Understanding Complex 4D Sonar Operating Conditions Using Virtual Environments

Ross Young, Jay Cooper
Systems Engineering and Assessment Ltd
Bristol, United Kingdom
ross.young@sea.co.uk, jay.cooper@sea.co.uk

ABSTRACT

The properties of the ocean environment have a significant impact on the performance of the sonar systems used on both surface ships and submarines. The types of data that need to be considered range from ocean temperature and salinity to the position of wrecks on the seabed and current marine traffic. These properties can dramatically change a sonar system's ability to detect a given target, with certain conditions potentially rendering sonar targets completely invisible in what is known as a "shadow zone" or obscuring them within environmental noise.

Understanding how these environmental properties interact in three dimensions and how they evolve over time is essential for developing effective anti-submarine warfare (ASW) strategies, or conversely for developing an evasion strategy for a submarine. A software system has been developed that presents a variety of different ocean data layers in a virtual environment that can be viewed either via a desktop application, a virtual reality (VR) headset or using a holographic display.

Different data presentation styles are adopted for different datasets to best display the relevant information to the user, with the user able to toggle multiple layers on or off and adjust various layer properties to interrogate details of the datasets. An acoustic ray-tracing model has been integrated into the system, allowing detailed propagation paths between two points in the environment to be computed, considering the specific oceanographic properties between two selected points in the model.

This paper describes the software development approach used to create the virtual environment and describes how different data types are displayed. The system has also been trialed ashore with current multinational naval operators to assess the usability of this tool on future operations. A summary of their feedback is presented along with a discussion of future developments.

ABOUT THE AUTHORS

Ross Young is an APMP qualified Project Manager with an experience of working on Research & Simulation projects which includes Virtual Reality programs such as the Multi Environment Display. Ross also has over 10 years of tactical planning and implementation within Royal Navy Operations Rooms, working closely with Ship Commanding Officers and Principal Warfare Officers as a Tactical Aircraft Controller prior to joining SEA.

Jay Cooper is a Software Developer with 5 years Software Development experience using C#, Unity and .NET. Jay has previously worked on both VR and Non-VR Unity applications as well as working with SQL databases.

Understanding Complex 4D Sonar Operating Conditions Using Virtual Environments

Ross Young, Jay Cooper
Systems Engineering and Assessment Ltd
Bristol, United Kingdom
ross.young@sea.co.uk, jay.cooper@sea.co.uk

INTRODUCTION

Anti-submarine warfare (ASW) is extremely challenging. The objective is to detect hostile sub-surface vessels within an environment that is filled with ambient noise from wind, waves and shipping, and the output from the sonar sensors may be cluttered with noise from hundreds of civilian or friendly vessels that are not of interest. At the same time, the hostile vessels will be minimizing their noise whilst also trying to conceal themselves within the environment.

To add to this complexity, the propagation of sound within the ocean is itself extraordinarily complex, depending on the sound velocity profile (SVP), bathymetry profile, seabed characteristics and surface wave conditions. All these parameters will vary depending on the location in the ocean and the direction you are looking in, while the SVP and surface waves will also vary as a function of time.

When planning ASW operations it is necessary to have a clear understanding of the complete ocean environment to position your sonar assets for maximum effectiveness. Conversely, if you were planning a submarine operation and wished to evade detection, you would need a similar understanding of this ocean environment. At present, the oceanographic data is often presented to users as a series of 2D charts, but this makes it difficult to build a complete understanding of the underwater environment, which may be three- or even four-dimensional, as described above. Presenting this data in a more easily comprehensible format would be of significant advantage to personnel in the undersea battlespace.

The objective of this work was to build a digital environment that allows multiple data layers to be displayed and interrogated in a single 3D world view via either a desktop application, a VR headset, or a holographic table. These displays were then trialed to identify the formats and features that are most beneficial to operators.

The data types displayed in the model are described in **Model Data**, while the software architecture and implementation are described in

MED SYSTEM Development. This is followed by the results of the user evaluations and the conclusions, which include plans for future work.

MODEL DATA

The objective of this work was to understand the extent to which a 3D display can help users to understand complex 3- and 4-dimensional datasets. This section describes the list of data types that have been included in the current model, and why they might be of interest to people engaged in ASW operations. This list is by no means exhaustive, and one of the outputs of the user evaluations is to identify which layers are most useful and whether there are any additional layers that should be included.

