

UNDERSEA COMMAND AND CONTROL VISUALIZATION

Richard Shell and Lauren Mathews
Naval Undersea Warfare Center Division Newport

Robert King
Naval Research Laboratory

Fernando Das Neves
Virginia Polytechnic Institute

ABSTRACT

This research investigates the creation and integration of a comprehensive three-dimensional submarine undersea battlespace visualization (SUBV) with existing platform fire control systems to enhance the weapons fire control decision-making process. Using 3-D visualization to represent undersea environments, mission planners and command and control (C&C) personnel can gain a tactical advantage by reducing the excessive cognitive demand currently placed upon the user through the exploitation of visualization and interaction methods.

A discussion of the basic areas of research and results to date is presented. The focus of this paper is the methods used for bounding the complex regions in which targets can be found, methods for presenting received (passive) towed array data in a 3-D display, preliminary results from comparative testing of SUBV and current fleet capability, and a look at future efforts (i.e., transition to the fleet via the Future Naval Capabilities (FNC) program).

ACKNOWLEDGMENTS

The authors thank the many people who have supported this project. Particular thanks go to John Baylog of the Naval Undersea Warfare Center (NUWC Division Newport) for providing the acoustic model that formed the foundation for the model used in this project; Mike Medeiros (NUWC Division Newport) for his generous gifts of code, for sharing his experience, and for his many other technical contributions to advance the project; Marcus Graham (NUWC Division Newport) for his technical assistance, particularly on the analysis of the measurement formation; Michael Walsh (NUWC Division Newport) for his support and original ideas in 3-D target tracking; and James Kelly (NUWC Division Newport) and Ann Silva for promoting this interdepartmental and multi-institutional effort. The authors would also like to thank Ron Kriz (VT) for coordinating the support provided by Virginia Tech. We wish to extend special thanks to Mr. Paul Quinn and Dr. Larry Rosenblum of the Office of Naval Research for their continuing encouragement and support for this new and exciting research in 3-D visualization.

1. INTRODUCTION

1.1 PROBLEM

A significant issue in today's Navy is how data and information can be operationally integrated in naval combat systems to increase battlespace awareness and speed of command at both the platform and force levels. For example, decision makers currently develop a "mental model" of the battlespace and of a target's location by assimilating data from two-dimensional (2-D) displays and paper plots. In highly dynamic and/or complex environments, the cognitive processing (fusion) required to extend 2-D representations to a third dimension yields target solutions that are often untimely, inaccurate, or both.

The Naval Undersea Warfare Center (NUWC) has been tasked by the Office of Naval Research (ONR) to evaluate the feasibility of three-dimensional (3-D) rendering of complex, tactical information in a Navy-relevant domain. Partnering with NUWC for this project, entitled "Visualization for Multiwarfare Planning and Execution," is the Naval Research Laboratory (NRL), and Virginia Polytechnic Institute ("Virginia Tech" (VT)).

The specific objective is to assess the potential of 3-D visualization to enable rapid and/or accurate assimilation of complex information in a passive target-tracking context. The potential payoff from this work is faster and/or more accurate decision-making and improved planning and decision aids both from a network-centric and platform-centric perspective.

1.2 APPROACH

Current barriers to effective fleet command and control (C&C) include: (1) an excess of information, (2) increased operator workload, (3) inadequate time to interpret data for decision-making, and (4) increased complexity of the data. This research project represents an attempt to exploit 3-D rendering coupled with high-fidelity acoustic models to trade "cognitive data fusion" for "visual data fusion."

The approach taken in this project was to select a well-understood, familiar problem (submarine passive target tracking in shallow water via conical angles) to develop a means to assess the "value added" of a 3-D representation of the problem using the current submarine capability (CCS Mk2) as a baseline, and to make this assessment using fleet operators.

When working with towed array conical angle data today, target motion analysis (TMA) operators view and process only a single 2-D depth slice of an ideal 3-D conical angle, making underwater "target localization"¹ a cognitively intensive process. Probabilistic estimators are often not reliable because the scenarios do not always lend themselves to an observable solution,

¹ The term "target localization" used throughout this paper connotes minimizing the region in which a potential target is located.

thus leaving the tasks of interpreting data and developing target localization to the operator. A conical angle represents the angle at which energy from a target was received at the horizontal array towed by ownship. The information the conical angle provides is truly 3-D and informs an operator that a contact may be located at anywhere on the surface of the “cone.” The conical angle in the real world seldom looks like the idealized cone (see figure 1) and is often grossly distorted (see figure 2). The causes of this distortion include multiple interface interactions (bottom and surface), 3-D sound velocity variations, array location error, sensor error, and the fact that the problem is frequency dependent. Most important, the operator’s task of interpreting the data and minimizing the target region becomes significantly more complex and time consuming.

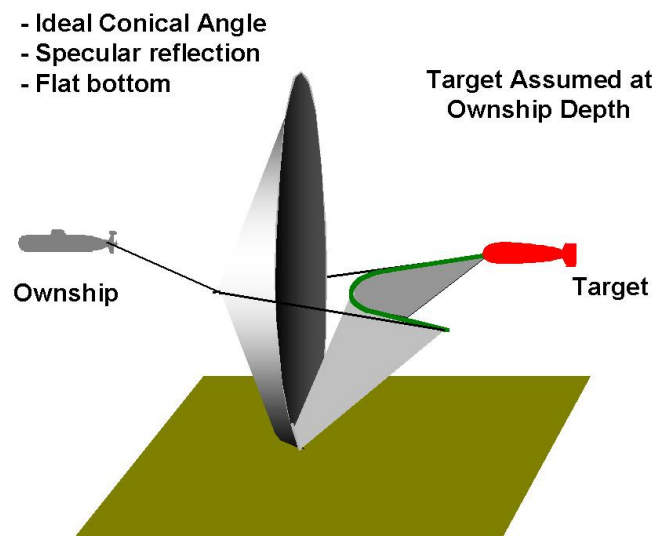


Figure 1. Ideal Conical Angle Measurement

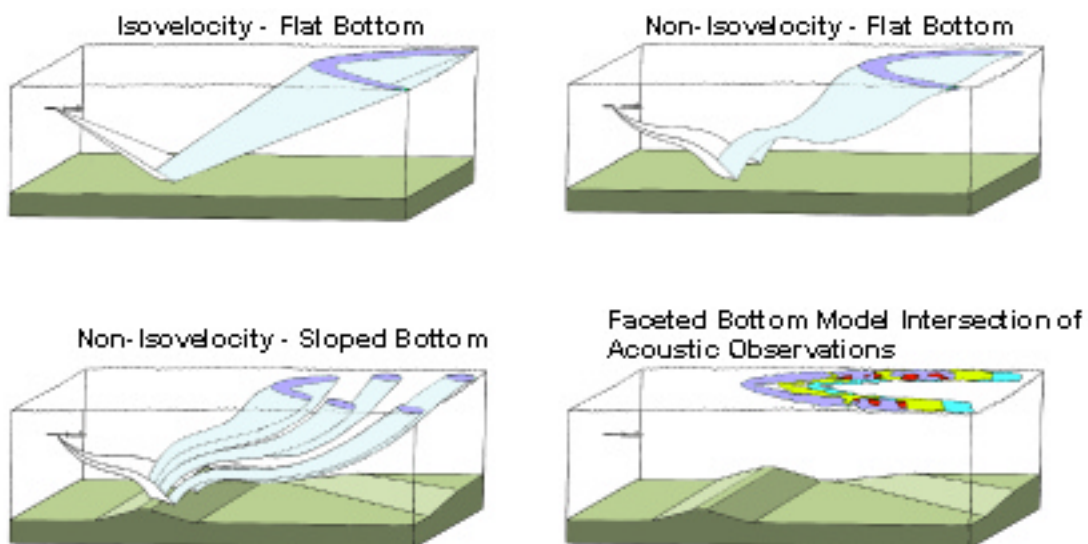


Figure 2. Distortions of Conical Angle Measurement

(Artistic renditions by William Gilroy, NUWC)

