NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

REACTIVE OBSTACLE AVOIDANCE FOR THE REMUS AUTONOMOUS UNDERWATER VEHICLE UTILIZING A FORWARD LOOKING SONAR

by

Tyler H. Furukawa

June 2006

Thesis Advisor:	Anthony J. Healey
Second Reader:	Douglas Horner

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Reactive Obstacle Avoidance for the REMUS Autonomous Underwater Vehicle Utilizing a Forward Looking Sonar			**5. FUNDING NUMBERS**
6. AUTHOR(S) Tyler H. Furukawa			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

One day fully autonomous AUV's will no longer require human interactions to complete its missions. To make this a reality, the AUV must be able to safely navigate in unfamiliar environments with unknown obstacles. This thesis builds on previous work conducted at NPS's Center for AUV Research to improve the autonomy of the REMUS class of AUVs with an implemented FLS. The first part of this thesis deals with accurate path following with the use of look-ahead pitch calculations. With the use of a SIMULINK model, constraints surrounding obstacle avoidance path planning are then explored, focusing on optimal sensor orientation issues. Two path planning methods are developed to address the issues of a limited sonar field of view and uncertainties brought on by an occlusion area. The first approach utilizes a pop-up maneuver to increase the field of view and minimize the occlusion area, while the second approach creates a path with the addition of a spline. Comparing the two methods, it was concluded that spline addition planner provided a robust optimal obstacle avoidance path and along with the look-ahead pitch controller completes the design of a "back-seat driver" to improve REMUS's survivability in an unknown environment.

14. SUBJECT TERMS REMUS, AUV, UUV, Autonomous Underwater Vehicle, Reactive Obstacle Avoidance, Forward Looking Sonar, Vertical Plane, Pitch Controller, Spline, Gaussian, Occlusion, Optimal Sensor Orientation			**15. NUMBER OF PAGES** 79
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

REACTIVE OBSTACLE AVOIDANCE FOR THE REMUS AUTONOMOUS UNDERWATER VEHICLE UTILIZING A FORWARD LOOKING SONAR

Tyler H. Furukawa
Lieutenant, United States Navy
B.S., University of Washington, 2001

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2006

Author: Tyler H. Furukawa

Approved by: Anthony J. Healey
Thesis Advisor

Douglas P. Horner
Second Reader

Anthony J. Healey
Chairman, Department of Mechanical & Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

One day fully autonomous AUV's will no longer require human interactions to complete its missions. To make this a reality, the AUV must be able to safely navigate in unfamiliar environments with unknown obstacles. This thesis builds on previous work conducted at NPS's Center for AUV Research to improve the autonomy of the REMUS class of AUVs with an implemented FLS. The first part of this thesis deals with accurate path following with the use of look-ahead pitch calculations. With the use of a SIMULINK model, constraints surrounding obstacle avoidance path planning are then explored, focusing on optimal sensor orientation issues. Two path planning methods are developed to address the issues of a limited sonar field of view and uncertainties brought on by an occlusion area. The first approach utilizes a pop-up maneuver to increase the field of view and minimize the occlusion area, while the second approach creates a path with the addition of a spline. Comparing the two methods, it was concluded that spline addition planner provided a robust optimal obstacle avoidance path and along with the look-ahead pitch controller completes the design of a "back-seat driver" to improve REMUS's survivability in an unknown environment.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Professor Anthony Healey and Doug Horner for their guidance, assistance and expertise in a field of unlimited potential. Their enthusiasm on the subject provided an enlightening learning environment and proved they "have the coolest toys in the department".

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Autonomous Underwater Vehicles (AUVs) provide the Navy with unlimited potential. According to the Office of Naval Research's (ONR) Future Naval Capabilities program, AUVs will someday provide Maritime Reconnaissance, Submarine Track and Trail, Communication/Navigation Aid, and Undersea Search and Survey without direct human control [1]. A preview of this technology took place in April 2003 with the deployment of the Remote Environmental Monitoring Units (REMUS), a class of AUVs, in the Iraqi Port of Umm Qasr [2]. This first-ever intelligence gathering mission in hostile waters, aided in allowing 232 tons of critically needed food, water, blankets and other supplies to reach Iraqi civilians. According to Ken Jordan, the president of Hydroid (www.hydroidinc.com June 2006), the Navy's main benefit is the possibility for valuable resources such as human divers or multi-million dollar equipment to be replaced with a $250,000 vehicle which is undeterred by cold temperatures, murky waters, sharks or hunger. In order to make this a reality, the vehicle must be fully autonomous and require no human interaction to accomplish its mission in the presence of known and unknown obstacles.

B. PLATFORM

1. REMUS

The REMUS class of AUVs is the product of over 10 years of leading edge research and development by Woods Hole Oceanographic Institute and later the spin-off company, Hydroid LLC.

Figure 1. REMUS 100 (From: www.hydroidinc.com June 2006)

It is a compact, light-weight, autonomous underwater vehicle designed for operation in coastal environments up to 100 meters in depth. Only 67 inches long, 7.5 inches in diameter and weighing less than 80 lbs, the REMUS can be easily transported worldwide and deployed by a two-man team. Its list of applications include hydrographic surveying, harbor security operations, environmental monitoring, search and salvage operations, and fishery operations. For the United State's Navy, REMUS is the instrument of choice for shallow water mine counter measure operations due to its system features, ease of operations, and proven reliability.

Maximum Diameter:	19 cm (7.5 in)
Maximum Length:	160 cm (63 in)
Weight In Air:	37 kg (< 80 lbs)
Trim Weight:	1 kg
Max Depth	100 m (328 ft)
Energy:	1 kw-hr internally rechargeable Lithium ion
Endurance:	22 hrs at 1.5 m/s (3 knots)
	>8 hrs at 2.6 m/s (5 knots)
Propulsion:	DC brushless motor to open 3-bladed prop
Control:	2 coupled yaw and pitch fins
Navigation:	Long baseline (LBL)
	Ultra short baseline (USBL)
	Doppler-assisted dead reckoning
Standard Sensors:	Acoustic Doppler Current Profiler (ADCP)
	Doppler Velocity Log (DVL)
	Side Scan Sonar
	Conductivity & Temperature
	Pressure

Table 1. REMUS Specifications (After: www.hydroidinc.com June 2006)

2. Forward Looking Sonar (FLS)

In November 2005, the Center for AUV Research's REMUS at the Naval Post Graduate School's (NPS) was fitted with a ProViewer 450-15 multi-beam sonar manufactured by Blue View Technologies (www.blueviewtech.com June 2006).

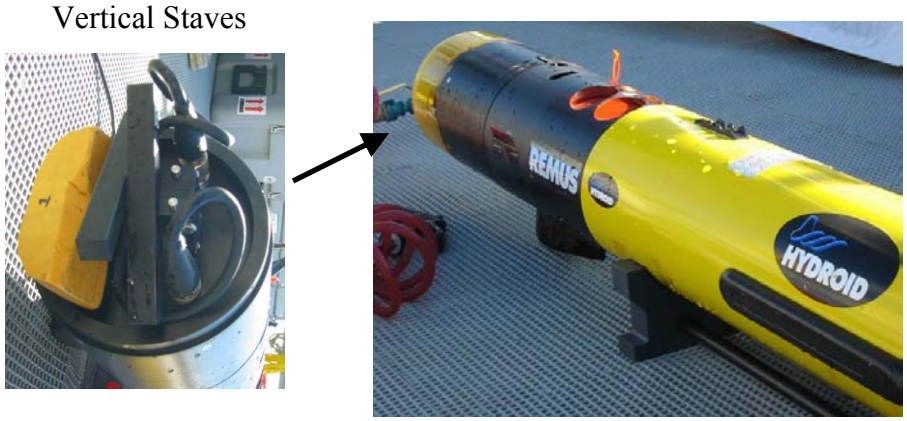

Figure 2. REMUS with BlueView's ProViewer 450-15 Acoustic Sonar

The ProViewer is a high performance acoustic imaging device that bounces sound waves off the ocean floor and objects within its insonified volume and turns them into two-dimensional digital images. The unit can be either mounted horizontally or vertically and the real-time streaming sonar images are rapidly updated for use in applications such as underwater inspection, search and recovery, port security, and fish tracking.

Range:	5 to 450ft
Update Rate:	Up to 10Hz
Transducter:	450 KHz
Field of View:	50°
Beam Width:	1° x 15° nominal
Range Resolution:	2 in
Depth Rating:	Up to 300ft deep
Power:	10 Watts @ 9-36 VDC
Comms Interface:	USB 1.1
Software:	Runs on Windows (2000/XP)

Table 2. ProViewer 450-15 Specification (From: www.blueviewtech.com June 2006)

The ProViewer is mounted vertically and positioned in a "forward looking" manner on REMUS. The forward looking sonar (FLS) easily detects and tracks targets in dynamic conditions and is the vehicle's primary sensor for obstacle avoidance.

C. MOTIVATION

For current REMUS missions, a predetermined path is entered beforehand into the vehicle by means of waypoints. An altitude control "auto-pilot" steers the vehicle to the

path set forth by these waypoints as it gathers information about the environment. Although this approach may seem autonomous, it is limited to known static environments. It is similar to telling a car where and when to turn based on a satellite snap shot of the terrain, without taking into account traffic, pedestrians or changing light signals that may occur during the trip. The introduction of unknown variables into the environment severely degraded the likelihood of a successful pre-planned mission. Since most REMUS operations will take place in the ever-changing littoral waters, it is imperative that the vehicle be able to react to an unknown environment

D. PREVIOUS RESEARCH

This thesis expands on previous research work that has been conducted at the NPS Center for AUV Research dealing with the autonomy issues of REMUS. Healey has shown that the REMUS's normal altitude control "auto-pilot," using only the RDI Doppler Velocity Log, is unable to maintain a safe altitude over sharp rises in the ocean floor of 45 degrees or greater [5]. In anticipation to its installment on REMUS. Churan [6] and Hemminger [7] both looked into using a FLS for obstacle avoidance in the vertical plane. Churan studied the use of a "danger bearing" algorithm, while Hemminger looked into the use of a Gaussian-based additive function for obstacle avoidance. Although both provide a reasonable solution to sharp rises in the ocean floor or objects protruding off the bottom, little or no emphasis is placed on sensor orientation for optimal obstacle path planning.

E. APPROACH

This thesis explores the issues surrounding the development of obstacle avoidance for the REMUS class of AUVs. The goal is to design a "back-seat driver" to work in conjunction with the current onboard altitude control "auto-pilot" to safely navigate the vehicle in the presence of previously unknown obstacle and threats.