It should also be noted that in this iteration of the model all the data layers are generated from static datasets, rather than dynamically obtaining the data on demand. For this reason, all illustrations shown in this paper are for the region of the north Atlantic between Scotland and Iceland.

Bathymetry

The most obvious feature of the ocean that needs to be included in the model is the ocean depth. The water depth inevitably has a significant impact on the acoustic propagation, with shallow water leading to propagation paths with multiple reflections from both the surface and seabed. In deep water the SVP becomes much more significant, leading to complex, refracted sound propagation paths (See Figure 1).

Alongside the depth, the seabed gradient is also of interest for ASW operations, as sharp seabed gradients can create regions that are difficult to penetrate with sonar. The seabed gradient is computed directly from the bathymetry data using the second order central difference algorithm implemented within the gradient function of the Python NumPy package. Given that the bathymetry points are provided on a regular latitude / longitude grid, it is necessary to first convert the positions to meters so that the seabed gradient is correctly estimated at all locations. This is done using a local flat-earth approximation using a radius of 6371 km, which is the WGS84 ellipsoid equal volume approximation for the earth's radius.

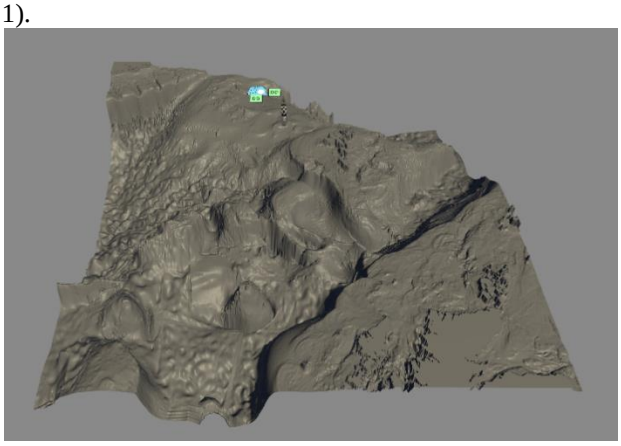

Figure 1. Bathymetry

Oceanographic Data

The key information for understanding the acoustic environment is the oceanographic data, specifically the sea temperature and salinity. This data is used to compute the SVP in the ocean which, in turn, drives the acoustic propagation. Changes in sound speed with depth lead to refraction of the acoustic propagation paths, with the sound bending away from regions with high sound speed, towards regions with low sound speed. If there is a local minimum in the SVP this creates what is known as a sound channel, where sound is trapped at a specific depth and can therefore propagate over very great distances. Alternatively, an SVP that decreases with depth will lead to all sound paths bending towards the ocean floor, creating shadow zones.

In our model (see Figure 2) the oceanographic data is obtained from the Copernicus Marine Service (CMEMS) [Ref 1] that provides an API for requesting temperature and salinity data for a given ocean region and forecast period. The data is provided in netCDF format as a 4D array which is function of time, depth, latitude, and longitude. The data is provided at a spatial resolution of 0.083° × 0.083° and at 50 depth values, with the forecast data at one-hour intervals from the current time for a period of 10 days. The sound speed is calculated from the temperature and salinity data using the international standard UNESCO algorithm due to Chen & Millero [Ref 4]. This sound speed data may be viewed directly within the model but is also used as an input to the acoustic propagation model (see Section **Propagation Loss**).

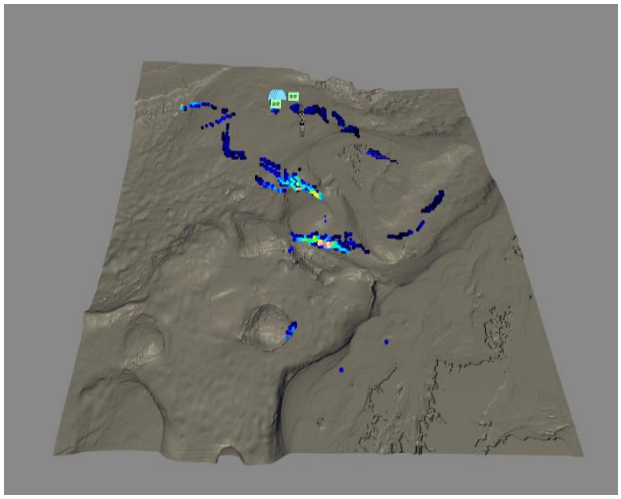

Figure 2. Ocean Fronts

Like the atmosphere above the ocean's surface, fronts develop below the surface where there are rapid changes in ocean

