp` initially improves x-coordinate accuracy. Once `Nsamp` satisfies inequality (1), the x-coordinate stabilizes regardless of further increases.
- With fixed `Nsamp`, increasing `Nmics` improves y-coordinate accuracy initially. When `Nmics` is large enough that the obstacle's y-coordinate falls within Y_{mics}, the y-value stabilizes with further increases in `Nmics`.

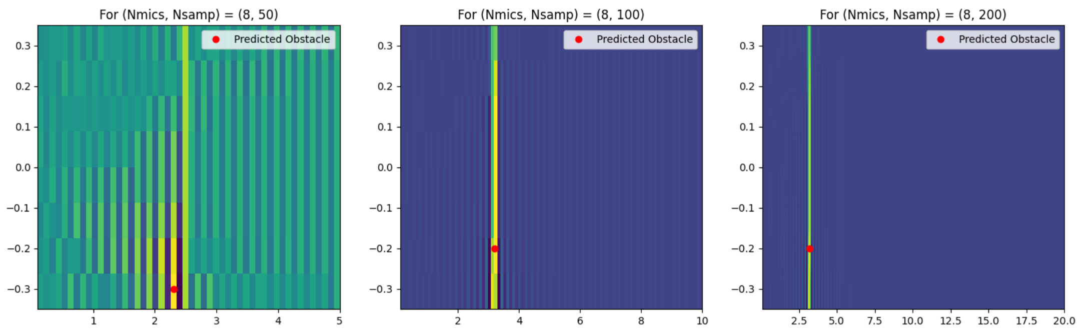


Figure 3: For `Nmics` = 8

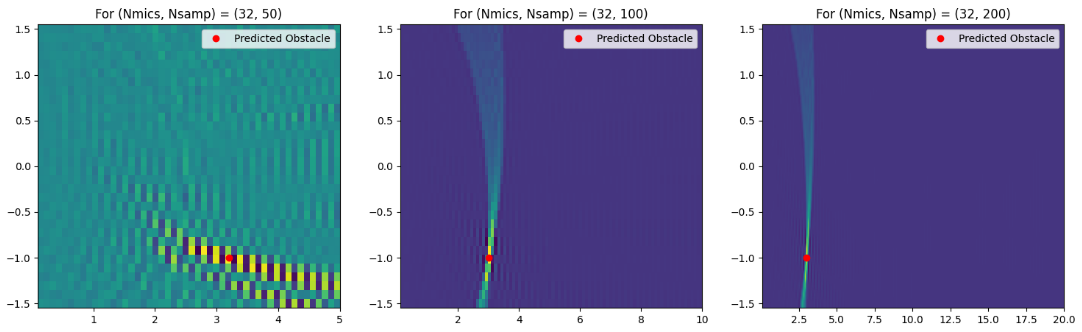


Figure 4: For `Nmics` = 32

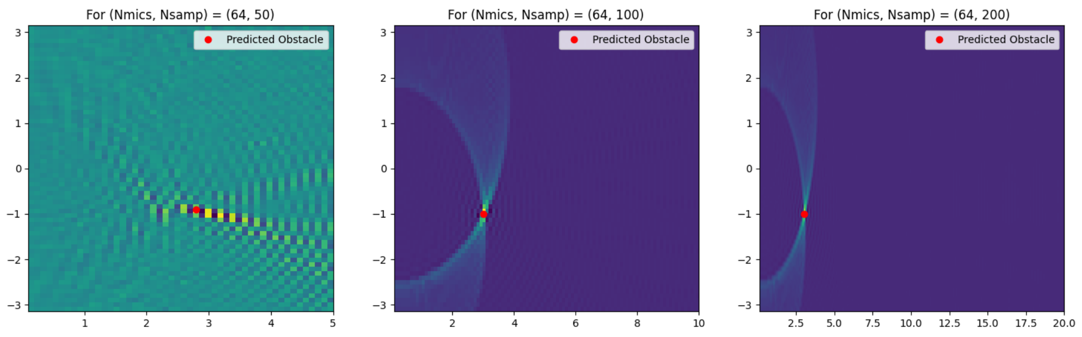
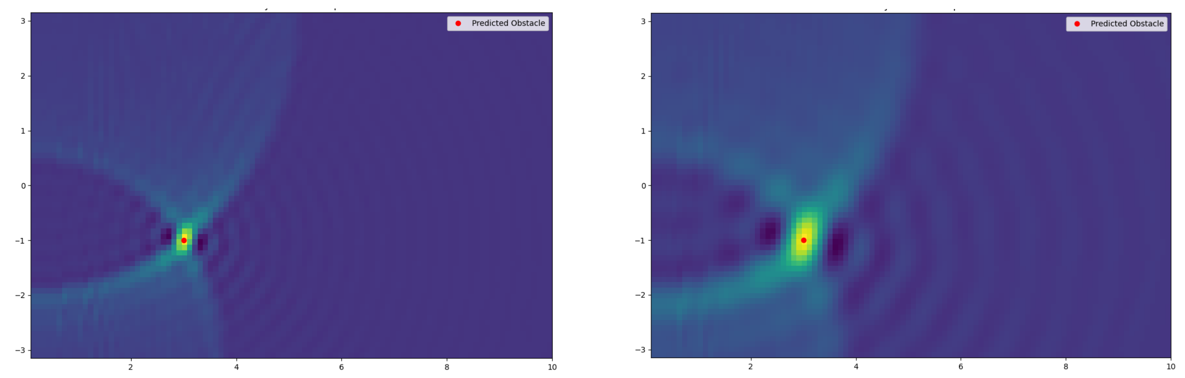


Figure 5: For `Nmics` = 64

	Nsamp		
Nmics	50	100	200
8	(2.3, -0.3)	(3.2, -0.2)	(3.2, -0.2)
32	(3.2, -1.0)	(3.0, -1.0))	(3.0, -1.0)
64	(2.8, -0.9)	(3.0, -1.0)	(3.0, -1.0)

Table 2: Predicted obstacle location for varying `Nmics` and `Nsamp` values.



(a) Reconstructed image for $C = 1$ (b) Reconstructed image for $C = 2$

Figure 6: Plots for **Q5**

6 References

[1] StackOverFlow: How to find maximum value in whole 2D array with indices ?

[2] Youtube: A gentle introduction to beamforming

[3] Matplotlib docs: Axes.imshow