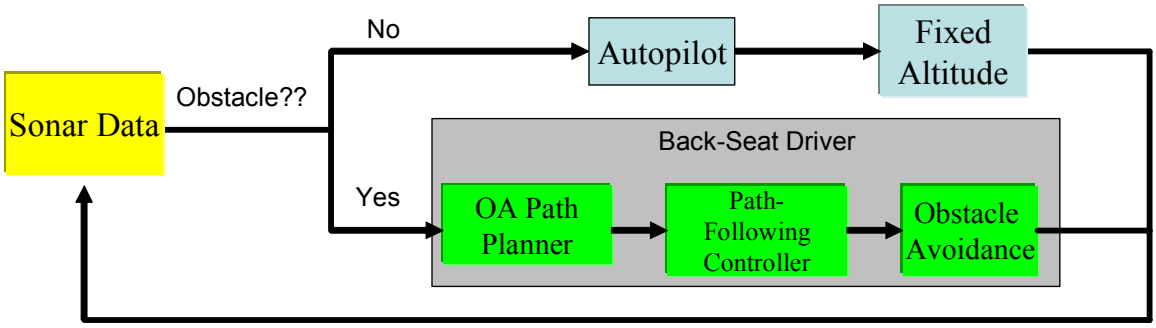

Figure 3. REMUS's Control Architecture

The back-seat driver consisting of a path generator and path-following controller would over-ride the normal configured autopilot when an obstacle is detected. Ideally, the path generator takes the information gathered by its FLS, provides an optimal path to avoid the detected obstacle, and then the vertical pitch controller follows the path with as little error as possible.

The first part of the thesis is the design of an accurate path-following controller by using the equations of motions (EOM) previously derived by Healey [8] to model REMUS's pitch response to stern plane deflections. Next, using a FLS model and simulated hazardous two-dimensional ocean environments, optimal vehicle sensor orientation with regards to obstacle avoidance planning is defined; and issues regarding obstacle height determination and occlusion areas are shown. An occlusion area is a portion of the normal insonified region that goes "unseen" by the sonar and is created when an obstacle blocks the sonar's acoustic waves from "seeing" behind it.

Two approaches are presented to maximize the information gathered about the environment by optimally positioning the vehicle to increase the FLS's field of view. The first approach utilizes an initial pop-up maneuver to determine the obstacle's height and minimize the occlusion area, which enables the planner to generate a safe avoidance path. The second approach alters a standard fixed obstacle avoidance path with the addition of a piece-wise continuous curve for safe navigation in the presence of previously undetected obstacles. The goal to design an effective "back-seat driver" was achieved by implementing the path planning method which most closely satisfies the defined criteria for optimal reactive avoidance along with the ideal path following pitch controller.

THIS PAGE INTENTIONALLY LEFT BLANK

II. VEHICLE KINEMATICS AND DYNAMICS

A. ASSUMPTIONS

In order to achieve realistic simulated results, an accurate model or equations of motions (EOM) must first be derived to describe the maneuvering and motion control of the vehicle. Healey [8] made the following initial assumptions when describing the maneuvering and motion control of REMUS:

- the vehicle behaves as a rigid body;
- the earth's rotation is negligible as far as acceleration components of the vehicle's center of mass is concerned;
- the primary forces that act on the marine vehicle have inertial and gravitational origins and hydrostatic, propulsion, thruster, and hydrodynamic forces from lift and drag.

B. EQUATIONS OF MOTION

Using a Newton-Euler approach and Euler angle transformations, rotational and translational equations were developed, and incorporating the weight/buoyancy forces, Healey [8] derived the EOM for a six degree of freedom model.

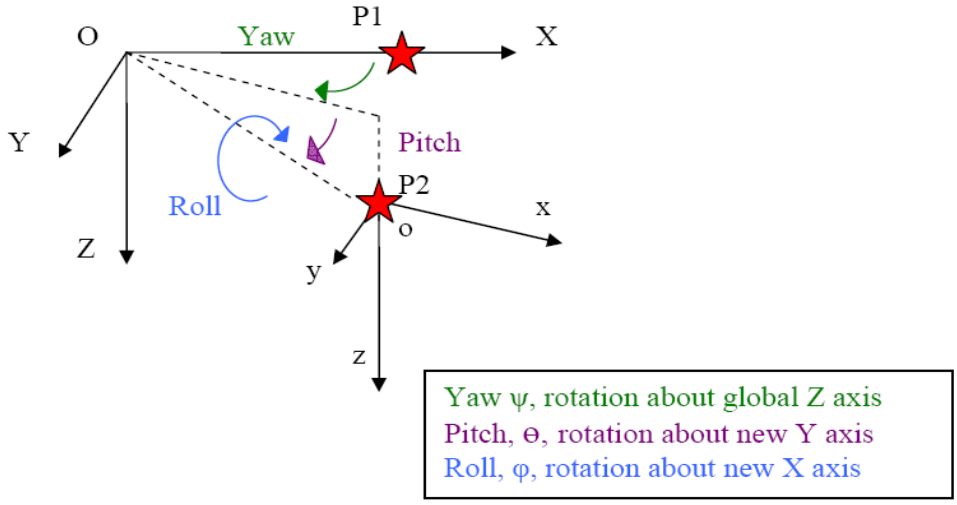

Figure 4. Coordinate System with Euler Angle Transformations (From: [7])

SURGE EQUATION OF MOTION

$$m[\dot{u}_r - v_r r + w_r q - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr + \dot{q})] + (W - B)\sin\theta = X_f$$

SWAY EQUATION OF MOTION

$$m[\dot{v}_r + u_r r - w_r p + x_G(pq + \dot{r}) - y_G(p^2 + r^2) + z_G(qr - \dot{p})] - (W - B)\cos\theta\sin\phi = Y_f$$

HEAVE EQUATION OF MOTION

$$m[\dot{w}_r - u_r q + v_r p + x_G(pr - \dot{q}) + y_G(qr + \dot{p}) - z_G(p^2 + q^2)] - (W - B)\cos\theta\cos\phi = Z_f$$

ROLL EQUATION OF MOTION

$$I_x\dot{p} + (I_z - I_y)qr + I_{xy}(pr - \dot{q}) - I_{yz}(q^2 - r^2) - I_{xz}(pq + \dot{r}) + m[y_G(\dot{w} - u_r q + v_r p)$$
$$-z_G(\dot{v}_r + u_r r - w_r p)] - (y_G W - y_B B)\cos\theta\cos\phi + (z_G W - z_B B)\cos\theta\sin\phi = K_f$$

PITCH EQUATION OF MOTION

$$I_y\dot{q} + (I_x - I_z)pr - I_{xy}(qr + \dot{p}) + I_{yz}(pq - \dot{r}) + I_{xz}(p^2 - r^2) - m[x_G(\dot{w}_r - u_r q + v_r p)$$
$$-z_G(\dot{u}_r - v_r r + w_r q)] + (x_G W - x_B B)\cos\theta\cos\phi + (z_G W - z_B B)\sin\theta = M_f$$

YAW EQUATION OF MOTION

$$I_z\dot{r} + (I_y - I_x)pq - I_{xy}(p^2 - q^2) - I_{yz}(pr + \dot{q}) + I_{xz}(qr - \dot{p}) + m[x_G(\dot{v}_r + u_r r - w_r p)$$
$$-y_G(\dot{u}_r - v_r r + w_r q)] - (x_G W - x_B B)\cos\theta\sin\phi - (y_G W - y_B B)\sin\theta = N_f$$

where:

W = weight

B = buoyancy

I = mass moment of inertia terms

u_r v_r w_r = component velocities for a body fixed system with respect to the water

p, q, r = component angular velocities for a body fixed system

x_B y_B z_B = position difference between geometric center and center of buoyancy

x_G y_G z_G = position difference between geometric center and center of gravity

X_f Y_f Z_f K_f M_f N_f = sums of all external forces (body fixed directions)

C. VERTICAL PLANE SIMPLIFICATIONS

The scope of this thesis only deals with motion in the vertical plane, therefore, the six-degrees of freedom EOM were simplified by neglecting all of the horizontal components, v_r, r, p, ϕ, ψ, and Y. The EOM were further simplified by assuming the following:

- the center of mass of the vehicle lies below the origin
- x_g and y_g are zero
- the vehicle is symmetric in its inertial properties
- u_r equals the forward speed, U_o

The resulting simplified EOM that model REMUS's pitch and depth dynamics in response to stern plane action is:

$$u_r = U_0 \tag{1}$$

$$m\dot{w}_r = mU_0 q + (W - B)\cos\theta + Z_{\dot{w}}\dot{w}_r + Z_{w_r} w_r + Z_{\dot{q}}\dot{q} + Z_q q + Z_\delta \delta_{pl}(t) \tag{2}$$

$$I_{yy}\dot{q} = (z_B B - z_G W)\sin\theta + M_{\dot{q}}\dot{q} + M_q q + M_{\dot{w}}\dot{w}_r + \mathrm{M}_{w_r} w_r + M_\delta \delta_{pl}(t) \tag{3}$$

$$\dot{\theta} = q \tag{4}$$

$$\dot{X} = w_r \sin\theta + U_0 \cos\theta \tag{5}$$

$$\dot{Z} = w_r \cos\theta - U_0 \sin\theta \tag{6}$$

where,

w_r = heave velocity
q = pitch rate
θ = pitch
$\delta_{pl}(t)$ = Stern plane deflection
X = horizontal position
Z = depth

D. MATRIX FORM

The above simplified equations were reduced to the Linear Time Invariant (LTI) form which is used later in the Simulink model:

$$[\mathrm{m}]\dot{\mathbf{x}}(t) = [\mathrm{a}]\mathbf{x}(t) + [\mathrm{b}]\delta(\mathrm{t}) \quad \text{where, } \mathrm{x(t)} = [w_r, q, \theta]^T \text{ and,}$$

$$[\mathrm{m}] = \begin{bmatrix} (m - Z_{\dot{w}}) & -Z_{\dot{q}} & 0 \\ -M_{\dot{w}} & (I_{yy} - M_{\dot{q}}) & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad [\mathrm{a}] = \begin{bmatrix} Z_w & (mU_0 + Z_q) & 0 \\ M_w & M_q & (z_B B - z_G W) \\ 0 & 1 & 0 \end{bmatrix}; \quad [\mathrm{b}] = \begin{bmatrix} Z_\delta \\ M_\delta \\ 0 \end{bmatrix};$$

From the above equations, x(t) is a state matrix and $\dot{\mathbf{x}}(t)$ is the vehicle's response in the vertical plane with respect to time. The input matrices [a] and [b] contain geometry dependent hydrodynamic coefficients of REMUS and [m] is defined as the system's mass matrix.

The forward speed of the vehicle, U_o, was set to a constant 1.5 meters per second since it provided its maximum endurance of 22 hours. The hydrodynamic coefficients used taken from a previous thesis by Prestero in 2001 [9] and shown below.

z_g = 1.96e-2; W = 299;
B = 306; I_{yy} = 3.45;
U_0 = 1.5; m = 299/9.81;
M_q = -6.87; $M_{\dot{q}}$ = -4.88;
M_w = 30.7; $M_{\dot{w}}$ = -1.93;
M_δ = -34.6; Z_q = -9.67;
$Z_{\dot{q}}$ = -1.93; Z_w = -66.6;
$Z_{\dot{w}}$ = -35.5; Z_δ = -50.6;

The Matlab code "Vehicle_Dynamics" [Appendix] took the simplified EOM and the hydrodynamic coefficients and calculated the state space matrices of REMUS for use in the SIMULINK model.