temperature. These ocean fronts are of interest to sonar operators as the abrupt change in properties often creates a barrier to acoustic propagation that can significantly limit the range and effectiveness of a sonar sensor. In our model the temperature gradient is calculated on each depth plane within the CMEMS data, with the data filtered to return only those points where the gradient exceeds a predefined limit. A 3D point cloud of these points is provided to the model at each forecast time, allowing the user to inspect the shape and location of the ocean fronts, and understand their evolution with time.

Seabed Data

The nature of the seabed can have a significant impact on the propagation of sound in the ocean. A hard, rocky seabed will reflect a substantial proportion of the sound, while a soft, muddy seabed will reflect little. The thickness of any seabed sediments will also impact the acoustic propagation. The sediment type is included within the model as a map of Bottom Sediment Type (BST) classification, with the data being obtained from the UK Hydrographical Office. The data is displayed using a colourmap where each different BST group is assigned a color.
Another seabed feature that is of interest to sonar operators is the location of wrecks. When using active sonar, the operator is primarily interested in echoes from unexpected directions, that do not coincide with known ship locations. When such an unexpected echo is received, one of the first things that is checked is to establish whether there is a wreck on the seabed at the bearing from which the echo was received. Having a clear knowledge of the locations of shipwrecks can aid users to make quicker evaluations of sonar data and may also assist in the planning of sonar operations.

Wreck positions are currently included in the model as a geotiff that is generated from the publicly available list of wrecks published by the UK Hydrographic Office [Ref 2], where wreck positions are drawn using the Mil-Std-2525d notation [Ref 3]. The list contains a large number of wrecks, and this data has been filtered to only include wrecks whose stated length is greater than 20m. For the region of interest in the North Atlantic used in this study, this left 169 wrecks.

Sea Surface Data

There are various sea surface features that may be of interest for ASW operations, two of which are included within this model: surface wind speed (see Figure 3) and shipping density. The primary impact of both features is on the level of background, ambient noise that will be present in the sonar data. Understanding how these regions of high noise interact with other environmental conditions, and how they evolve with time due to meteorological effects is key to effective planning of ASW operations. Representative surface wind data was supplied by the Joint Operations Meteorology and Oceanography Centre (JOMOC). This includes a 20-day forecast of the surface wind speed over a region of the North Atlantic at a relatively low resolution. Wind direction is not included in the current model.

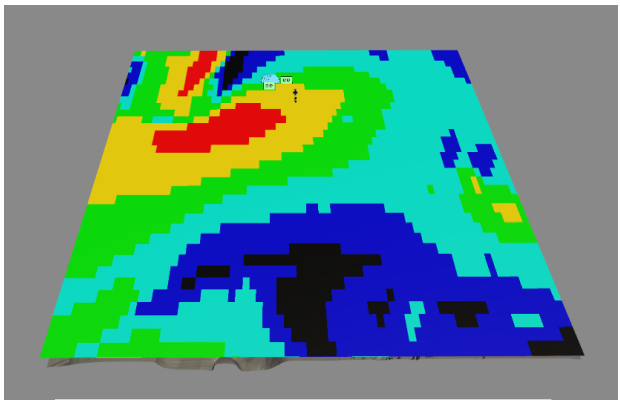

Figure 3. Surface Wind

Shipping density data (See Figure 4) is obtained in geotiff format from European Marine Observations and Data Network (EMODnet) who publish an annual data map of average shipping density, as obtained from Automatic Identification System (AIS) data. The data is provided separately by month, and by twelve different vessel categories, but for the purposes of providing a general understanding of the shipping density data, our model simply includes the annual average for all vessel types.