This research project has focused on two objectives, the first of which is the development of a real-time 3-D simulation and visualization methodology for a submarine shallow-water passive tracking problem. This objective encompasses the following three specific tasks (organization responsible for the task appears in parentheses after the task): (1) the development of an acoustic scattering model/scenario generator (NUWC), (2) the development of a 3-D rendering/graphical-user interface (GUI) (NUWC/NRL/VT), and (3) the development of an interactive device interface/integration/assessment (NRL).

The second objective for this research project is conducting a 2-D versus 3-D assessment, including the 3-D interactive device, using fleet operators.

1.3 CURRENT STATUS

A prototype command and control (C&C) 3-D visualization, the Submarine Undersea Battlespace Visualization (SUBV), has been created (see figure 3).

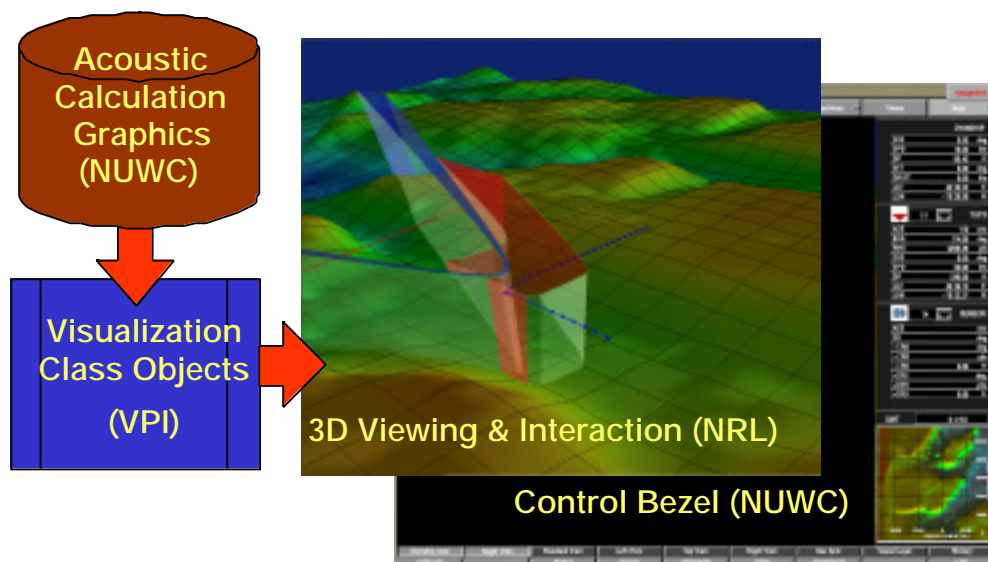


Figure 3. Submarine Undersea Battlespace Visualization

SUBV addresses the problem of visualizing the 3-D nature of measurements received from a submarine towed line array and comprises three main areas of visualization in the context of a submarine antisubmarine warfare (ASW) mission:

1. design and development of algorithms that model the undersea environment and ship kinematics,

2. identification, implementation, and validation of data representation techniques, and
3. development of principles for guiding 3-D development for undersea information and on the value added in the use of 3-D techniques.

Significant work still needs to be done in the area of determining the volumetric shape, size, and location of multiple disparate reduced target containment regions resulting from the intersection of previous containment regions expanded as function of time between towed array updates.

1.4 PRESENTATION OF INFORMATION

Section 2 is a detailed discussion of the problem, methodology, and work to date on the high-resolution acoustic propagation models necessary to determine these complex volumetric shapes; it also provides the results of the comparison testing with the CCS Mk2 baseline system. Sections 3 through 5 describe the methodology and work to date on visualizing the acoustic data. Section 6 summarizes the project accomplishments and future plans.

2. SCENARIO GENERATION, TARGET TRACKING, AND THE ACOUSTIC AND GEOMETRIC 3-D MODELING OF 3-D SENSOR MEASUREMENTS

2.1 TARGET TRACKING

The objective of this project is to design, develop, and assess 3-D visualization as an aid for the ASW decision maker to (1) rapidly assimilate the battlespace (2) understand the problem uncertainty and develop more accurate solutions, and (3) acquire new insights into the battlespace. To conduct this assessment, a tactically relevant problem (shallow-water passive tracking) in a complex acoustic environment that included measurement error and uncertainty was utilized.

To conduct a 2-D/3-D assessment using fleet operators, a scenario generator was developed. In essence, this generator allows an operator to prosecute a contact in real time and permits the operator to make decisions (such as an ownship maneuver) during the target containment process. This scenario generator simulates target locations and the measurements that would be received from a linear towed array (conical angles) during an actual engagement for the particular scenario geometry; it incorporates target and sensor error models and includes a high-resolution acoustic scattering model.

The 3-D acoustic scattering model can incorporate varying degrees of complexity that depend on the available environmental information. High-resolution acoustic and bathymetry data, while preferable, are not required for the model to run. The error associated with the target containment region computation, however, is inversely proportional to the resolution of the data. The model determines the conical angles between ownship and target based upon a particular geometry and bathymetry (Baylog, 1997). The measurement surface is sampled along a discrete

set of rays that are propagated through the undersea environment². The scattering that results from ray-boundary interactions (with the bottom and/or surface) is then computed. The bottom is modeled by a set of triangular facets that join the bathymetry samples. Bathymetry constructed from actual site survey data may also be used. The slope of each facet is determined, allowing 3-D reflections to be computed. This model, when combined with the scenario generator, permits the propagation paths associated with received measurements to be rendered in three dimensions. Variable sound velocity profiles (SVPs) along the individual propagation paths are incorporated to compute the warped surface (i.e., the measurement surface that has been changed by the environment). Sensor (towed array) measurement noise is incorporated into the model as an error source. This error manifests itself in the model as a “thickness” along the surface of the conical angle (i.e., it gives thickness to the cone).

Three-dimensional representation of the conical angles reveals the ambiguities that exist. A single measurement does not provide either left/right or up/down information. The entire conical surface defines the set of possible target locations, each consisting of a unique azimuth bearing, range, and depth to the target. Figure 1 depicts an ideal conical angle in an ideal acoustic environment: the sound velocity profile is constant throughout the medium (yielding straight-line acoustic propagation), the sea bottom is perfectly flat, the reflection off the bottom is specular (the angle of incidence equals the angle of reflection), and the sound rays do not penetrate through the bottom.

