Autonomous Underwater Vehicles: A From-Scratch Perception & Control Approach

Sreeganesh Valathara Rajendran

January 4, 2025

Contents

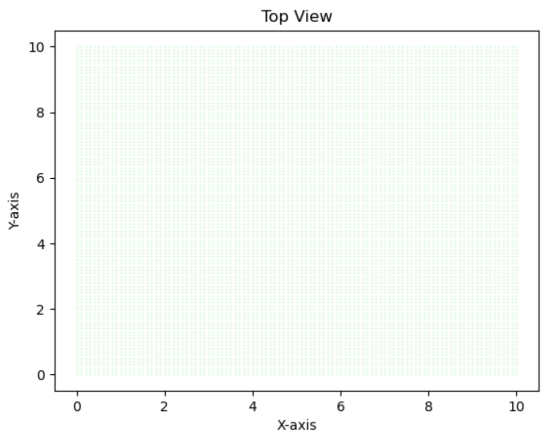
Chapter 1

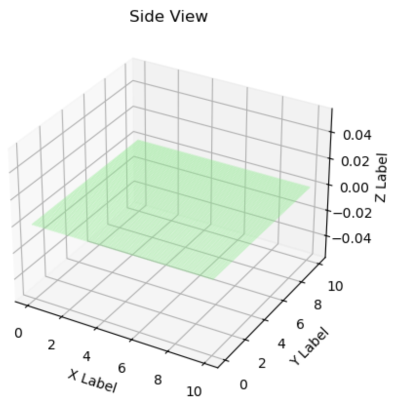
Setup

1.1 Topology Setup

The sea-floor is represented in a discrete manner. That is, the sea-floor is represented by a number of points that has some coordinate and some reflectance value. These two attributes are stored in two separate tensors as follows:

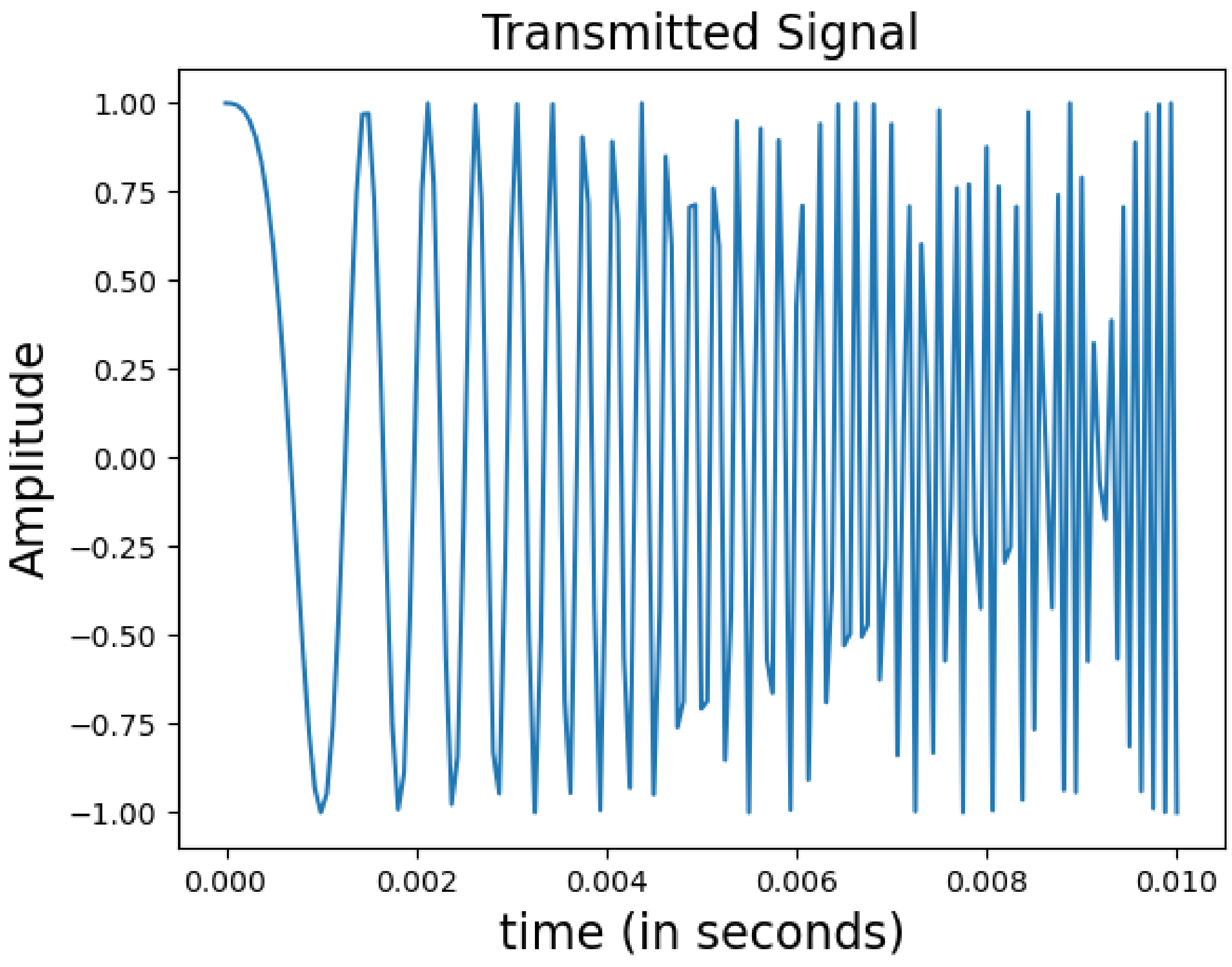
- *location tensor*: this tensor contains the location of all the scatters that are used to represent the sea-floor
- *reflectance tensor*: this tensor contains the reflectivity of the points representing the sea-floor.