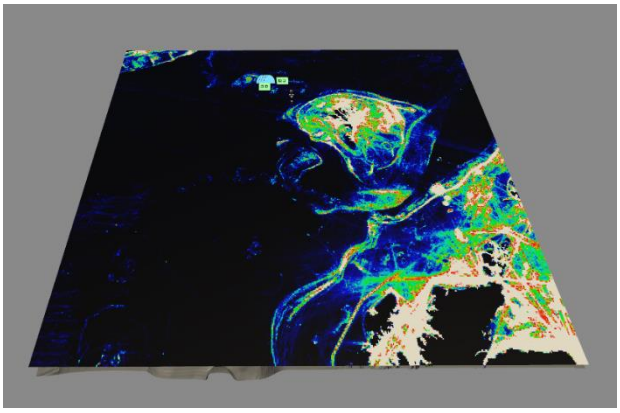

Figure 4. Shipping Density

TACTICAL INFORMATION

The model includes a selection of layers that are included to aid tactical decision making and planning. The first, and most obvious of these is the inclusion of nautical charts to provide the familiar navigational information used in naval operations. In the current model, the chart is provided as a single image file for the simulated region of the North Atlantic.

The model also includes a tactical picture layer that displays the current position of known contacts within the simulated region, as well as historical course information. This information is supplied as a real-time Data Distribution Service (DDS) data stream from external simulation that aims to replicate the combined outputs from a combat ship's various sensor systems. The contacts are displayed in the model using a simplified version of the Mil-Std-2525D [Ref 3].

As an aid to the planning process the model also includes the ability to add notes to the display. These notes are generated using a freehand drawing tool and are transcribed onto the 2D plane at the sea surface.

Processed Layers

Some of the data layers within the model are the result of significant processing of the environmental data, which is performed outside of the model. These layer types are described below.

Ambient Noise

Ambient noise in the ocean can have a major impact on the effectiveness of sonar sensors, with high background noise levels making it extremely difficult to detect other vessels; particularly those that may be trying to evade detection. SEA have developed software that is able to aggregate ambient noise sources within the ocean - including wind, waves shipping etc. – and generate an ambient noise forecast that is a function of position, depth, frequency, time, and look-direction. This is known as the Ambient Noise Prediction System, or ANPS.

This data has been incorporated into the model so that a user can inspect the ambient noise levels at certain frequencies and depths of interest that cover the operating envelopes for particular sonar sensors. In the current version of the model only the omnidirectional noise output from ANPS has been included (see Figure 5).

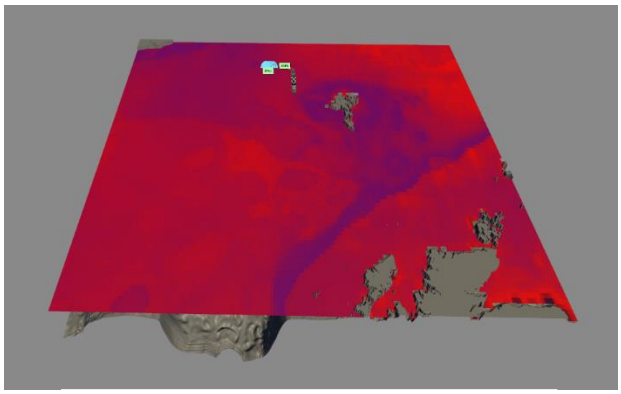

Figure 5. Ambient Noise

Propagation Loss

As previously described, the propagation of sound in the ocean is complex, and, even with all the environmental data to hand, it is not easy to sense whether sound will travel from one location to another. To allow detailed point-to-point acoustic propagation paths to be evaluated within the model, an acoustic ray-tracing model has been integrated into the system to compute the paths between two selected locations and generate a propagation loss map.

The calculation of the propagation loss is performed in two steps. The first step is to identify the start and end points for the propagation calculation and extract the environmental data between them. This generates a 2D slice of data that contains bathymetry, sound speed, seabed type and surface wind speed as a function of range. These data sets are linearly interpolated from the existing datasets along a great-circle path between the two points. In this version of the model the entity defined as the ownship is always the starting point for the propagation calculation, and the user then selects another entity within the model as the end point.

Once the data along the 2D slice has been extracted the information is passed to an acoustic ray-tracing model. This model computes a high-resolution fan of acoustic rays that propagate out into the environment, and then combines the ray data at a grid of points to compute a map of the propagation loss (see Figure 6).