Target tracking from conical angle measurements is accomplished in several ways in current fleet systems. The process can be manual wherein an operator adjusts target motion parameters until the residual errors between the projected and actual measurements (i.e., the sum of squared errors cost function) are near zero, or the process can be automatic via several mechanisms. One such mechanism is a direct search where an algorithm runs through a grid of parameter variations (usually in cartesian coordinate space) to determine the maximum value of the likelihood function or the minimum value of the sum of squared residuals. The areas on the grid are plotted and colorized based on the value of the likelihood function at the nearest grid vertex. Typically, the axes are range to target in X and range to target in Y.

Another automated tracking mechanism³ entails the use of recursive methods for optimizing the selected cost function and are typified by the extended Kalman filter (Blackman, 1999). Alternately, batch estimation methods, which are somewhat more robust, are based on iterative nonlinear optimization methods that require an initialization of the parameters being estimated. A poor initialization can cause divergence or large errors in the estimator outputs. Further, it is important that the motion model utilized by any estimator matches the actual target motion. Under conditions where a significant mismatch exists, such estimators can produce tracking estimates with large errors.

² Michael Medeiros, NUWC, Code 8221 created a 2-D ray trace routine that was converted to 3-D for this task.

³ This section is not meant to be a definitive source for a review of estimation techniques. For further information, see Blackman (1999) and Bar-Shalom (1993).

When these tracking methods are employed, the assumption is often made that the target of interest is at the same depth as the array, or at least at a fixed depth, and that the target maintains a constant velocity. These methods typically require ownship maneuvers to properly estimate target motion parameters. A maneuver often (but not always—because of geometry dependence) makes the situation *observable*; that is, it guarantees a unique solution to the target motion parameters. These tracking techniques are especially reliable when the assumptions match the operational environment and when measurement data are abundant.

The tracking approach developed in this research uses a recursive filtering approach, where information from past measurements is retained in the form of a probable target containment region. After each conical angle measurement is received, the cone (see figure 4) is expanded between measurements to compensate for a selectable, maximum target speed until the next measurement is received. The expanded prior measurement and new measurement are combined in 3-D space to develop a volume common to both. The resulting volume is called the target containment region. No motion model is used; this process makes no assumptions regarding the actual speed, depth, or course of the target. The process of computing the intersection of the expanded containment region and the next measurement is repeated until target containment minimization is accomplished. Of key significance is that sparseness of data does not limit this process: because the data are not averaged, they need not be received on a regular basis. Instead, past measurement regions continue to be expanded until the arrival of the next measurement.

Artistic Rendition of 3D Containment Region Intersections

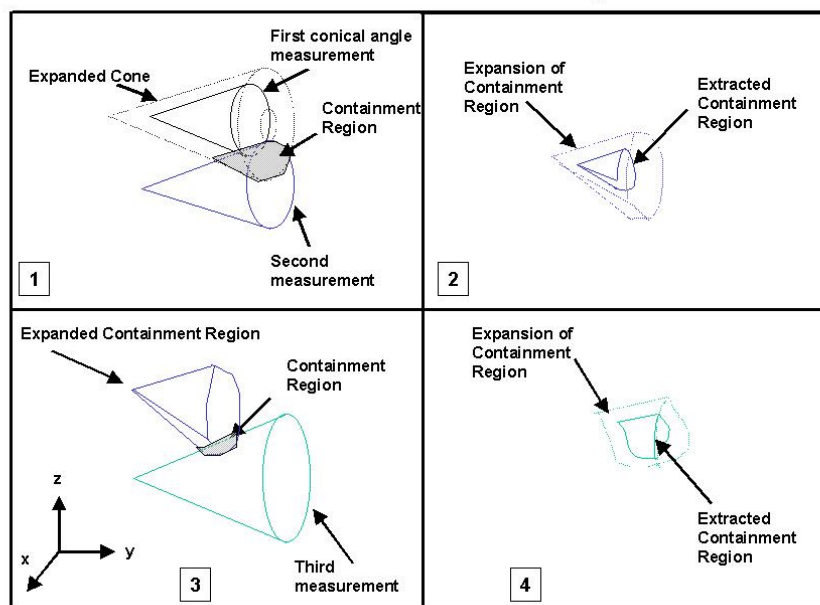


Figure 4. 3-D Containment Region
(Artistic rendition by Megan Gibson, NUWC)

An initial 2-D versus 3-D comparative test was performed (see figure 5). In this example, a geometry was developed that placed the target in the lower half of the cone (in the lower direct path rays) and in the (single) bottom bounce section of the cone. Actual bathymetry from an area north of Cape Cod, MA, was utilized, and the conical angle measurements were simulated.