III. PATH FOLLOWING CONTROLLER

A. PITCH CONTROLLER

As previously stated, the requirement for a "back-seat driver" arises due to the inability of REMUS's auto-pilot to safely maneuver over sharp rises in the ocean floor or obstacles proud of the bottom. The most basic necessity for any "driver" or pilot is that it must be able to execute where and what it is ordered to accomplish. Endless research and resources spent on developing a perfect path planning method are wasted if there were no way of following the path; therefore, the first step toward designing an effective "back-seat driver" is developing an accurate path-following controller. Simply stated, it is a modification of the line of sight (LOS) heading controller developed by Healey and Lienard [8], which minimizes the cross track error (CTE) between the vehicle and the adjacent ordered track. Figure 5 defines the variables used in calculating the pitch command for the controller.

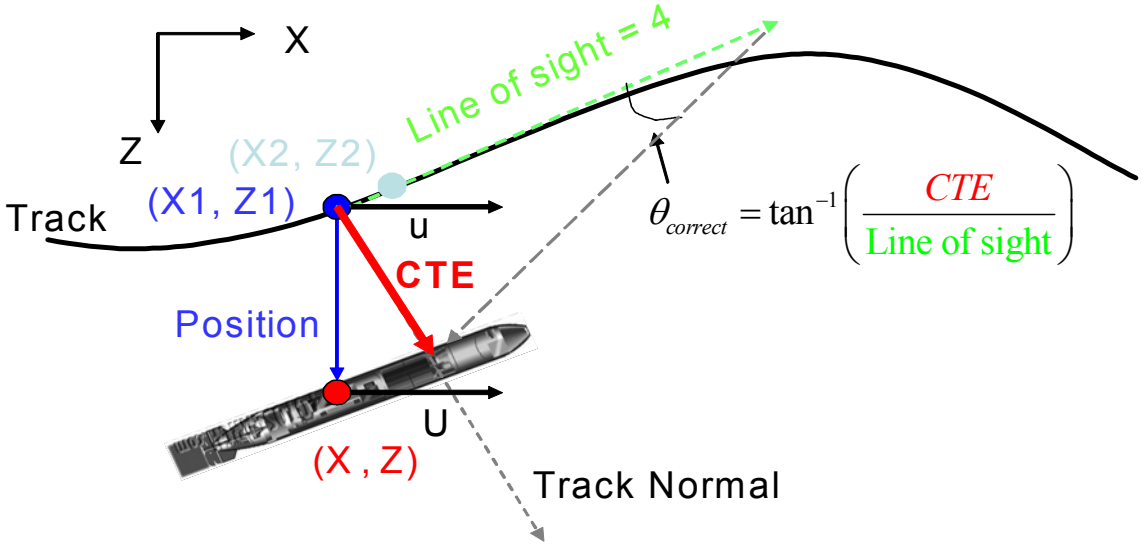

Figure 5. Definitions for Pitch Command Calculation

The coordinates (X, Z) represent the global position of the vehicle. Coordinates (X1, Z1) is where the vehicle should vertically be at its present horizontal "X" position if it were on the ordered track. In other words, "X1" and "u" are always equivalent to the vehicles horizontal position and velocity, "X" and "U," respectively and "Z1" is the

track's altitude at "X". The "Position" vector is a result of the difference between (X, Z) and (X1, Z1). Coordinates (X2, Z2) composes of adding 0.1 to "X1" and calculating the tracks altitude at "X2". The track's tangent is the difference between (X2, Z2) and (X1, Z1). "Track Normal" is the negative inverse of the track's tangent and the "CTE" is the projection of the "Position" vector onto the "Track Normal" vector. "Line of sight" is a constant distance set in front of the vehicle where it is aiming to regain the ordered track. A real-world example to the "Line of sight" approach is steering to the "previewed" road seen out of the car's windshield instead of the "passing" road seen out the driver's side window. The angle or pitch command is then calculated by taking the inverse tangent of the "Error" divided by "Line of sight". A Matlab code entitled "Tracking" [Appendix] was created with the above definitions and used in the next section to provide vehicle pitch commands during simulations.

B. SIMULINK MODEL

A SIMULINK model was created to observe the vehicle's simulated response and the designed controller's path following ability.

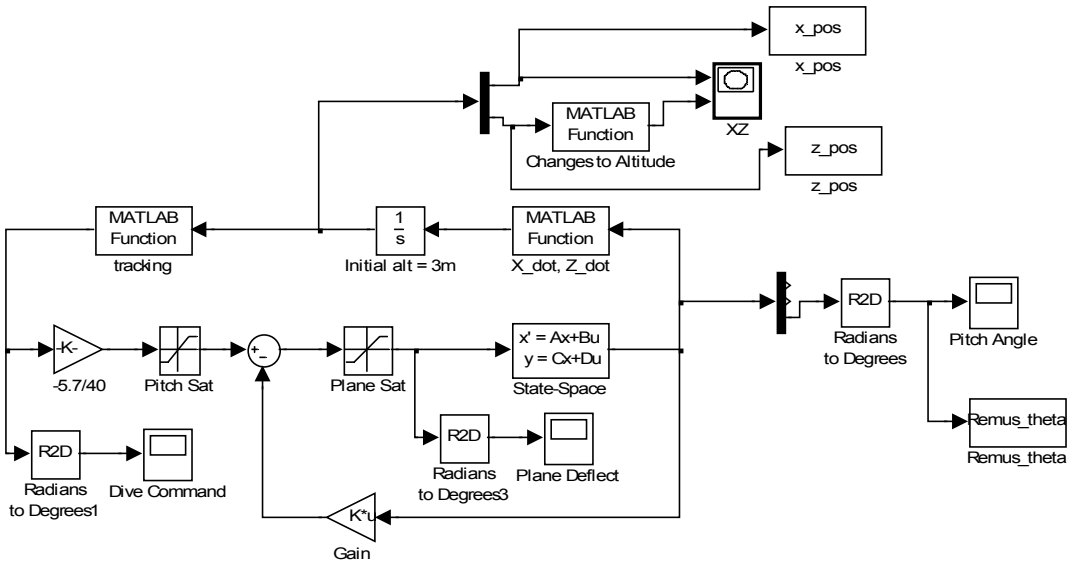

Figure 6. SIMULINK Model

The "State Space" block contains the state space matrices "A' and "B" that were derived in Chapter II. The Matlab Function block "X_dot, Z_dot" takes the outputs $w, q,$ and θ from "State Space" and uses Equations 5 and 6 from Chapter II to calculate $\dot{X}$ and $\dot{Z}$. $\dot{X}$ and $\dot{Z}$ are then integrated, resulting in the vehicles position (X, Z) which the Matlab "Tracking" code uses to calculate the vehicle's pitch command. The pitch command is fed back into the state space system along with the state-feedback gains to close the loop. The state-feedback gains -1.6807, 0.2935, and 0.0889 were determined with a simple pole placement technique of using the Matlab command "place." The location of the "placed" poles, -.5+.866i, -.5-.866i, and -1, were found using the Butterworth pole pattern method [10].

To keep the simulations as accurate as possible, real-world mechanical vehicle limitations were incorporated into the SIMULINK model. First, "Plane Sat" was added to limit the stern plane deflections to +/- 0.4 radians (22.9 degrees) due to stall restrictions [8]. Next, "Pitch Sat" prevents the ordered pitch angle from exceeding the maximum pitch achievable by the vehicle. The model's maximum achievable vehicle pitch was determined to be +/- pi/3 radians (60 degrees) by placing and holding the stern planes at their maximum deflection for an extended period of time. Lastly, the -5.7/40 multiplying factor was included to the pitch command for a desired one-to-one ratio in the ordered and responding vehicle pitch.

C. PATH FOLLOWING SIMULATIONS

Simulations were conducted with the vehicle trying to follow a "generic" obstacle avoidance path generated by a Gaussian potential function. The Gaussian function was initially chosen based on Hemminger's reasonable obstacle avoidance success using potential functions [7]. Hemminger concluded, "The characteristics of the potential function alone control vehicle avoidance maneuvers by creating a repulsive field around an obstacle that forces the vehicle to trace the potential field in order to regain its commanded trajectory. Also, not only is the obstacle avoidance path smooth and efficient but its magnitudes can be updated and optimized by assigning certain parameters with the actual Gaussian potential function."

In one-dimension, the Gaussian function is the probability function of the normal distribution as shown in Figure 7.

$$P(x) = h_o * \exp^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

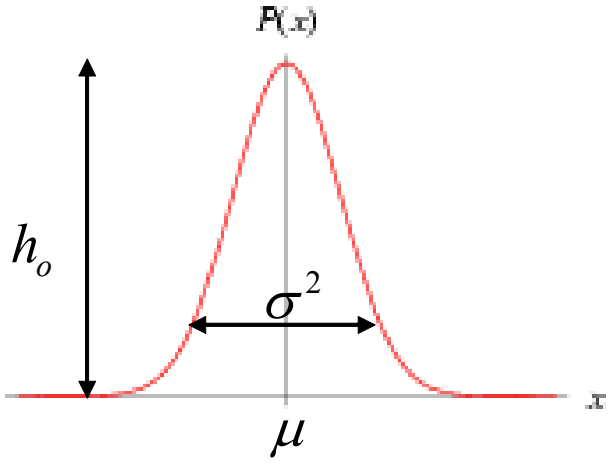

Figure 7. Gaussian Function (After: www.mathworld.com May 2006)

The multiplying parameter, h_o, linearly affects the maximum height of the potential curve at the location of its mean, μ, therefore, increasing h_o by 10 will also increase the maximum height by 10. The parameter σ^2 is the function's variance and represents the width or distribution of the function. The choice of the parameters determines the resulting ordered trajectory for the vehicle that can be customized to satisfy various mission goals. In the case of generating an obstacle avoidance path, h_o would be the desired vertical clearance of an obstacle located at μ, and σ^2 would determine the execution and termination distance from the obstacle for the avoidance trajectory. For the controller path following trials, a Gaussian function with $h_o = 2$, $\sigma^2 = 25$ and a mean at 60 meters were used. The "generic" path would order the vehicle to begin pitching up about 15 meters prior to the location of the obstacle and provides an altitude change of two meters above its original course. Figure 8 shows the simulated vehicle's response in attempts to follow the Gaussian path using the designed pitch controller.