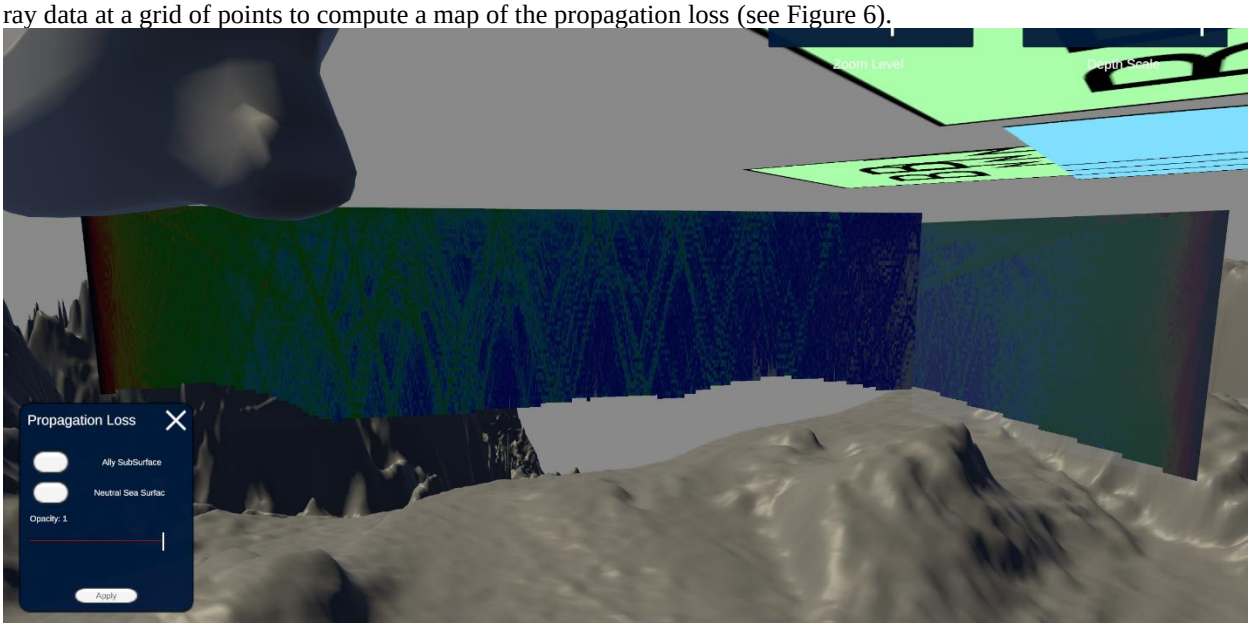

Figure 6. Propagation Loss

MED SYSTEM DEVELOPMENT

The objective of this work was to evaluate the benefits of different display formats (desktop, VR, Holotable) in aiding a user's ability to understand complex 3D and 4D datasets. The software was therefore developed in a way that made it possible to create a version of the same application that would run on all platforms. The architecture and implementation approach for this development is described below.

Architecture

The Multi Environment Display (MED) application is split up into three components. 3D display, Python server and ray-tracing server. Each component has its own unique use and is designed to be compatible with other software. This is done so that we can write more custom software that can interact with these components. The VR, desktop and Holotable applications are separate components that can all connect to the same Python server. Doing this also removes added complexity of integrating Python into Unity. Time was spent researching this and, in the end, seemed unnecessary. Creating the Python server as reusable software was the obvious choice. Any programming language

will have a way of performing Uniform Resource Locator (URL) data requests and the JavaScript Object Notation (JSON) JSON format is most common.

The 3D display does exactly as it should. Display data. Data is sent into a converter function that assign latitude, longitude coordinates with a data value. We call this the LatLonDataConverter. The converter will iterate over returned 2D, 3D and 4D data sets to produce a LatLonDataConverter<t> object. The T being the selected layer type. This object is then put into a nested for loop that assigns an RGB color based on the value. It catches the lowest and highest value so that color is evenly spread in a specified range. Depending on the data set, the color array is applied to different textures. There are multiple ways MED can display data.

- 2D slice
- 2D projected
- 3D point cloud
- 3D entity position

Another part of the application is the ray-tracing server. This manages all complex ray-tracing calculations. This is managed separately as it is primarily written in C++ and used across other applications. Only the Python server connects to raytracing as it needs the output data that ray-trace produces in its propagation loss calculation.

Data Servers

There are two servers that run parallel to the main 3D application. The main server is written in Python and uses a web framework package called Flask. Flask provides functionality for defining endpoints that can be queried by the display. Each request packages the queried data set into a JSON object which is then sent to the display. The display converts the JSON object into a custom C# object to be used for visualization. Endpoints can be queried with parameters to obtain a custom slice of data from a data set. This is primarily used when calculating propagation loss.