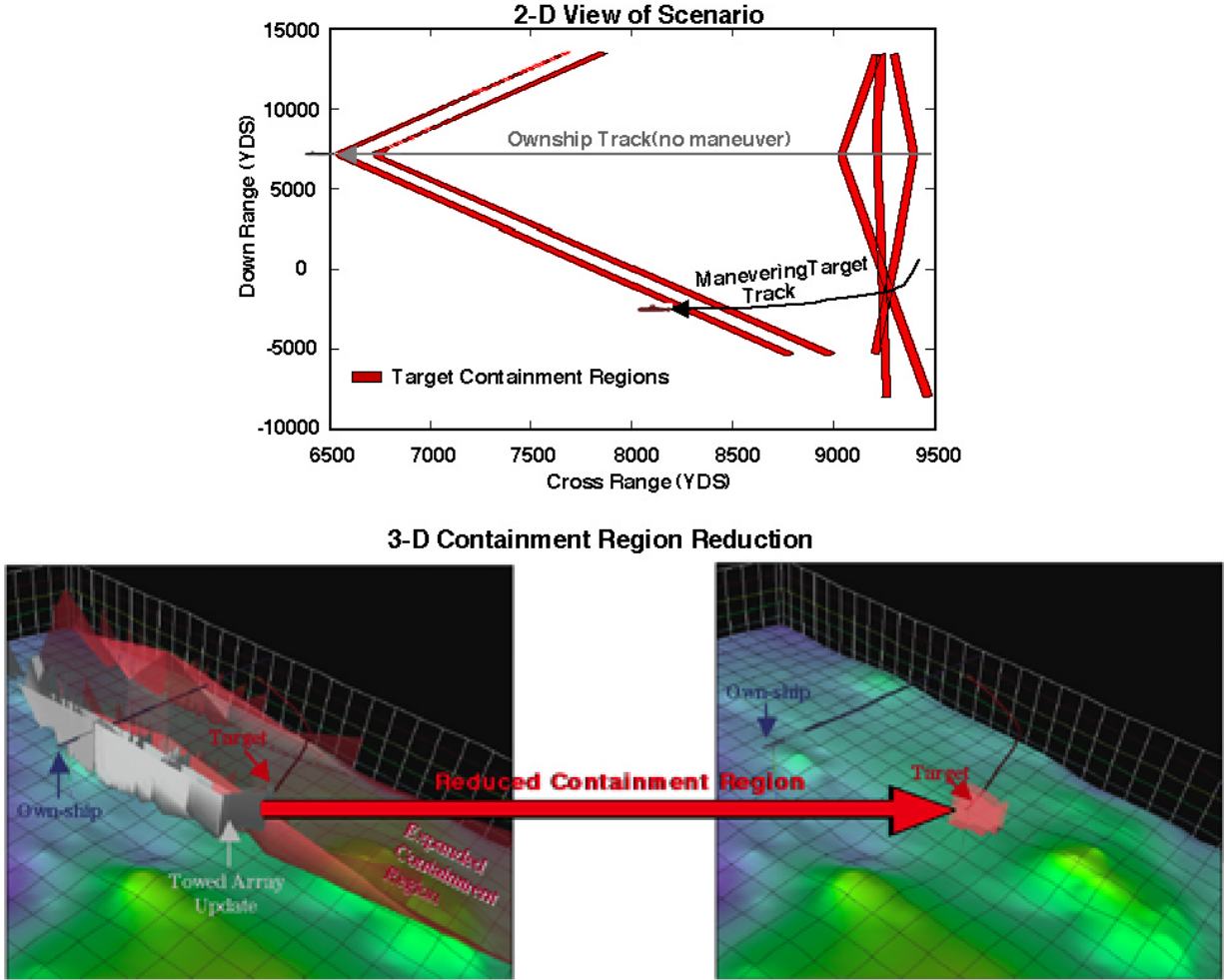


Figure 5. 2-D Versus 3-D Comparative Test – Continuously Maneuvering Target

The results of this test were particularly interesting for several reasons. In the selected scenario, the target was constantly maneuvering, which caused the current TMA system to have great difficulty in tracking the target because of the constant velocity assumption implicit in the TMA tracking models. The NUWC-developed 3-D tracking model, however, was able to provide target location information throughout the scenario.

It was also apparent that in contrast to current fleet systems, the 3-D tracking system could incorporate bottom bathymetry (utilizing its roughness and variability) into the target containment computation. While the current systems were unable to develop any reasonable target location information, the 3-D system was able to bound the target within an acceptable region for weapon deployment after only five measurements *without an ownship maneuver*.

2.2 EXTRACTION TECHNIQUES

The computation of the target containment regions is a difficult problem. Several methods have been employed to date. These approaches discretize the measurement surface into a set of points and use a distance measure as the parameter by which regions of intersection were found and extracted. Several semi-manual processes were evaluated and proved computationally fast, but because these processes are labor intensive, they are not feasible for fleet deployment. Determining the boundaries of objects and the intersection of these boundaries is an issue that game developers deal with constantly. A number of publications address this issue (Melax, 1999), and a few techniques are under review (Preparata et al., 1985). The approach currently being pursued breaks the cone into a set of convex hulls and searches for intersections between the hulls from successive measurements. There are trade-offs between computational speed and the number of samples used to represent the measurement surface.

2.3 ERROR ANALYSIS

Several references in the preceding paragraphs have addressed error approximation. The process of determining how much error is incurred in linearizing the conical angle measurement surface (in transforming of the cone into discrete sections) is currently underway. Another source of error results from having a less than precise environmental characterization. If, for example, actual bathymetry exhibits a high degree of variability that is not accurately incorporated into the acoustic data set, significant error could result. Bottom composition and texture is another source of error that will be studied.

3. 3-D VISUALIZATION BACKGROUND AND OVERVIEW

The overall goal to evaluate 3-D rendering of complex tactical information in a Navy-relevant domain defines these technical issues for the visualization effort:

1. incorporation of all desired battlespace information into multiple layers of a single picture allowing for comprehensive understanding of the common tactical picture (CTP),
2. methods for visually representing acoustic information error,
3. reduction of the excessive cognitive demand placed upon the user using visualization and interaction methods, and
4. exploitation of a multi-modal interface.

Because NUWC as technical lead needed to explore the general issues associated with 3-D visualization of undersea environments, an existing 3-D visualization, called CONRAY and developed by Shell, was converted to display towed array data in addition to its original task as a 3-D weapon visualization. An early form of CONRAY, called 3DVIEWER, (Shell, 1997) was developed to visualize acoustic returns from forward-looking sonars on unmanned undersea

vehicles (UUVs). The software was modified to include threat containment regions generated from submarine towed array data.

CONRAY does not have an object-oriented (O-O) architecture; it is written in the C programming language and uses the *Performer*^{TM 4} Applications Programmers Interface (API) to render its 3-D graphics, called a scenegraph. The goal to convert CONRAY to the O-O architecture necessary for transition to the fleet-compatible TAC⁵ workstations was successfully accomplished (see sections 4 and 5 for details). CONRAY continues to be used as a prototyping tool, but that role will diminish as the new software, called SUBV, matures.

SUBV is software derived from the virtual reality battlefield visualization system, termed Dragon, developed by NRL (see Durbin, 1998 and Julier, 1999) for the virtual reality responsive workbench. The most current version, *Dragon II*, is a core component of the Interoperable Virtual Reality System (IVRS)⁶.

During the second year of the project, VT joined with NUWC and NRL to create a software package called SUBV. SUBV is built upon the visualization paradigms found in CONRAY, but it is a totally new O-O based software product. The SUBV software combines the following components:

1. NUWC-developed information panel,
2. NUWC-developed acoustical displays and bottom bathymetry,
3. NRL-developed 3-D interactive package (IVRS),
4. NRL-developed interprocess communication package (SOCKETLIGHT),
5. VT-developed bottom rendering and display grid, and
6. VT-furnished scene builder/controller.

The SUBV software package achieves the goal of creating an O-O visualization that integrates with a 2-D control panel to approximate the “look and feel” of the current CCS Mk2 system (see figure 3).

⁴ Silicon Graphics Performer is a software environment for developing high-performance, real-time graphics applications. It is compatible with all Silicon Graphics (SGI) graphics platforms.

⁵ Tactical Advanced Computer (TAC) Joint Workstation is a General Services Administration (GSA) acquisition award designed to support the Department of Defense. Hewlett-Packard is the authorized vendor for this contract award.

⁶ Interoperable Virtual Reality System (IVRS) is a reconfigurable, dynamically extensible, virtual reality API being developed to operate on a number of computing platforms and with different combinations of display devices and interaction hardware.

4. CONVERSION OF 3-D VISUALIZATION TO O_O ARCHITECTURE

4.1 RATIONALE FOR AN O-O ARCHITECTURE

As a collaborative research project among NUWC, NRL, and VT, SUBV added challenges to the development and stability of the code base. After development of an initial proof of concept, prototype efforts were directed toward design of an O-O architecture that would support the following objectives:

1. parallel development (i.e., for both expertise and security reasons, different parts of the system had to be written in parallel in different geographical locations for cone detection, control bezel, navigation, and rendering,); and
2. iterative design cycle (i.e., a flexible scheme allowing rapid modifications after evaluation and multiple modifications along the development, compelling all sites to proceed in a “lock-step” fashion all the time, that is, waiting for a site to finish its evaluation and redesign before continuing with the project).