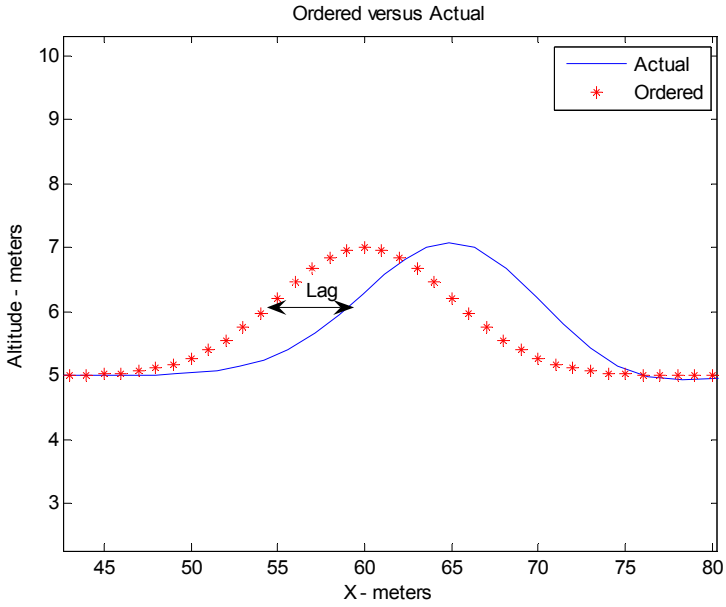

Figure 8. Ordered Versus Actual

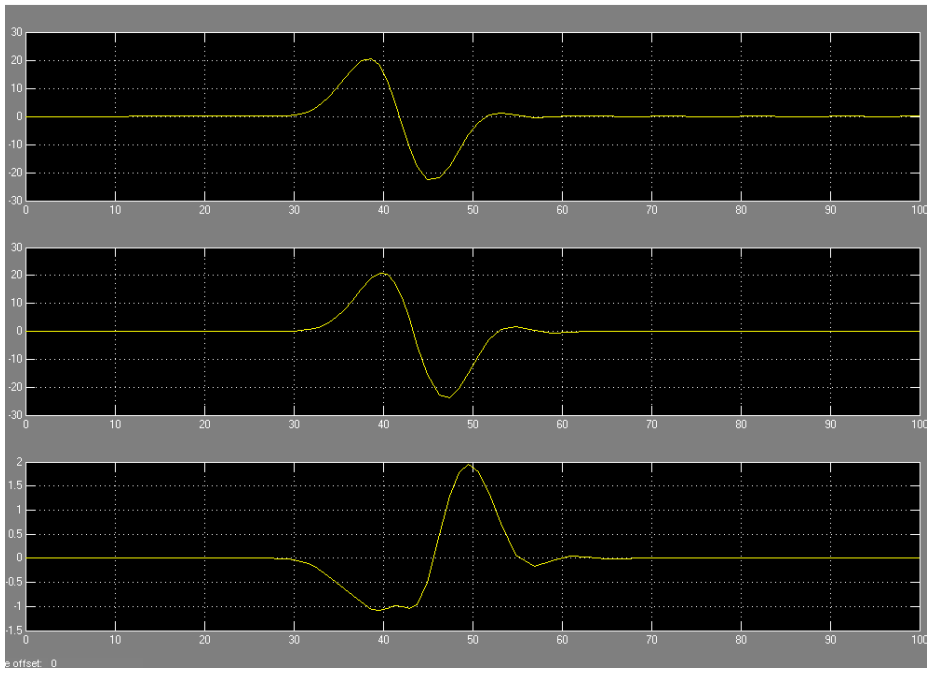

Figure 9. Ordered Pitch, Actual Pitch, Stern Plane Deflection in Degrees

Although the desired contour is achieved using the LOS pitch controller, a horizontal lag of about five meters exists between the "Actual" and "Ordered" plots. The commanded and responding vehicle pitch shown in Figure 9 are nearly identical and display a one-to-one ratio. Also, the stern plane deflection in relation to its saturation limit is small and indicates that the vehicle could handle more radical contours if needed.

The controller's path-following error was defined as the vertical difference between the actual altitude of the vehicle and the altitude of the ordered path at a given "X". The Matlab file "Controller_Errors" [Appendix] uses the following equations, to calculate and plot the vertical and total errors for the entire simulation.

$$error(X) = \sqrt{(Zpos(X) - Zpath(X))^2}$$

$$total\_error(X) = \sum_{1}^{n} error(X_n)$$

$$average\_error = \frac{total\_error}{n}$$

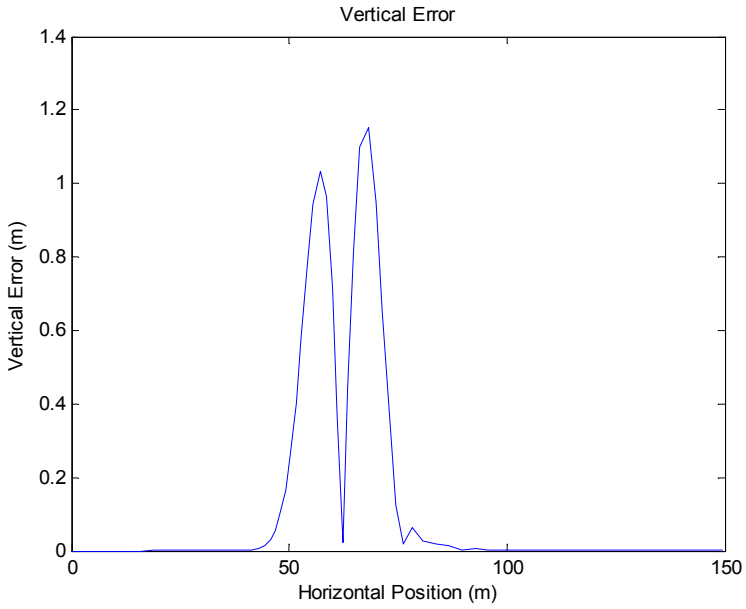

Figure 10. Vertical Error

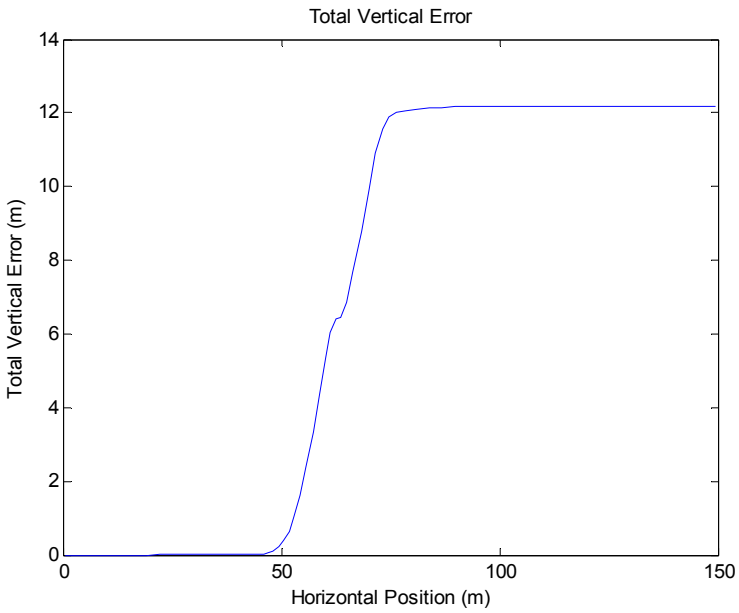

Figure 11. Total Vertical Error

The maximum vertical deviation was found to be 1.152 meters with an average error of 0.1716 meters, and the total error stabilized at 12.1849 meters. Although an average error of 0.1716 meters may seem reasonable, having deviations reach 1.152 meters at times represents a very high relative error of 57.6% when compared to an ordered two-meter altitude rise. In Section D, the design of an effective path-following controller continues as two approaches are studied in attempts to minimize the current controller errors.

D. REMOVING THE LAG

1. Including the Path's Slope

The first attempt to reduce the lag and large relative controller error was to include the slope of the ordered path into the pitch command calculations. The pitch command would be the result of adding the correcting angle and the slope of the path, therefore, in theory increasing the response of the vehicle. The addition of the slope would also prevent any rapid growth in the deviations in the event of an extreme track slope. The definitions governing this theory are shown below in Figure 12.

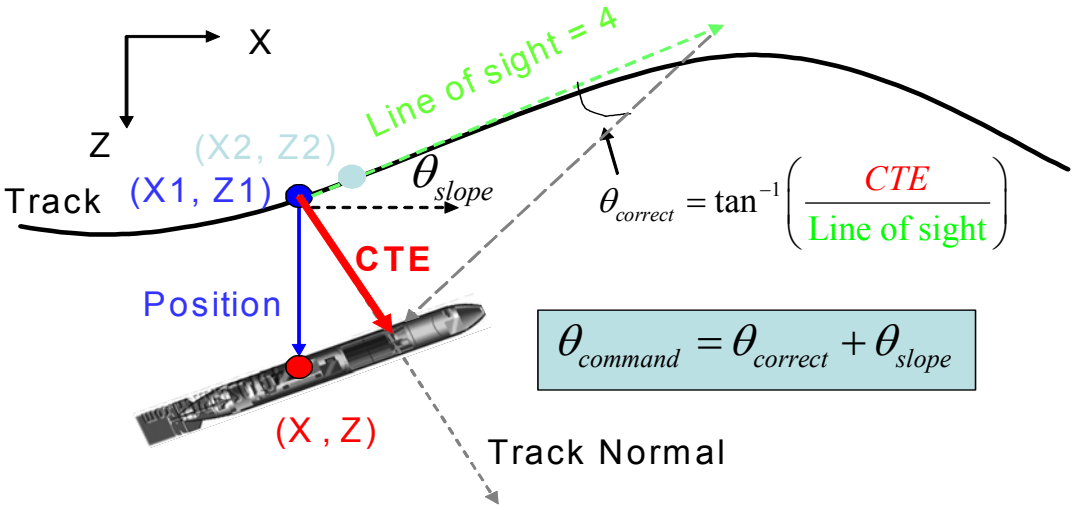

Figure 12. Slope Addition Definitions

All the definitions regarding the coordinates remain the same as before and the slope of the path, θ_{slope}, is found by taking the arc tangent of the difference between "Z2" and "Z1" and "X2" and "X1". $\theta_{correct}$ is calculated exactly like the pitch command from the previous approach and added to θ_{slope} to obtain $\theta_{command}$. The Matlab code "Tracking_plus_slope" [Appendix] was written using these definitions and replaces the previous "Tracking" code in the SIMULINK model. The simulation was repeated as before and the vehicle's path-following response utilizing the slope addition controller is shown in Figure 13.