Propagation loss modelling is performed using a separate server application, which runs the range-dependent acoustic ray-tracing model in response to an http get request. This request comes from the Python Server, which interpolates the environmental data along the great circle path between the two points selected in the display and sends this within a request to the ray-tracing server.

The returned object contains propagation loss values on a vertical grid of points between the two selected entities. The resolution of this grid is currently hard coded within the Python server application to be 10m for depth and 100m for range. For long range propagation the ray tracing calculation can take a significant amount of time to be rendered on the display (up to around 2 minutes). This is largely due to the data being passed around in an expedient but inefficient manner, and the underlying ray-tracing calculation only takes a few seconds. Any future implementation would look to replace the data transfer method to improve the speed.

Development Environment

The 3D display is powered by the Unity game engine. This game engine is used due to its ease of use and platform compatibility. There are currently three versions of MED: Desktop, VR and Holo (see Figures 7 and 8). There are plans to make a fourth version for Augmented Reality (AR).

Unity projects are made up of scenes. Each MED version has its own scene. The user experience (UX) needs to be accessible on every platform. A mouse input does not work when running the application in VR for example. A lot of time was spent designing and implementing the UI controls so that navigating the software was intuitive for the interface being used.

Virtual Environments

Part of our research is to find the most effective way of displaying complicated data to an end user. To do this, multiple versions were created to evaluate the advantages and disadvantages of each display.

Desktop
Desktop MED is designed to be the base version of MED. There are no external devices or software needed to run. Runs on most systems that have a graphics card installed and is primarily used for demos to show off its capability. As most software nowadays is displayed on 2D screens as apps, it seemed like a logical choice to make this as a baseline project.

Virtual Reality
The VR version is built using Unity's Extended Reality (XR) toolkit. An XR rig is added to the scene which runs scripts to initialize the XR runtime and detect XR hardware. XR is the combination of VR and AR. The current VR headset of choice is the Oculus Rift S. This is due to its use in previous VR projects. The headset comes with the software Meta Quest Link and will not work without the software running. Meta Quest Link auto detects a plugged-in headset and goes through a multi-step process to set the environment up correctly.

Developers can create a mock XR rig that works with keyboard and mouse. This makes the testing process much easier as it does not require a headset connection. This was important when creating the UI and controls for navigating the software. Development was done in iterations where the software was being ran multiple times. Having to constantly put on and remove the headset would be uncomfortable and time consuming. Once development had completed a cycle, a proper VR run through was done to ensure the software worked as intended.

To aid in the development of the VR controls, an API reference guide on Unity was available. The guide was useful when setting up the XR rig as well as finding where to create the VR button interactions. The guide can be found here: https://docs.unity3d.com/Manual/XR.html.

Holographic Table
The holo-graphic table is made by a company called Axiom and is powered by a PC and plugged into a table with projectors and a screen. It comes with two sets of AR glasses and two wands (as well as some spectator glasses). The glasses enable the user to see the application in 3D while the wands are used to interact with the software. The wand is used to interact with the software via pointing and clicking.

A holo-table developer guide gives instruction on how to import the holo-table software into a new or existing project. The holo-table software is compatible with both Unreal Engine and Unity. Development for MED had already begun in Unity, so we chose the Unity package. Setting up the environment to be compatible with the holo-table consisted of adding the holo-table prefab into the project and selecting the "Holo-Table" option on a configuration file.

The holo-table prefab contains a 3D model of the table. This table mocks the real-world table so any objects placed inside the prefab, will be displayed, when going into play mode, the software runs as it would in real life. This was fantastic as developers could iteratively evaluate their code without needed to package up the software and export onto the real table. The prefab also contains a glasses and wand object that activates in play mode so developers can assess user interaction seamlessly. Once ported over to the table, software (for the most part) worked well. There were some minor adjustments made after testing on the table due to the size and feel of the table. Notably size and position of text for a better reading experience,

USER EVALUATION

One of the challenging factors of the software design was the viewport of the AR screen. At sharp angles, 3D models can be cut off. This meant that objects cannot be placed too far high up in world space. During development, an idea for displaying SVP data at a coordinate was to put it on a vertical graph with an arrow pointing down to the click position.