To accomplish these objectives, it was necessary to design for minimizing dependencies. An O-O architecture provided a clear delineation of problems, well-defined responsibilities represented in the code, and minimized the impact of changes using design-by-contract (Meyer, 1992) and generic interfaces.

Since the purpose of the project is to assess the improvements in decision-making that might be derived from 3-D visualization, the rendering subsystem served as the focal point for all the other subsystems. In the current stage of development, the Silicon Graphics proprietary product *Performer*TM (OpenGL Optimizer Programmer’s Guide, 2000) is used to implement the scene rendering (called scenegraph), which is divided into three layers: data layer, visualization layer, and virtual environment (VE) display layer. The data layer contains objects that are able to load the data (bathymetry raw points, target information, etc). These objects implement a protocol that abstract the data representation, providing the functionality needed in the visualization layer, such as surface interpolation for the bathymetry data grids. The visualization layer uses the data grid to build a scenegraph containing objects that react to changes in the data layer objects, and visualizes the changes. The display layer performs the rendering steps and projections necessary for a particular display device, be it a CRT monitor or *ImmersaDesk*TM, and a particular VE toolkit, like IVRS or DIVERSE. Separating the display layer from the visualization layer and physically encapsulating all display layer-dependent code in a single, top-level file was a successful early design decision permitting the use of DIVERSE⁷ while IVRS was still in development. Code that was not API-dependent allowed for independent experimentation with different 3-D navigation techniques using a wider variety of virtual environments. As a result, more mature display layer code was later adapted to work with IVRS requiring only minimal coding changes.

⁷ Device Independent Virtual Environments-Reconfigurable, Scalable, Extensible (DIVERSE) is an API designed to facilitate the development of immersive computer graphics programs which can transparently use a variety of graphics displays and input devices.

4.2 SYSTEM ARCHITECTURE

The rendering subsystem is written in C++ and consists of approximately 5000 lines of code and approximately 46 classes of generic 3-D object prototypes (excluding prototype classes within the *Performer*TM API). Figure 6 is a class diagram that shows part of the structure of the rendering subsystem.

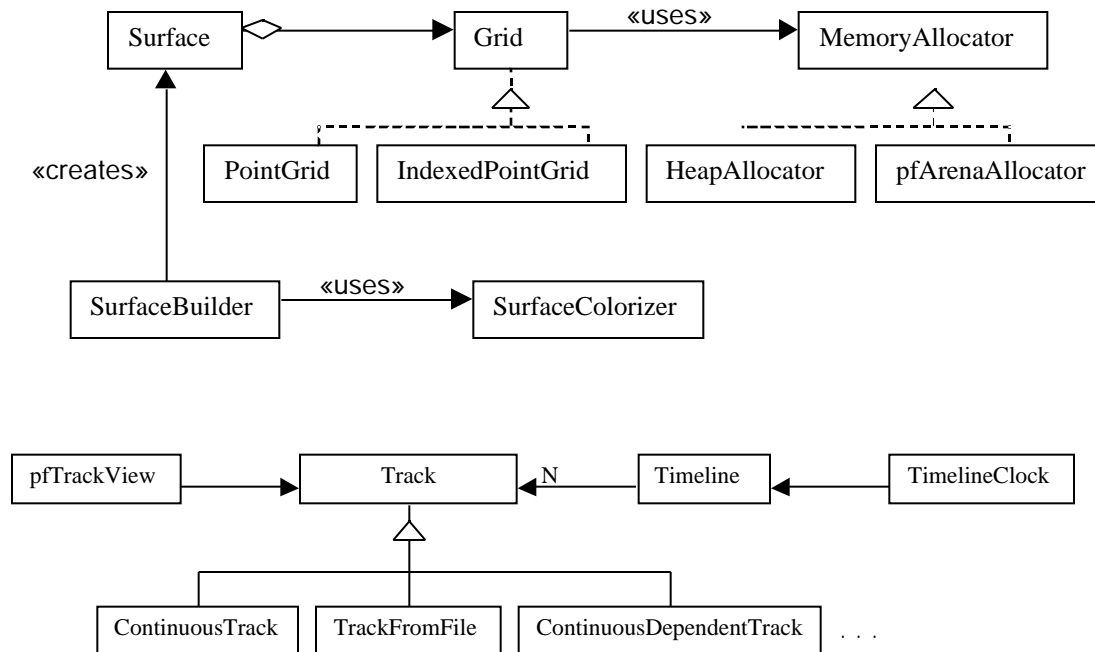


Figure 6. Structure of the Rendering Subsystem

The architecture of the rendering subsystem is centered on four concepts: tracks, markers, surfaces, and observed/observed relationships. Tracks conceptually are a sequence of positions in space and a current position, each one with a timestamp. Tracks represent any moving object in the visualization, like a submarine or a torpedo. A track may even be a “view” of another track and depend on other track’s state. In the prototype for example, an instance of the class TrackFromFile was used to get from disk a sequence of discrete points that mark some of the submarine’s route position. An instance of ContinuousDependentTrack, which depended on an instance of TrackFromFile, was used to represent the submarine’s position as a continuous path by interpolating between the discrete points.

A marker is a representation of a position in a track. Markers and tracks are related by an observer/observed relationship in the spirit of the model/view (Krasner, 1988), a particular case of the observer pattern (Gamma, 1995). Some of the tracks by multiple inheritance are also an observed subclass (the models), and markers (like class pfTrackView) are subclasses of observer (the views). Every time the model changes, it broadcasts a notification of its state change by calling `observed::notify()`, which in turn calls `observer::update()` for all observers who have added themselves to the observed list of interest. Tracks are passive objects; that is, they don’t have any concept of current time or when to advance or go back in time. Tracks are coordinated by an instance of timeline, which depends on a time beat and the current time given by

TimelineClock. For every execution of the main loop, TimelineClock gives its current time to the Timeline, and the Timeline decides which tracks to notify according to the `timeOfNextEvent()` of the track. TimelineClock works as a view of the current system clock, but can also accelerate time or go back in time, at a speed always proportional to the system clock.

Non-moving objects representing the bathymetry, map grids, and sound layer are all cases of surfaces. A surface is usually created in three steps. In the first step, the data needed to create a representation of the surface are gathered and stored in a grid object. Different types of grids store and synthesize information for different types of regular or irregular meshes, with different types of information per point. Then a builder creates a surface object from a grid. Unlike a grid, a surface can be rendered, and may add information like surface normals to the grid. The allocation of memory is mediated by a single subclass of MemoryAllocator, which provides a uniform interface to request and free memory regardless of where the memory is obtained, like shared memory or local heap. This method is used to parameterize the creation of object and to create temporary instances that live in the heap. Similarly, other instances of 3-D objects that will be part of the *Performer*[™] scenegraph must reside in shared memory.

Contrary to current design trends in C++, SUBV uses pure O-O design, with no use of templates or generic programming (Musser, 1989). The code was crafted so that there are no virtual methods in the critical path, allowing the compiler to optimize the method dispatching at compile time. Templates were avoided because their weak type-checking characteristic make them difficult to debug, which would have made it very difficult for other project participants to diagnose an error. This inability to diagnose an error would have hampered the project team's efforts for parallel development among the different geographical locations. Templates also increase compile time and have a spread effect because classes that instantiates templates must know the exact type of the class parameters or be a template itself. While generic programming is generally considered a powerful technique in the multi-paradigm C++ arsenal, it was felt that a pure O-O design was better suited for the constraints of this particular project.