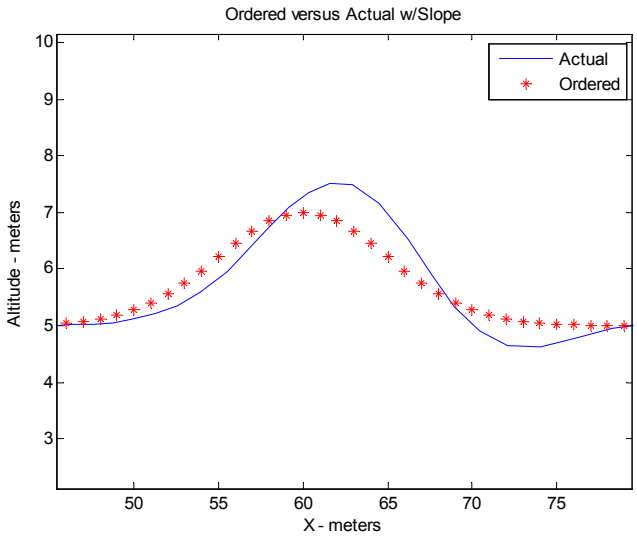

Figure 13. Order versus Actual with Slope

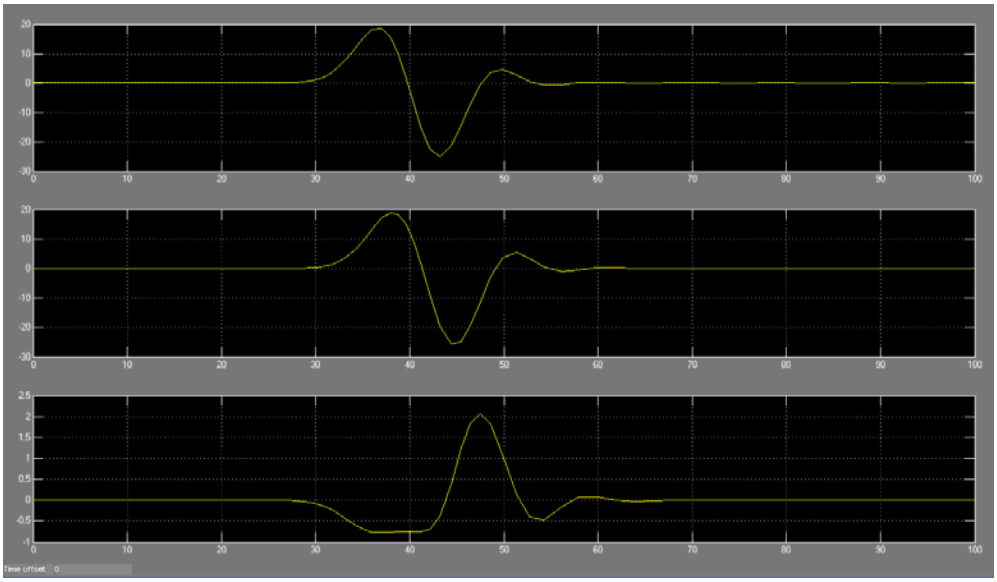

Figure 14. Ordered Pitch, Actual Pitch, Plane Deflection with Slope

The horizontal lag from the vehicle's original response has been reduced from five meters to approximately one meter by incorporating the track's slope into the pitch command. Although improvements with the lag issue have been made, overshoots at the peak and conclusion of the Gaussian path of almost one-half meter have been introduced. The controller errors were calculated in the same fashion as before to accurately evaluate any increases or decreases in the controller performance.

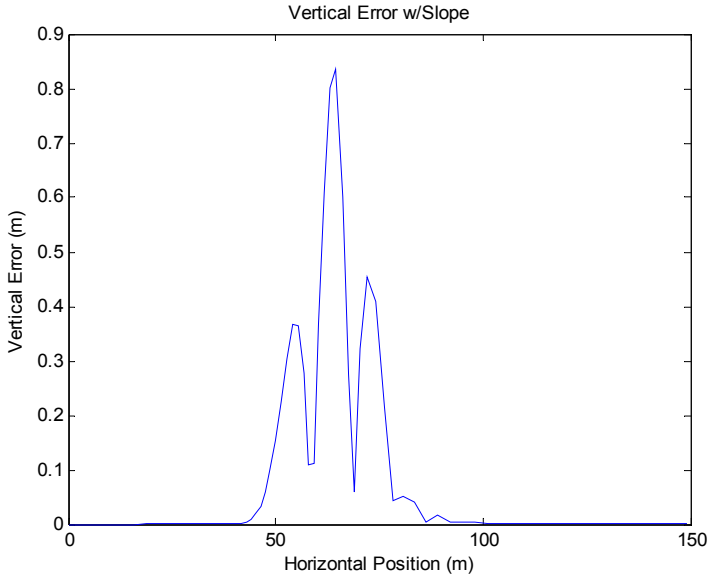

Figure 15. Vertical Error with Slope

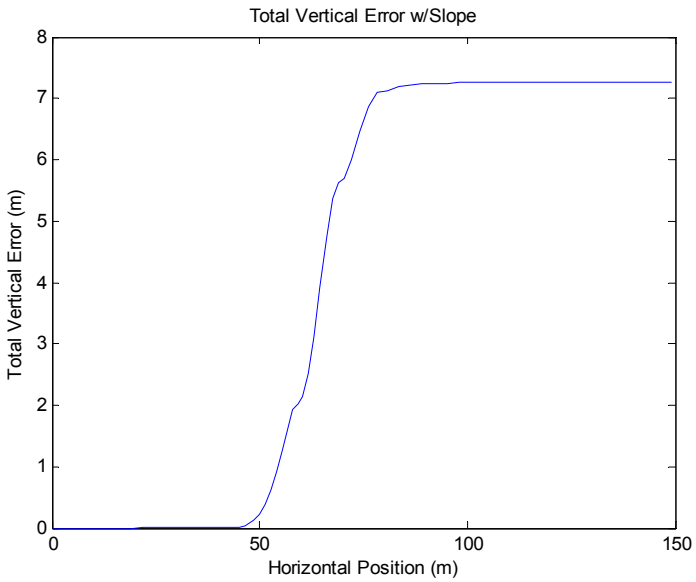

Figure 16. Total Vertical Error with Slope

The inclusion of the path's slope showed an improvement over the original LOS pitch controller in all three errors. The maximum vertical error was reduced by 27.4% to 0.8365m, the average error by 42.02% to .0995 meters, and the total error by 40.4% to 7.2607 meters.

2. Include a "Look Ahead"

Controller performance improvements were made with the slope-inclusion approach; however, the maximum vertical error still resulted in an unacceptable relative error of 41.8% to the ordered two-meter altitude change. The overshoots also pose a problem when an autonomous mission requires precise maneuvering in an unknown hazardous environment. A second approach involving the vehicle "looking ahead" was tested in an attempt to reduce the controller errors within acceptable tolerances. When "looking ahead," the vehicle's pitch command is calculated identical to the original LOS approach, however, it uses coordinates at a "look ahead" distance from the vehicles current position. A visual representation to this approach is shown in Figure 17.

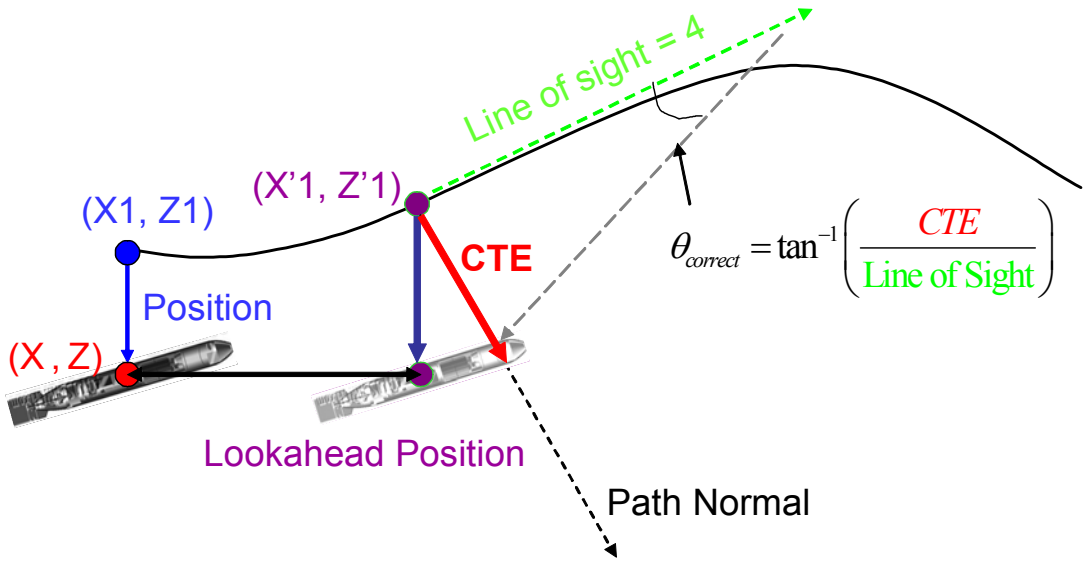

Figure 17. Look ahead Definitions

Using this approach, the vehicle receives a pitch command that it would have originally received further down the track, therefore, increasing the controller's responsiveness. For example, Figure 17 shows the pitch command being calculated based on the upcoming climbing trajectory instead of the diving trajectory that the vehicle is currently on. The Matlab code "Tracking_lookahead" [Appendix] included this philosophy and was written for calculating the new vehicle pitch command.

The SIMULINK simulations were once again conducted using the same Gaussian function as before and "Tracking" was replaced with "Tracking_lookahead" in the Matlab Function block. The initial simulation was conducted with a "look ahead" set at the horizontal lag distance of five-meters. Various distances were then tested to find the ideal "look ahead" and summarized in Table 3.

Various Look Ahead Distances			
ho=2m	sig = 5	U=1.5 m/s	aim = 4
Look Ahead	Max Error	Avg Error	Total
5	0.1686	0.0228	1.6165
4.9	0.1591	0.0198	1.4276
4.8	0.1489	0.0177	1.273
4.7	0.1389	0.0158	1.1373
4.65	0.1368	0.0148	1.0688
4.6	0.1346	0.0145	1.0405
4.59	0.1342	0.0145	1.0405
4.58	***0.1337***	***0.0145***	***1.0413***
4.57	0.1333	0.0146	1.0491
4.56	0.1329	0.0147	1.058
4.55	0.134	0.0148	1.0668
4.54	0.1364	0.0149	1.0756
4.53	0.1389	0.0151	1.0844
4.52	0.1413	0.0152	1.0933
4.51	0.1437	0.0153	1.1051
4.5	0.1462	0.0156	1.12
4.4	0.1703	0.0177	1.2761
4.3	0.194	0.0202	1.4526
4.2	0.2176	0.0228	1.6444
4.1	0.2413	0.0258	1.8559
4	0.2649	0.029	2.0889

Table 3. Errors at Various Look Ahead Distances

Since the three minimal controller errors occurred at different "look ahead" distances, the ideal "look ahead" was determined to be 4.58 meters and used in all future simulations, as it caused the greatest reduction in all three errors. The following three figures compare the response and errors for all three approaches.