1.2 Signal Simulation

The transmitted signal that we're using for this experiment is a chirp signal.


Transmitted Signal
Amplitude
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.000
0.002
0.004
0.006
0.008
0.010
time (in seconds)

Chapter 2

Software

2.1 Classes

2.1.1 Class: AUV

The following is the class definition used to encapsulate attributes and methods of the AUV entity.

```python
"""
"""
# packages/libraries
import numpy as np
import os
import pdb
import matplotlib.pyplot as plt
import scipy

# class representing AUV
class AUV:
    # init function
    def __init__(self,
                 location,            # current location of AUV [tensor]
                 velocity,            # velocity of AUV [tensor]
                 acceleration,        # acceleration of AUV [tensor]
                 pointing_direction): # direction in which AUV is pointed [tensor]
        """
        Initializing parameters related to AUV
        """
        # fundamental attributes
        self.location            = location           # current location of AUV
        self.velocity            = velocity           # velocity of AUV
        self.acceleration        = acceleration       # acceleration of AUV
        self.pointing_direction  = pointing_direction # direction in which AUV
            is pointed

        # add-on attributes
        self.projector_starboard = None  # projector to the right
        self.projector_portside  = None  # projector to the left
        self.projector_fbls      = None  # projector to the front

        self.ula_portside        = None  # ULA mounted on the left
        self.ula_starboard       = None  # ULA mounted on the right

    def summarize(self):
        print(">location           = \n", self.location)
        print(">velocity           = \n", self.velocity)
        print(">acceleration       = \n", self.acceleration)
        print(">pointing_direction = \n", self.pointing_direction)

    def update_timestep(self):
        """
        Updating the after each time step
        """

    def simulate_signal(self):
        """
        Aim: Simulate signals
        Note:
            > Project signals from projectors
            > Return signals from scatters
            > simulate signals received by ULAs
        """
        pass
```

2.1.2 Class: Scatter

The following is the class definition used to encapsulate attributes and methods of the scatterers.

```
"""
"""
# packages/libraries
import numpy as np
import os
import pdb
import matplotlib.pyplot as plt
import scipy

# class representing scatters
class Scatterer:
    # init function
    def __init__(self,
                 coordinates   = None,
                 reflectivity  = None):
        self.coordinates  = coordinates
        self.reflectivity = reflectivity

    # function: produce top view

    # function: produce side-view

    # function: project to angle
```

2.1.3 Class: Projector

The following is the class definition used to encapsulate attributes and methods of the projectors used.

```python
"""
Aim: Classes
Note:
    ULA: Class for Uniform Linear Arrays
    AUV: Class for Autonomous Underwater Vehicle
    Projector: Class for Projector
"""
# packages/libraries
import numpy as np
import os
import pdb
import matplotlib.pyplot as plt
import scipy
from Classes.Class_Scatterer import Scatterer

# class representing project
class Projector:
    # init function
    def __init__(self,
                 location          = None,    # location of projector
                 azimuthal_angle   = None,    # pointing direction of projector
                 elevation_angle   = None,    # pointing direction of projector
                 azimuth_beamwidth = None,    # sound projection between axis and
                     max
                 vertical_beamwidth = None): # sound projection between axis and
                     max
        """
        Init function
            - location:   Location of projector
            - direction:  direction of projector
        """
        self.location            = location
        self.azimuthal_angle     = azimuthal_angle
        self.elevation_angle     = elevation_angle
        self.azimuthal_beamwidth = azimuth_beamwidth
        self.vertical_beamwith   = vertical_beamwidth

    # subset signals
    def subset_scatters(self,
                        scatterers: Scatterer):
        """
        Aim: Take a set of scatters and subset them
        Note:
            > Takes in a tensor representing the points
            > subset the scatterers that are within FOV of current projector
            > returns the subset
        """
```

2.2 Function