5. THREE-DIMENSIONAL INTERACTIONS

5.1 BACKGROUND

The Virtual Reality Responsive Workbench (VRRWB) (Durbin, 1998 and Julier, 1999) is a 3-D graphics system that allows multiple participants to interact in a shared virtual environment and physical space. A graphical representation of the battlespace, including the terrain, bathymetry, and military assets that lie within it, is displayed on a projection table. Through careful use of models, user-interaction metaphors, and information displays, the system presents a detailed view of the battlespace without overloading the user with huge quantities of information. Much of the success of a visualization system depends on the ease with which a user can navigate through the virtual environment and interact with objects.

The key questions asked when designing the 3-D interactive components were:

1. What capabilities should be incorporated into the user interface initially to establish a baseline implementation for evaluation?
2. What features should eventually be included to ensure a robust tactical tool?

In a formal development effort, a user task analysis would generally be performed following methodologies such as described in Hix (1999) and Cohen (1997). Such an analysis was not possible for this effort because of the limited scope of the project and the relative newness of interfaces for virtual environments. While metaphors and techniques for 2-D human/computer interfaces are mature, research in 3-D interfaces has just begun. As pointed out by Hinckley (1994), few of the design issues in spatial input have been subjected to formal user studies. Indeed, it is in research projects such as this that 3-D interfaces are being pioneered.

Although research in 3-D interfaces is admittedly limited, there is nonetheless a body of work that addresses 3-D interfaces. NRL, which has several years experience in designing user interfaces for immersive virtual environments, is conducting formal studies to evaluate navigational techniques, the initial results of which have been incorporated into this project.

For this project, a baseline usage scenario was defined and examined for opportunities to apply 3-D interaction techniques. For example, consider the following scenario:

The users stand in front of an immersive virtual reality display (IVRD) (see figure 7). They see a 3-D representation of the acoustic environment (ocean bottom, ocean surface, and a graphical representation of the mixed layer) and one or more submarines in it. The operator is holding a device that can be used to interact with the display system. As the tactical problem unfolds, the users are presented representations of acoustic detections (i.e., conical angles).

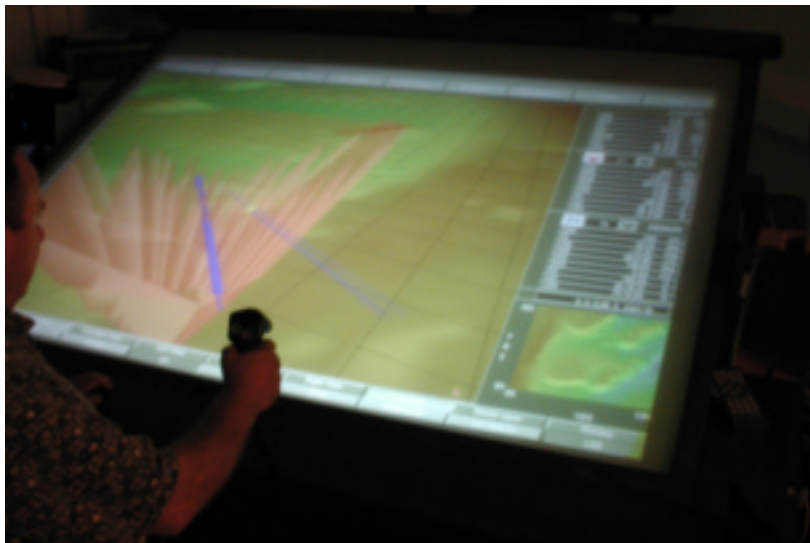


Figure 7. Immersive Virtual Reality Display

Next, try to imagine and list the things that the user might want to do. For example, the operator may want to (1) adjust her/his view of the data (often, viewing spatial data from several different angles is the key to understanding it); (2) interact with the data to select, manipulate, or query by pointing or gesturing; or (3) control the application by choosing, loading, and discarding data sets; by changing problem parameters; or by starting, stopping, or otherwise controlling the application.

These possible operator actions are labeled “navigation,” “interaction,” and “control.” Because of the limited scope of this project, the number of techniques that could be applied was restricted. Hix et al. (1999) observed that “navigation—how users manipulate their viewpoint to move from place to place in a virtual world (in this case, the map for battlefield visualization)—profoundly affects all other user tasks.” Accordingly, navigation, because it is so important to get right, was the chosen focus for this research project.

5.2 THREE-DIMENSIONAL DEVICES

The techniques applied depend upon the input device(s) being used. Over the past several years, NRL has investigated several general interaction methods for the VRRWB (Durbin, 1998 and Julier, 1999), including gesture recognition using a pinchglove, speech recognition, and a flightstick. Candidate devices for the SUBV software included mouse or trackball, flightstick, wand or other pointing device, pinchglove or data glove, and (possibly) voice. Three devices were selected as the principal focus of this project: flightstick, mousepointer, and the trackball.

A flightstick (see figure 8) is a free-space interaction device constructed from a PC joystick, magnetic tracking system, and interface box⁸. The tracking system provides continuous position and orientation (quaternion) six degrees of freedom (DOF) input. The flightstick, which has one or



⁸ Available commercial devices include TNG-3B, cereal box and interface box. If three or fewer buttons are implemented, a PC mouse may be re-wired to serve as the button interface.

Figure 8. VR Flightstick

more buttons to provide digital input, has two key advantages: (1) it is easy to use (an operator can stand in one position and select and highlight entities on any part of the visible display) and (2) it is physically robust (it contains few moving parts and does not rely on precise calibration). In the virtual environment a “laser ray” is attached to the flightstick to aid the user in fine control.

The number of buttons used on a flightstick is a compromise between functionality and simplicity—more buttons mean a greater number of options are available to the user without resorting to the keyboard, but more buttons also means a greater cognitive load on the operator. The flightstick constructed for this project has two buttons: *trigger* and *thumb*. Simplicity was deemed most important for this purpose. Cost, ease of construction, and availability of a suitable button interface device were also factors in determining the number of buttons.

A mousepointer is a logical device that uses the system mouse to emulate, as nearly as possible, a flightstick’s control logic. For the most part, this process consists of mapping the X,Y coordinates of the mouse onto two of the six DOF inputs at a time. Also, since the current viewpoint and viewport geometry are known, the X- and Y- coordinates from the mouse can be converted into a pointer in the virtual world when one is required. Mousepointer devices are useful in “desktop immersion” environments when the use of magnetic tracking is impractical or impossible; they are also useful in the development and debugging of user interfaces.

Trackballs were selected for this project because they are currently employed in virtually every Navy tactical system. A trackball is functionally similar to a mousepointer, except that accommodations are made for the physical differences between the devices (see sections 5.4 and 5.5).