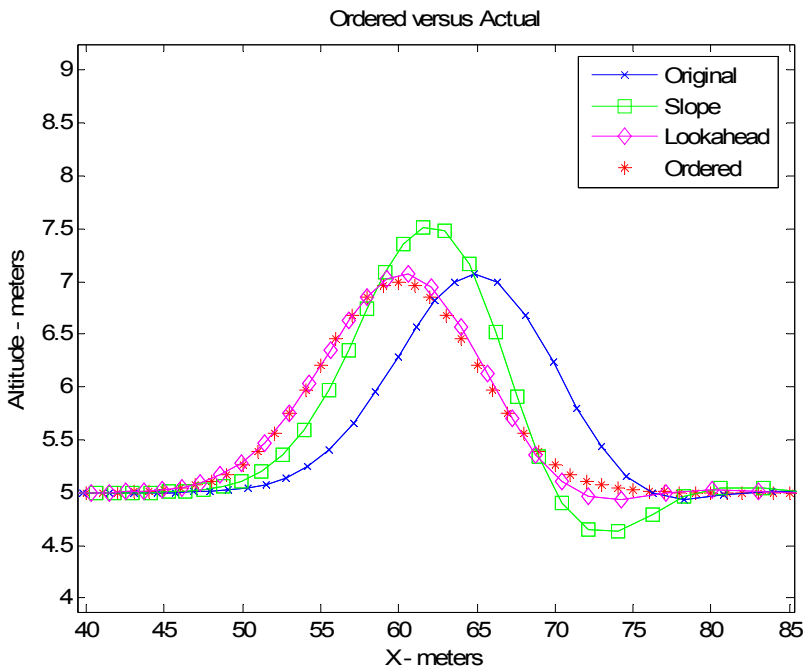

Figure 18. Ordered vs. Actual with Look Ahead

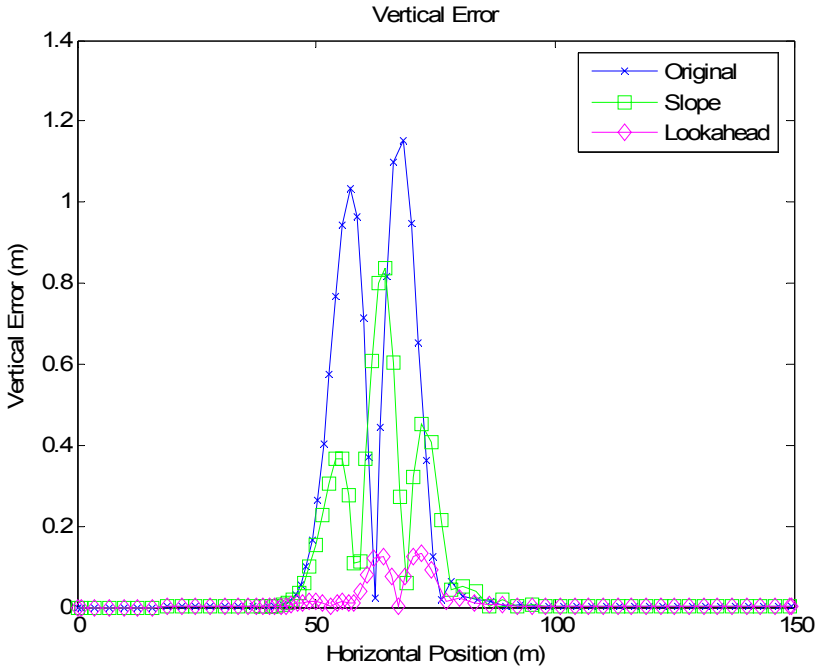

Figure 19. Vertical Error with Look Ahead

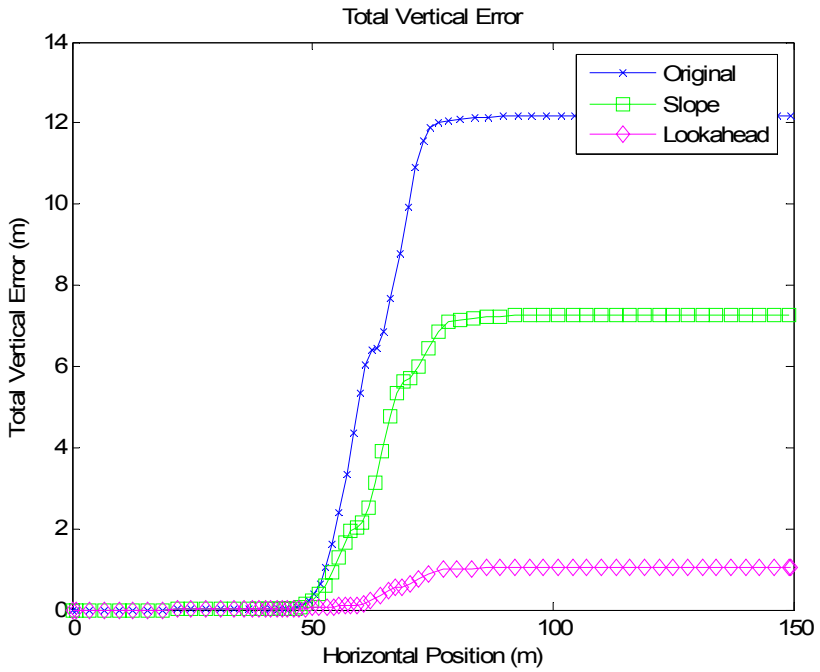

Figure 20. Total Vertical Error w/Look Ahead

Using this "look ahead" reduced the original maximum vertical error by 88.46% to 0.1329 meters with an average error of only 0.0145 meters. The total error was reduced by 91.46% to 1.0405 meters. Table 4 summarizes the controller errors for all three controllers and their improvement over the original errors.

Approach	Max Error	Redution %	Avg Error	Redution %	Total Error	Redution %
Original	1.152		0.1716		12.1849	
Slope Addition	0.8365	27.39%	0.0995	42.02%	7.2067	40.86%
Look Ahead	0.1329	88.46%	0.0145	91.55%	1.0405	91.46%

Table 4. Summarization of Controller Errors (Meters)

It is evident that the "look ahead" approach successfully reduces the original horizontal lag problem, and unlike the slope addition approach, does so without the introduction of large vertical overshoots. The drastic reduction in controller errors to within acceptable tolerances support the fact that the first requirement of an effective path following controller for the obstacle avoidance "back-seat driver" has been met.

IV. OPTIMAL SENSOR ORIENTATION FOR OBSTACLE AVOIDANCE PLANNING

A. PATH PLANNING STRATEGY

With the design of an adequate path following controller, the remainder of this thesis addresses effective obstacle avoidance path planning for the REMUS's "back-seat driver". An adaptation of what Horner [12] described as some constraints associated with optimal reactive avoidance path planning are:

- Avoid obstacle
- Smooth continuous navigation
- Vehicle limitations
- Optimal sensor orientations

The list of constraints is used as a basis in developing an effective strategy for REMUS's "back-seat driver". The obstacle avoidance path must first and foremost provide safe and collision free navigation for the vehicle. Collision free is a clear cut requirement and for simplicity reasons, a "safe" path was deemed as avoiding the obstacle by a minimum vertical distance of two-meters. The freedom of the path is further limited by the requirement that the path must be smooth and continuous for pitch command calculations. Next, the vehicle limitations such as maximum pitch angle and stern plane deflection, processing/information relay time, and actuator lag time can all have an affect on the sharpest path curvature the vehicle would be able to accurately follow, also known as the vehicle's maximum achievable turning radius. Since the vehicle's maximum pitch and stern plane deflection are the most limiting on the vehicle's turning radius, they are the only two vehicle limitations incorporated into the Simulink model.

The first three path planning constraints discussed above consist of predetermined variables and provide limited effect to an obstacle avoidance strategy. Optimal sensor orientation on the other hand varies within dynamic unknown environments and is the focus for this chapter. To date, NPS research on optimal sensor orientation has mainly dealt with the orientation of the vehicle's side scanning sonar, which found that flying at a low fixed altitude would provide consistent sonar images and reduce the "near-nadir"

region [12]. The implementation of the FLS as the vehicle's primary obstacle avoidance sensor introduces another element to the optimal sensor orientation constraint. Now the vehicle must be positioned in such a way to maximize the information gathered by the FLS about upcoming hazards, while trying to maintain the low-fixed altitude for the side scanning sonar.

B. MODELING

1. Environment

Models of possible hazardous environments ["Ocean_Model," Appendix] were created in MATLAB to explore the issues governing optimal sensor orientations. Since REMUS's auto-pilot is currently unable to steer to sharp rises [5], the environment modeled consists of either one or two sea wall(s) present on a flat ocean floor. A sea wall is also a simple representation for the FLS's perspective of large protruding obstacles sitting on the ocean bottom.

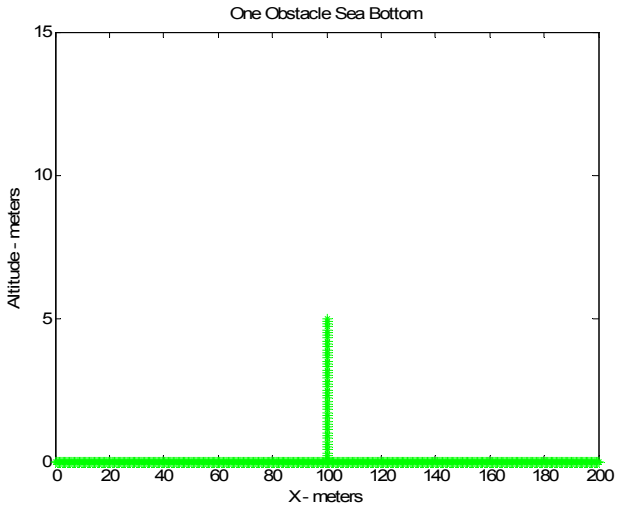

Figure 21. Ocean Model with One Sea Wall

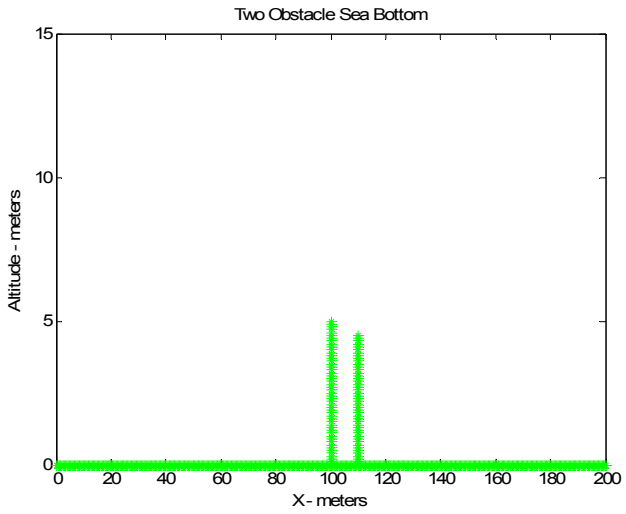

Figure 22. Ocean Model with Two Sea Walls

The first ocean model, shown in Figure 21 contains a five meter sea wall located 100 meters from the global origin. The second model including two sea walls was created to study the situation when an obstacle is located within a FLS occlusion area created by the first obstacle. The second sea wall located 10 meters aft and a half-meter shorter then the first sea wall will be undetected when the vehicle is flying at the fixed altitude of three meters; a problem which is discussed in more detail later in this chapter.

2. Sonar

Although the FLS installed on REMUS contains two sonar staves, for simplification, only one stave was modeled using the MATLAB code ["Sonar," Appendix] to represent the two-dimensional field of view for the vehicle. The vehicle's field of view was defined as the area bounded by the upper most sonar beam, the lowest most sonar beam, and the nominal maximum sonar range of 100 meters as shown in Figure 23.

