Chapter 3

Acoustic Positioning System

The acoustic positioning system is the core of the iSBL-SF algorithm. In this system, an acoustic pulse is sent by a transmitter and recorded by receivers on the iSBL array. The time shift between the signals from each receiver is calculated, and these shifts are used to estimate the transmitter's position relative to the iSBL array using an acoustic propagation model. Chapter 3 describes how the orientation estimate of the platform is determined (and how dead-reckoning is introduced), and Chapter **??** details how Kalman filters are used to get the best position estimate by combining multiple data sources. This chapter describes only the acoustic positioning system, the central component of this thesis.

The first section in this chapter is dedicated to previous implementations of acoustic positioning systems, and how the iSBL approach differs from them. Next, the mechanical design of the receiver array is discussed. The electrical design of the acoustic positioning system, including the wiring diagram of the full iSBL array, is then detailed. The ultrasonic receivers receive their own section, where the design and manufacturing of the active band-pass filter PCB is explained. Then, the assembly of the full iSBL array is described, followed by the hardware and software design of the transmitter system. The final three sections are devoted to the software of the acoustic positioning system: how an acoustic pulse is detected and recorded, how the time shift between signals is calculated, and finally, how a position estimate is formed.

3.1 Previous Works

Acoustic positioning systems are not a new invention, and many different designs have been tested and implemented over the years. The rise of autonomous underwater vehicles has accelerated research into this field. In this section, the history of acoustic positioning systems is recounted, and the primary difference among these systems (the baseline length) is described. Lastly, similar implementations to this thesis are detailed.

3.1.1 History of Acoustic Positioning Systems

The first known underwater acoustic positioning system was deployed in 1963, when a short baseline acoustic positioning system was used to guide the bathyscape *Trieste 1* to the wreck of the US nuclear submarine USS *Thresher*. The system barely worked, only guiding the *Trieste* to the wreck once after ten unsuccessful attempts. The technology advanced as commercial applications became evident; oil and gas exploration in the 1970s was a primary driver for advancement. In 1998, a long baseline acoustic positioning system was used by a salvage company to navigate a debris field of a sunken WW2 Japanese submarine at 5240m deep. Current advancements in the field are primarily driven by the oil and gas industry, academic research, and defense industry applications [**?**].

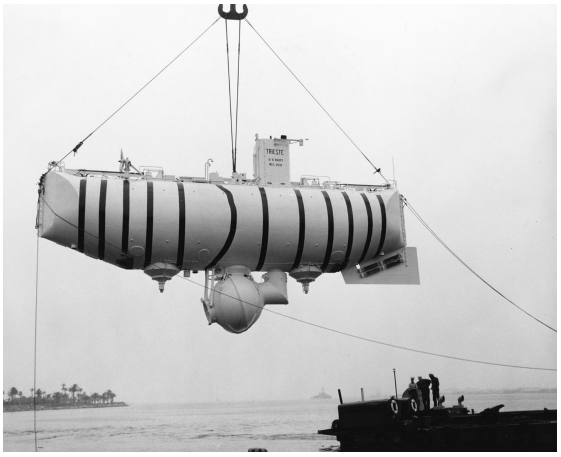

Figure 3.1: Bathyscape *Trieste 1*, which used the first short baseline acoustic positioning system to locate a nuclear submarine wreck [**?**]

3.2 Active Band-Pass Filter Design

This implementation is designed to work with 40kHz audio signals. This frequency was chosen for a few reasons:

- The above-water frequency should be similar to the planned underwater system's frequency to allow for simple future implementation
- The audible range should be avoided to minimize disturbance to marine animals (see Section **??** for more details on the underwater system extrapolation)
- Lower frequencies require lower ADC sampling rates to record a single wave
- Lower audio frequencies attentuate much less rapidly in seawater compared to higher frequencies (see Figure **??**)
- 40kHz is a very common frequency for air-based ultrasonic transducers, and fairly common in water-based ultrasonic transducers

```
message_int ++;      // pick the next number
message_int %= 256;   // between 0 and 255 (8-bit length),
message_char = (char)message_int;   // and save it as a char

// save in the last 8 bits of message_string
for(int i = 8; i > 0; i--){
  message_string[14-i] = (message_int >> (i-1)) & 1;
}

```

```
  message_string[0] = 1;  // initialize the message with a header
  message_string[1] = 1;
  message_string[2] = 1;
  message_string[3] = 1;
  message_string[4] = 0;
  message_string[5] = 0;
```

Table 3.1: Truth table for motor driver

Logic level, InA	Logic level, InB	Voltage, Out1	Voltage, Out2
0	0	0V	0V
0	1	+12V	0V
1	0	0V	+12V
1	1	+12V	+12V

$$\mathbf{q} = q_0 + iq_1 + jq_2 + kq_3 \tag{3.1}$$

$$\mathbf{q}_n = (1-\gamma)\mathbf{q}_{acc/mag} + \gamma(\mathbf{q}_{n-1} + \delta t \times \mathbf{q}_{gyro}) \tag{3.2}$$

Algorithm 1 Hooke-Jeeves Search Algorithm in $\mathbf{n}$ Dimensions

Require: InitialPosition, MaxIterations, MinResidual, MinSpacing, ScaleFactor, InitialSpacing
Ensure: OptimizedPosition

$position \leftarrow InitialPosition_{\mathbf{n}\times 1}$
$newPosition \leftarrow InitialPosition_{\mathbf{n}\times 1}$
$residual \leftarrow \infty$
$spacing \leftarrow InitialSpacing \times ScaleFactor$
$iteration \leftarrow 0$
$newDirectionNeeded \leftarrow false$
$direction \leftarrow \mathbf{0}_{\mathbf{n}\times 1}$
while $iteration < MaxIterations$ **and** $residual > MinResidual$ **and** $spacing > MinSpacing$ **do**
    if $newDirectionNeeded$ **then**
        $spacing \leftarrow spacing/ScaleFactor$
        for each dimension $\mathbf{d}$ in $\mathbf{n}$ **do**
            $direction \leftarrow \mathbf{0}_{\mathbf{n}\times 1}$
            $direction(\mathbf{d}) \leftarrow 1$
            $testPosition \leftarrow position + direction \times spacing$
            $residualList(2\mathbf{d}) \leftarrow$ CalculateResidual($testPosition$)
            $testPosition \leftarrow position - direction \times spacing$
            $residualList(2\mathbf{d}+1) \leftarrow$ CalculateResidual($testPosition$)
        end for
        if $\min(residualList) < residual$ **then**
            $direction \leftarrow \text{argmin}_{direction}(residualList)$
            $residual \leftarrow \min(residualList)$
            $position \leftarrow position + direction \times spacing$
            $newDirectionNeeded \leftarrow false$
        else
            $newDirectionNeeded \leftarrow true$
        end if
    else
        $prevResidual \leftarrow residual$
        $newPosition \leftarrow position + direction \times spacing$
        $residual \leftarrow$ CalculateResidual($newPosition$)
        if $residual < prevResidual$ **then**
            $position \leftarrow newPosition$
        else
            $residual \leftarrow prevResidual$
            $newDirectionNeeded \leftarrow true$
        end if
    end if
    $iteration \leftarrow iteration + 1$
end while
return $position$

Bibliography