5.3 THREE-DIMENSIONAL NAVIGATION METAPHORS (FLIGHTSTICK)

Viewpoint is controlled by implementing a navigation metaphor such as flying, walking, or driving. The metaphor may be egocentric (user moves through the world) or exocentric (user moves the world). These navigation approaches fall into three basic groups: navigation by steering, by simple gesture, and by complex gesture. A number of alternative flightstick navigation algorithms (navigation metaphors⁹) from each of these groups were carefully considered, and the most promising ones for implementation and evaluation were chosen. A fourth group of algorithms—those that deal with viewpoint management—was also included because it provides methods for storing and retrieving (jumping to) viewpoints.

In 3-D navigation by steering, the user controls the direction of motion by using a pointing device, and the change in viewpoint occurs at a more or less constant rate. Familiar examples include driving and flying simulators. Typically, the user controls the direction and/or the magnitude of the operator’s motion vector. The control may represent a velocity or acceleration applied to the viewpoint in each time increment.

⁹ Properly, the *algorithm* implements the *metaphor*, but the terms are used interchangeably.

In 3-D navigation by simple gesture, the user changes the viewpoint by making a gesture. There are many possible meanings to the term gesture. In Cohen (1997), the term is used to refer to the sequence of line segments made while the user draws a symbol. In other systems, gesture refers to hand posture or the time history of complex inputs from tracking systems and/or data gloves. In this context, the term gesture is used to mean the motion—change in position and orientation—that the user makes with the 3D interaction device. A gesture has a start (when the user presses the flightstick button), a current value (the difference in current position and orientation with respect to the gesture start), and an end (when the user releases the button).

The user's gesture may be interpreted in terms of its displacement component, orientation component, or both. A displacement gesture consists of the vector between the current flightstick position and its position at the start of the gesture. An orientation gesture consists of the difference between the current flightstick orientation and that at the start of the gesture.

Simple and complex gesture algorithms are differentiated by the relationship between the gesture and user's motion. With simple controls, the motion is more or less intuitive—the view is changed in the direction that the flightstick is moved. Complex controls must be learned, but can be extremely powerful.

The first navigation algorithm implemented in the SUBV software is a simple gesture metaphor called *pan/zoom*. The pan mode simulates the motion of a flying saucer (i.e., the viewpoint moves without regard to the laws of physics). The user presses the trigger, then moves the flightstick in the direction he wants to travel: forward/back, up/down, right/left, or any combination of these. The gesture¹⁰ is transformed from tracking to virtual world coordinates and then multiplied by a scalar to become the user's velocity vector. Thus, the user's motion in the virtual world is proportional to the magnitude and direction of the gesture. A "dead zone" may be programmed to permit hovering—the user stops moving by returning the flightstick to very near where he started the gesture.

Rotate, another simple gesture algorithm, rotates the user's viewpoint using an orientation gesture—the difference in orientation between the flightstick's current orientation and that at the start of the gesture. It may be appropriate to restrict the degrees of freedom controlled by rotation: frequently, ignoring the roll component makes navigation easier.

An example of complex 3-D navigation is the *examine* navigation mode in the SUBV software. As the user points the flightstick, a yellow square appears to mark the center of interest (COI) where the flightstick laser pointer intersects the ocean floor. This intersection is an important visual reference. When the user presses the appropriate button, the gesture starts and the COI becomes fixed. The component of the user's displacement gesture is then calculated in the direction of the vector between the current flightstick position and the COI. This

¹⁰ A gesture may be measured in tracking system, display, or world coordinates, and is transformed from one coordinate system to another using matrix operators. For the discussion herein, viewpoint position consists of a Cartesian coordinate triplet (x,y,z) in an earth-tangent plane. (In other words, we are flat-earth). Note that many coordinate systems are available, and conversions between them are readily on hand. Viewpoint orientation may be expressed either in Euler angles heading, pitch and roll, or (preferably) as a quaternion.

component becomes a velocity control for egocentric motion towards or away from the COI. A second input control is developed from the flightstick orientation gesture. A twist of the flightstick results in an exocentric twist of the virtual world *about the COI*. A further calculation is made that adjusts the user's height with his motion towards or away from the COI. Finally, the direction of view is continuously updated to maintain the COI at the same position in the viewport as when the gesture started.

A summary of flightstick navigation is presented in table 1.

Table 1. Flightstick Navigation

Mode	Button	Action
Pan/zoom	Trigger	The user's motion is in the direction of the gesture, with a velocity directly proportional to the displacement gesture (egocentric).
Examine	Thumb	The user's (exocentric) motion around the COI is determined by the gesture's orientation component. The user's (egocentric) motion towards or away from the COI is determined by the displacement component.
Pitch/yaw	Both	The user's orientation is changed with an orientation gesture (egocentric).

5.4 MOUSEPOINTER CONSIDERATIONS

The mousepointer was made to behave as much like a flightstick as possible to preserve the nature of a 3-D interactive device. There are some inescapable differences, however, arising from the need to map mouse's two-DOF inputs onto six-DOF. For example, there is no laser pointer because the algorithm for calculating a pointer relies on the current viewpoint (the user would be looking down the axis of any laser pointer that could be rendered). Also, if the choice was made, for example, to map the mouse X,Y coordinates to the flightstick's X,Y, then the flightstick Z would be held constant, resulting in motion at constant altitude or depth. To gain greater flexibility, the shift keys are used to modify the variable mappings. Table 2 summarizes mousepointer navigation.

Table 2. Mousepointer Navigation

Mode	Button	Action
Pan/zoom	LEFT	The mouse X,Y coordinates are mapped to the flightstick X,Y (<i>resulting in motion at constant altitude or depth</i>).
	SHIFT + LEFT	The mouse Y-coordinate is mapped to the flightstick Z (<i>resulting in a change in altitude or depth only</i>).
Examine	RIGHT	The mouse X-motion is mapped to the flightstick twist. The mouse Y-motion is mapped to the flightstick Y.
	SHIFT + RIGHT	The mouse Y-motion is mapped to the flightstick Y (<i>only motion directly toward or away from the COI is possible</i>).
	CONTROL + RIGHT	The mouse X-motion is mapped to the flightstick twist (<i>only motion around the COI is possible</i>).
Pitch/Yaw	Middle	The mouse X-motion changes heading; Y-axis motion changes pitch.

5.5 TRACKBALL CONSIDERATIONS

Although a trackball is functionally similar to a mouse, it is physically different. The ball has mass that experienced operators use to their advantage when pointing or moving. It is often awkward to press a button while simultaneously rolling the ball. Arcade games, designed so that a player can use different hands for pressing and rolling, are a good example.

These differences are accommodated in the implementation. The buttons are treated as toggle rather than momentary buttons. Thus, a button is clicked once to activate and clicked a second time to deactivate. Once a mode has been activated, cursor warping is used to keep the cursor within the SUBV window. Finally, the pan mode navigation metaphor was changed to one that seems more appropriate for use with a trackball device. The results of these are summarized in table 3.