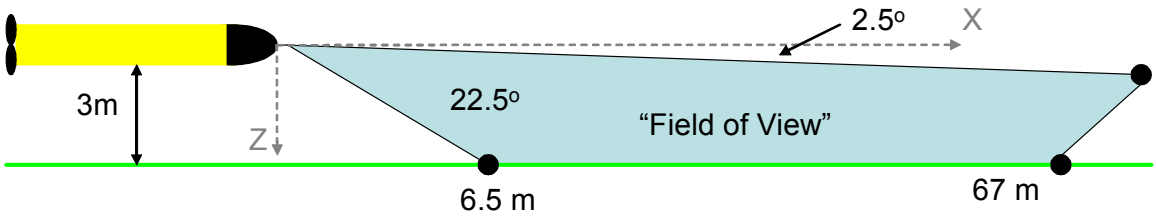

Figure 23. 2-D Field of View for REMUS

The sonar's beam width of 22.5 degree was broken up into 225 beams to simulate the ProViewer multi-beam sonar. The high number of beams were required to obtain accurate and smooth sonar images at ranges of 50 meters or greater. The sonar image was created by calculating where and if each sonar beam, originating from a given vehicle position and pitch, intersected the modeled environment. All the interception points were then gathered and plotted to create a simulated sonar image. If an occlusion was present, it was represented by a red line which bounded the "unseen" area. Figures 24 and 25 show the simulated sonar images plotted below a graphical representation of the vehicle/sonar position and orientation. The vehicle is indicated by the blue circle and the sonar beams by the black dashed lines. It should be noted that the axes vary largely in scale and cause a distorted representation in slopes and angles.

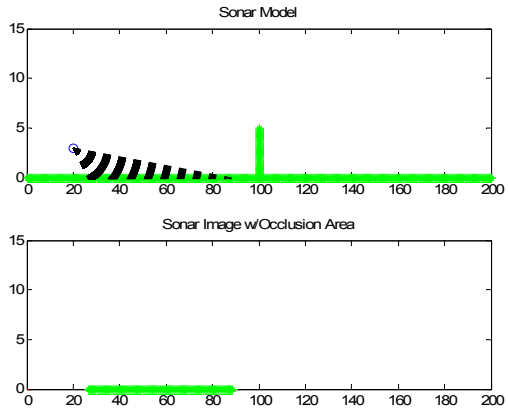

Figure 24. Simulated Sonar Image of an Ocean Floor

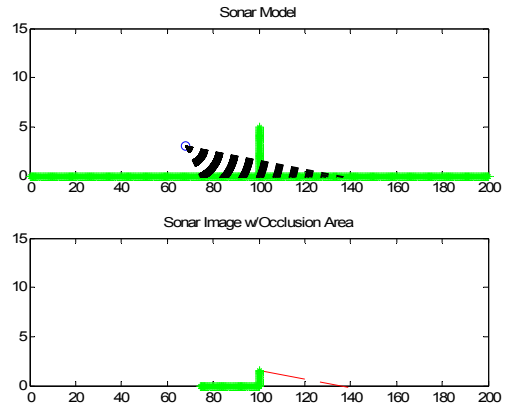

Figure 25. Simulated Sonar Image of a Sea Wall and Occlusion

To model the sonar for a moving vehicle, "Sonar" was modified with a "for" loop to repeat the imaging with varying vehicle position and pitch during a simulated obstacle avoidance run ["Sonar_Moving, Appendix]. A video clip of the sequential sonar images is generated to be used for studying possible sensor orientation issues.

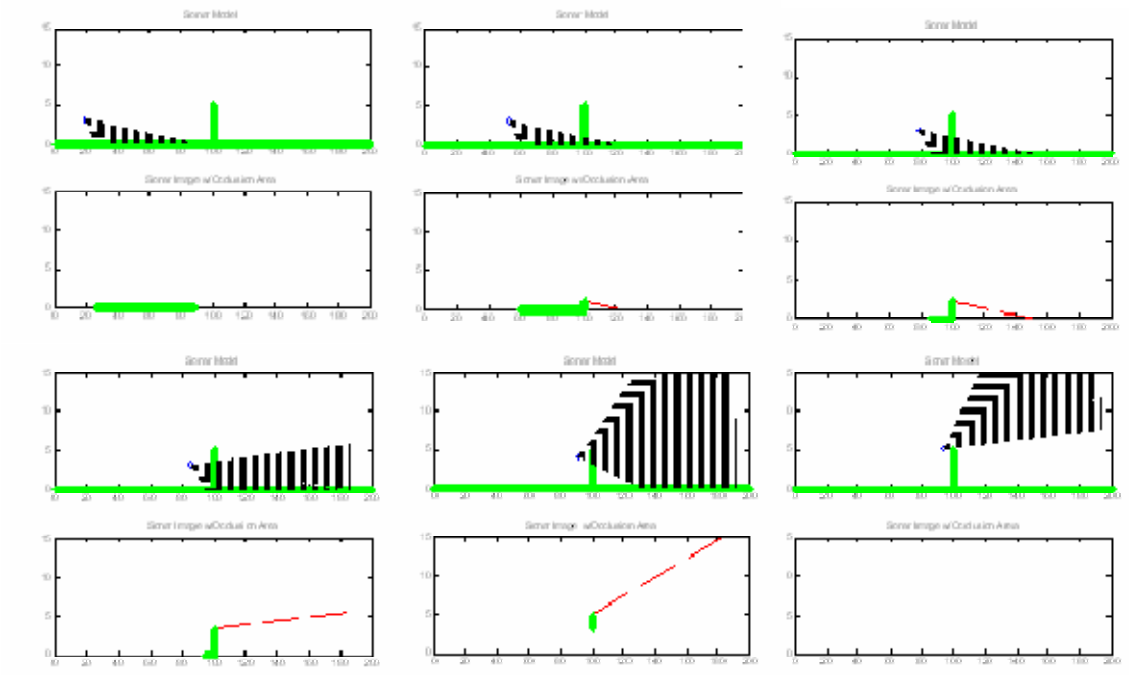

Figure 26. Stills from a Video Generated by "Sonar_Moving" Matlab Model

C. FLS ORIENTATION ISSUES

1. Limited Field of View

In order to satisfy the initial optimal sensor orientation constraint of minimizing the "near-nadir" region, the vehicle flies at a low fixed altitude of three meters. However, this increases the likelihood of encountering obstacles that protrude higher off the ocean floor then the vehicle is flying. Since the sonar is configured to search 2.5 degrees below the vehicle's zero-pitch horizon, the obstacle will always saturate the vehicle's downward-angled field of view if it remains on its fixed altitude path.

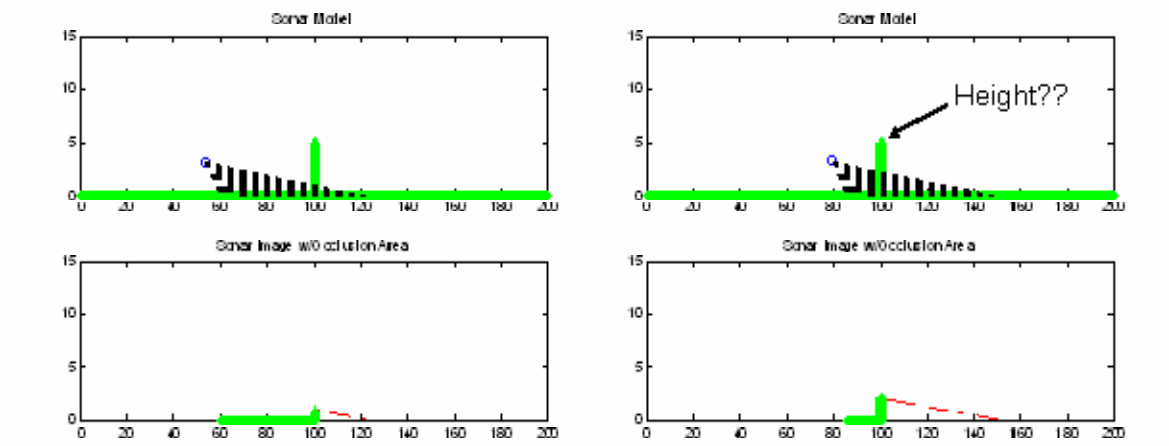

Figure 27. REMUS Unable to See Entire Obstacle

As shown in Figure 27, without pitching the vehicle upward, the vehicle never "sees" the entire five meter obstacle and the vehicle would have to guess a safe altitude required to clear the obstacle.

2. Occlusion Areas

Even if vehicle correctly "guessed" the altitude required to clear the obstacle, there still exists a problem if another obstacle lies within the occlusion area brought on by the first obstacle. Normal obstacle avoidance methodology of projecting a typical Gaussian may provide a solution for the first obstacle, but fails to safely avoid the previously "unseen" obstacle. Figure 28 shows the standard Gaussian path that would be generated based on only the information of the "seen" obstacle.

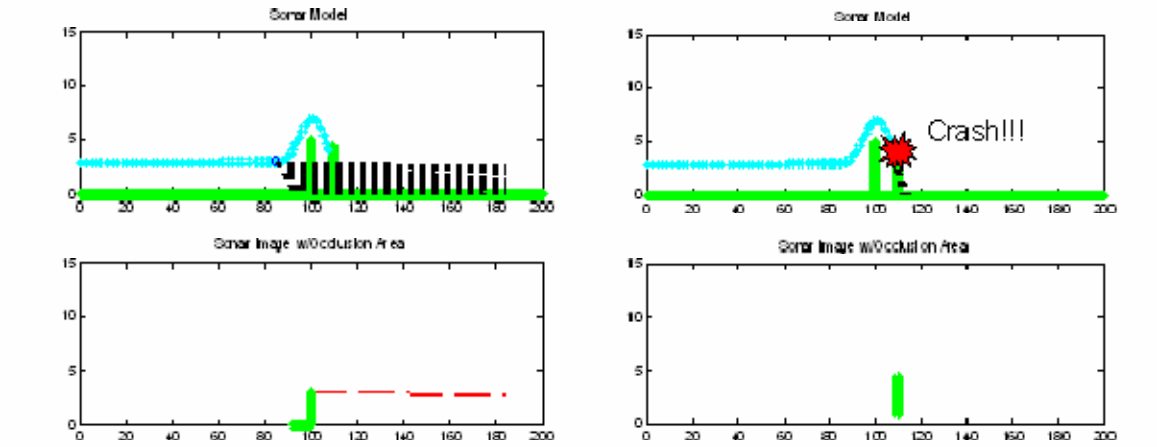

Figure 28. Stills Showing Occlusion Problem

The blue path is the ordered obstacle avoidance path. The vehicle first detects the second obstacle at the peak of the Gaussian curve as it pitch back down towards its original ordered altitude. Current methods prevent the predetermined Gaussian from being altered mid-course; therefore the vehicle is unable to avoid the recently detected obstacle. In Chapter V, two approaches are developed in an attempt to achieve optimal obstacle avoidance path planning while incorporating the additional FLS orientation constraints discussed above.