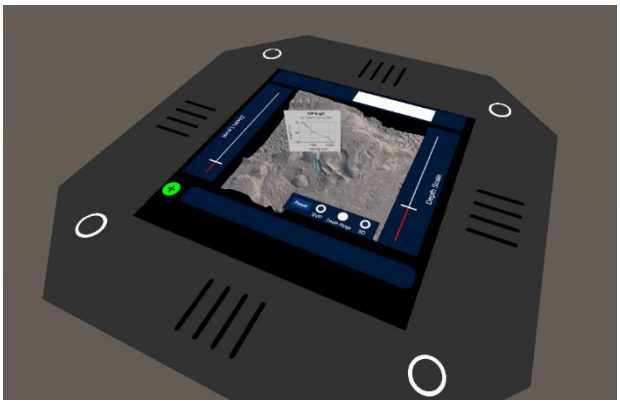

Figure 7. Holo Version SVP Angle 1

This would allow the user to better link the graph to the specified point. Originally, this was a promising idea to help show off the 3D nature of the table. However, this ended up being difficult to get right as multiple tests of the graph showed that at certain locations, the graph would be hard to read and end up being cut in half at distance. It was noted during a demonstration that the graph was still hard to read. A proposed fix was the make the graph 2D instead while keeping the arrow at the coordinate still visible. It seems that currently, the holo-table is better at displaying depth into the table, rather than height above the table.

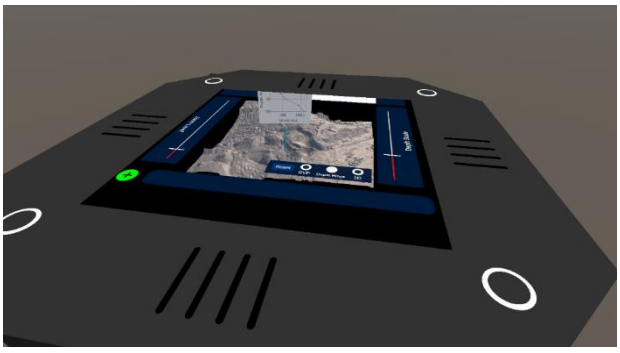

Figure 8. Holo Version SVP Angle 2

CONCLUSIONS

Future Developments

To create an immersive experience, MED VR allows the user to view the 3D map in whichever way they desire. Utilizing VRs potential allows the user to get a better depth perception of their surroundings. Future development for this work is to introduce networking so that other users can join in the same session and can communicate effectively. Using VR controllers, it allows the user to tag and mark areas of interest as well as adjust the scale and position of the map and data panels.

Augmented Reality

The implementation of this project in AR highlights the portability and intuitive nature of the software. Unlike traditional systems, AR does not require external camera setups, and the use of controllers is optional. This seamless integration of software and the real world exemplifies the concept of mixed reality. In theory, a user (or multiple!) will wear an Oculus Quest 3 headset with the software running. The user would see a 3D map of the underwater terrain around them and be able to point and click to manipulate the different layers to get a favored view of what they would like to see. The user interface would work both with and without controllers, hand gestures being the new UX

challenge. It would be interesting to see if a user could easily and more importantly, intuitively use the software only using their hands. With the future of AR hardware still evolving, it will be interesting to see how seamless mixed reality can be.

REFERENCES

[Ref 1] Global Ocean Physics Analysis and Forecast, E.U. Copernicus Marine Service Information (CMEMS). Marine Data Store (MDS). DOI: 10.48670/moi-00016

[Ref 2] UK Hydrographic Office Marine Data Portal, Wrecks and Obstructions, https://datahub.admiralty.co.uk/portal/apps/sites/?_gl=1*vq5uag*_ga*MTAxODg2NzA2Ni4xNjk4MDYxNDcw*_a_8PTW8GJL1R*MTY5ODA2MTQ3MC4xLjEuMTY5ODA2MTQ4MS4wLjAuMA..#/marine-data portal/items?tags=GlobalWrecks

[Ref 3] MIL-STD-2525D, Department of Defence Interface Standard, Joint Military Symbology, 10 June 2014

[Ref 4] Chen, C-T. & Millero, F.J. (1977), Speed of sound in seawater at high pressures, *J. Acoust. Soc. Am. 62*(5), 1129 1135