Table 3. Track Ball Navigation

Mode	Button	Action
Pan/zoom	LEFT	The trackball X,Y motion coordinates are used to control displacement in the XY-plane (<i>egocentric</i>).
	SHIFT + LEFT	The trackball X-motion is used to control altitude.
Examine	RIGHT	The trackball X-motion is mapped to the flightstick twist. The trackball Y-motion is mapped to the flightstick Y.
Pitch/yaw	MIDDLE	The trackball X-motion changes heading; Y-axis motion changes pitch.

One advantage to using the mousepointer or trackball is that when the mouse cursor is over the bezel, it is still a mouse. In the present implementation, interaction with the bezel is not possible with the flightstick. In FY02, experiments will be conducted to assess operator performance using the flightstick, mousepointer, and trackball.

5.6 VIEWPOINT MANAGEMENT

The user has the option of storing a particular viewpoint for later retrieval. In the SUBV software, the user can “jump” to a previously saved view by pressing one of the keyboard numbers (0-9) and save a viewpoint by pressing SHIFT + the number, both of which can be written to and loaded from an ASCII text file. The user can also jump to one of four preset views (left, right, top, or standard) by clicking on the corresponding button on the bezel.

6. CONCLUSIONS AND FUTURE PLANS

The objective of this project is to design, develop, and assess 3-D visualization in tactically relevant scenarios as a means to determine whether 3-D permits the ASW decision maker to: (1) assimilate the battlespace more rapidly, and/or (2) develop more accurate solutions, compared to the current 2-D system. While this assessment is still underway, preliminary experiments suggest that by using 3-D visualization, complex information can be more readily assimilated, improved target localization accuracy can result, and that target localization may be accomplished more rapidly. While further testing will enable quantification of the improvements in these areas, it is clear that, for the scenarios modeled, the question is not whether 3-D provides an improvement, but rather how much improvement it provides over current systems.

This project has principally focused upon an assessment of 3-D visualization. Results obtained to date, however, combined with the models and rendering capabilities developed in the course of the project, have laid the foundation for a new concept for an integrated sensor-to-shooter decision aid for the undersea battlespace. This new 3-D tactical decision aid (TDA) uses a containment region/maximum error approach to target localization rather than a statistical estimation approach, although the system can overlay such solutions. This TDA will be composed of a number of modules that will enable it to accomplish the following tasks:

1. incorporate detailed acoustic parameters of the undersea battlespace;
2. accurately compute and depict the interaction of sensor information with the environmental profiles;
3. incorporate sensor information from any source;
4. perform and present real-time visual fusion of such information including target speed and measurement latency compensation;
5. develop accurate target containment regions using maximum error bounds;
6. assess weapon performance against target containment regions “on-the-fly”; perform automated weapon presetting and weapon fire recommendations in real time;
7. provide for the display of weapon post launch data/status within the integrated battlespace display; and
8. provide the capability to perform post launch guidance/correction.

In this TDA concept, all computations are performed in three dimensions and NO assumptions about target bearing, course, or speed are made (except for an operator input maximum possible target speed). This TDA is also unique because it provides the decision-maker with the capability to integrate his operations (OP) orders and intuition into the

problem solution in real time, and it performs subsequent computations based upon this input. Finally, rather than complicating decision-making by computationally combining various sources of data, this TDA concept will present all data in 3-D and provide the operator with a visual integration of the data to facilitate its transformation into information.

Since the obvious transition path for this work is the FNC program, the decision was made to address the most difficult challenges/constraints to the TDA concept in the final year of this project. The three major challenges fall into three areas:

1. error assessment/sensitivity analysis and representation for sub-optimal data sets;
2. real-time acoustic computations, multisensor/complex environment in HP workstation environment (60X increase required); and
3. real-time acoustic rendering for full undersea environment on HP workstation.

During FY02, work in these three areas, conversions of the existing base code to multicomputer platform OpenGL, and continuation of the 2-D/3-D experiments will comprise the scope of the overall effort. Modest success in achieving each of these objectives will mitigate risk and thereby facilitate transition of this exploratory development effort into a major FY03 FNC initiative.

BIBLIOGRAPHY

- Bar-Shalom Y., and X. R. Li, *Estimation and Tracking: Principles, Techniques and Software*, Norwood, MA: Artech House, 1993.
- Baylog, J. G., "Environmental Data Integration in Littoral Waters," *Proceedings of Fourth Specialists' Meeting*, TTCP MARGRU-TP-1, Naval Undersea Warfare Center Division, Newport, RI, 14-18 Apr 1997.
- Blackman S., and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House Publishers, 1999, pp. 164-168.
- Cohen, R., M. Johnston, D. McGee, S. Oviatt, J. Pittman, I. Smith, L. Chen, J. Clow, "QuickSet: Multimodal Interaction for Distributed Applications," *Proceedings of the Fifth ACM International Conference on Multimedia*, 1997, pp. 31-40.
- Durbin, J., J. Edward Swan II, B. Colbert, J. Crowe, R. King, T. King, C. Scannell, Z. Wartell, T. Welsh, "Battlefield Visualization on the Responsive Workbench," *Proceedings IEEE Visualization 98*, October 18-23, Research Triangle Park, North Carolina: IEEE Computer Society Press, 1998, pp. 463-466.
- Gamma, E., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

- Hinckley, K., R. Pausch, J. C. Goble, N. Kassell, "A Survey of Design Issues in Spatial Input," *Proceedings of the ACM Symposium on User-Interface Software and Technology*, 1994, pp. 213-222.
- Hix, D., J. E. Swan II, J. L. Gabbard, M. McGee, J. Durbin, T. King, "User-Centered Design and Evaluation of a Real-Time Battlefield Visualization Virtual Environment," *Proceedings IEEE Virtual Reality '99*, IEEE Computer Society Press, 1999, pp. 96-103.
- Julier, S., R. King, B. Colbert, J. Durbin, and L. Rosenblum, "The Software Architecture of a Real-Time Battlefield Visualization Virtual Environment," *IEEE VR '99*, March 1999, pp. 29-36.
- Krasner, G., and S. Pope, "A Cookbook for Using the Model View Controller User Interface Paradigm in Smalltalk-80," *Journal of Object-Orientated Programming*, August/September 1988, 1(3), pp. 26-49.
- Melax, S., "Convex Hull Simplification with Containment by Successive Plane Removal," Department of Computing Science, University of Alberta, 1999, <http://www.cs.ualberta.ca/~melax/polychop/>
- Meyer, B., "Applying Design by Contract," *IEEE Computer*, pp. 40-51, October 1992.
- Musser, D. R., and A. A. Stepanov, "Generic Programming," *ISSAC '88 Symbolic and Algebraic Computation Proceedings, Lecture Notes in Computer Science*, Springer-Verlag, P. Gianni, (ed.), vol. 358, 1989.
- "OpenGL Optimizer Programmer's Guide: An Open API for Large-Model Visualization," SGI Technical Publication 007-2852-002, January 2000.
- Preparata F. P., et al, *Computational Geometry : An Introduction (Texts and Monographs in Computer Science)*, Springer-Verlag, 1985.
- Shell, R., "3DVIEWER: A Three-Dimensional Visualization and Analysis Tool," NUWC-NPT Technical Memorandum 980039, Naval Undersea Warfare Center Division, Newport, RI, 20 April 1997 (UNCLASSIFIED).