THIS PAGE INTENTIONALLY LEFT BLANK

V. OPTIMAL REACTIVE OBSTACLE AVOIDANCE

A. GAUSSIAN POP-UP

The first method attempting to deal with the FLS orientation issues discussed in Chapter IV includes ordering the vehicle to conduct a "pop-up" when a possible threat has been detected. Figure 29 illustrates the methodology behind the "pop-up" method.

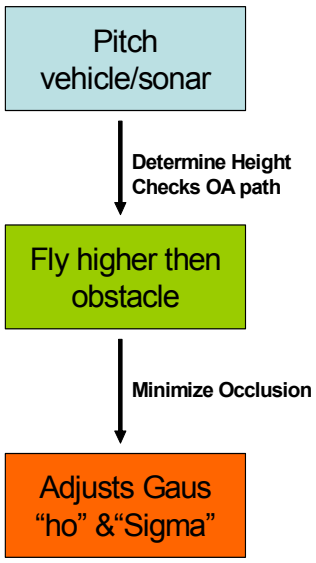

Figure 29. The "Pop-Up" Methodology

The first stage of the pop-up pitches the vehicle upwards therefore tilting the downward-angled sonar upward. This increases the sonar's vertical field of view and allows the vehicle to "see" the entire obstacle to determine its height. It also allows the surveying of the area above the obstacle to ensure clear and safe waters for an obstacle avoidance path. The "pop-up" will then drive the vehicle to a high enough altitude so the sonar can look above and behind the detected obstacle. This will minimize the uncertainties in the occlusion area and provide additional information regarding any possible obstacles that were previously undetected. Finally, the original Gaussian obstacle avoidance variables "ho" and "sigma" are updated to provide a safe path based on the recently gathered information.

A Gaussian potential function was chosen as the "pop-up" trajectory for the same reasons explained in Chapter III, and it parameters "h_o" and "σ^2" were set to 5 and 25

respectively. A "h_o" of five meters drives the vehicle to a total altitude of eight meters or three meters higher then the first modeled obstacle, and a smaller "σ^2" decreases the horizontal distance and time the pop-up deviates from its original fixed altitude. The simulation was conducted with the Matlab file "Tracking_Popup" [Appendix] implemented into the SIMULINK model. Figure 30 shows the vehicle's response to the ordered Gaussian pop-up.

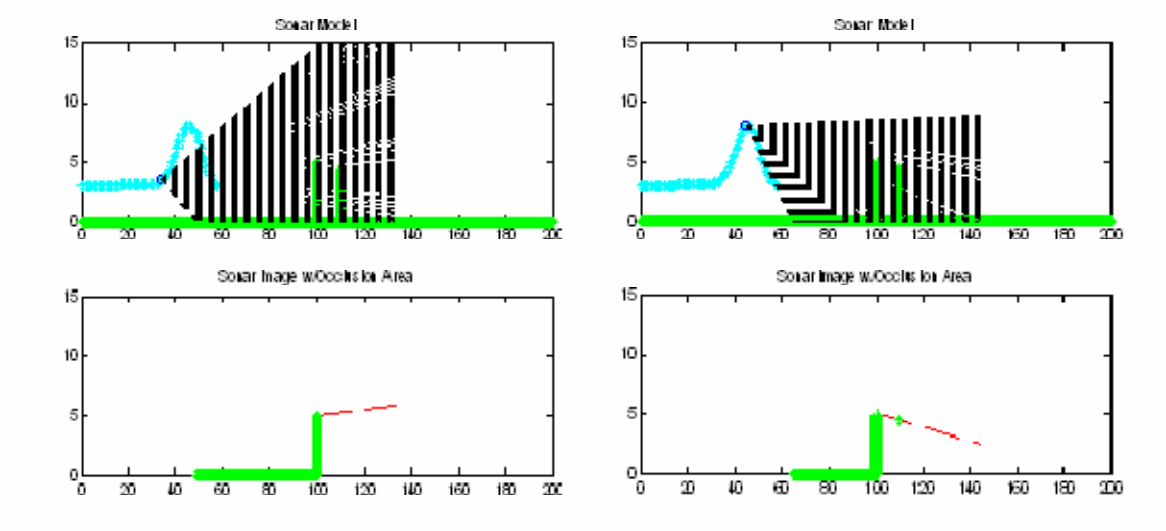

Figure 30. Height Determination and Occlusion Minimization using a Gaussian Pop-Up

The picture on the left shows the determination of the obstacle's height and the verification of clear waters. The pitching of the vehicle allowed the sonar to increase its vertical field of view from under three meters to over fifteen meters. Secondly, as the vehicle was driven higher then the obstacle, it allowed the sonar to detect the previously unknown second obstacle as shown in the second picture. With the amplifying environment information, the obstacle avoidance Gaussian's "h_o" and "σ^2" were then adjusted to 3 and 144 respectively to account for all detected obstacles. Figure 31 shows the vehicle response to the entire ordered path generated using the "pop-up" method.

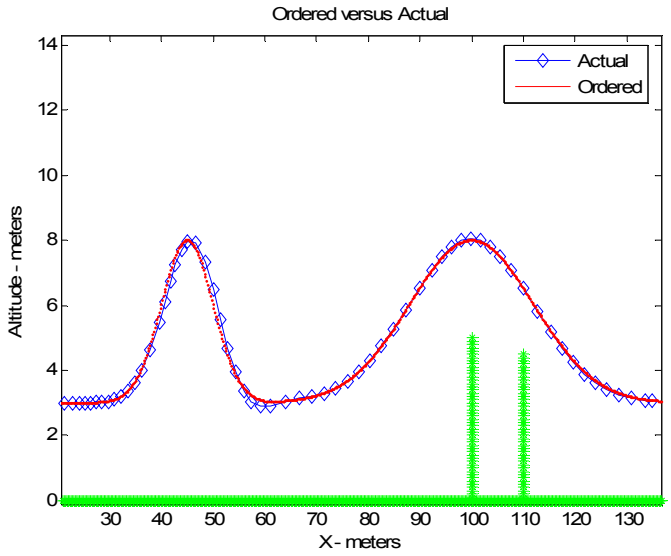

Figure 31. Response to Pop-Up

By comparing the path generated to the previously defined constraints for optimal obstacle avoidance, the "pop-up" method's effectiveness was determined. Most importantly, the path provided a collision free path that vertically cleared all obstacles by a minimum of two meters. Secondly, due to the characteristics of the Gaussian functions used, the path was smooth and met the requirements of containing no sharp corners or jumps for pitch command calculations and consistent sonar images. The designed pitch controller was also able to accurately follow the path with a maximum controller error of 0.197 meters. With regards to optimal sensor orientation for the side scanning sonar, the magnitude of the deviations from the fixed altitude of three meters was quantified for later comparison. The total vertical deviation (TVD) was defined as the area above the desired altitude bounded by the obstacle avoidance path and calculated to be 212.78 meters^2 using the following trapezoid approximation equation ["TVD_Cal," Appendix]:

$$TVD = \sum_{j=2}^{N} \left(\frac{Z_j + Z_{j-1}}{2} - 3meters \right) \left(X_j - X_{j-1} \right)$$

Lastly, the Gaussian pop-up reasonably dealt with the two issues surrounding optimal sensor orientation for the FLS. The detected obstacle's height was able to be determined by pitching the vehicle upwards and the second obstacle was briefly detected within the occlusion area which enabled the planner to provide a safe collision-free path.

Although the Gaussian pop-up method successfully cleared the obstacles in this simulated case, certain factors limit the method's robustness. For example, in a real world environment, the vehicle will encounter additional situations to the one simulated such as a third obstacle located in the occlusion area created by the second obstacle. The third obstacle may not be "seen" during the initial pop-up and is first detected when the vehicle pitches downward at the peak of the second Gaussian. The problem with not be able to alter the Gaussian mid-course once again arises; therefore, this method requires prior knowledge of all obstacles within the area to ensure a safe collision-free path.

Another limiting factor is caused by the pop-up's main purpose of driving the vehicle to an altitude above the detected obstacle to minimize the occlusion area. However, since the pop-up's purpose is to also determine the obstacle's height, the altitude it needs to drive up to in order to ensure an occlusion look is unknown prior to its execution. The pop-up is ordered using a predetermined "ho" which may be too high or not high enough to see over the obstacle and an optimal path for varying situations can not be guaranteed.

Lastly, the execution of the pop-up results in large deviation from the original fixed altitude due to the requirement of driving the vehicle to a high altitude A narrow "pop-up" would reduce the TVD, however, it would also increase the rising slope of the path and its peak's sharpness. These increases could cause inconsistent sonar images and decrease the number of "looks" the sonar had at the obstacle or occlusion area due to larger pitch velocities. To classify obstacles and accurately determine their location and height, real-world sonar imagery processing requires consistent and frequent returns. Due to the battling requirement for consistent and frequent returns, little improvements can be made to the pop-up method to reduce its TVD.

B. SPLINE ADDITION

The second method developed to deal with the FLS issues incorporates a mid-course Gaussian spline addition alteration in an attempt to achieve an optimal obstacle avoidance path. Figure 32 illustrates the methodology behind this approach.

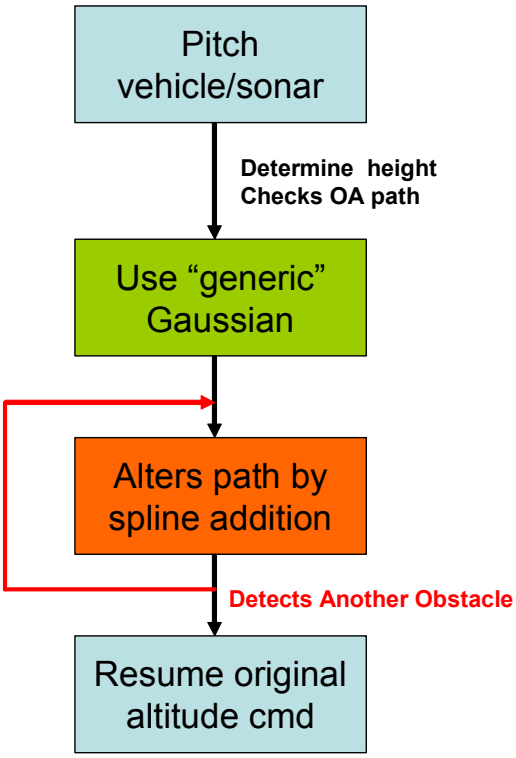

Figure 32. Spline Addition Methodology

When the vehicle initially detects an obstacle, it conducts a miniature predetermined Gaussian pop-up only to determine the height of the obstacle and check for clear waters. The small Gaussian keeps deviations from the original fixed altitude to a minimum and with parameter, $h_o = 0.75$ and "σ^2" = 4, the sonar's vertical field of view is increased from three meters to 10 meters. Simulations were again conducted using "Tracking_Spline" [Appendix] to implement the spline approach into the SIMULINK model. Figure 33 is the vehicles response and increased view while conducting the modified pop-up.

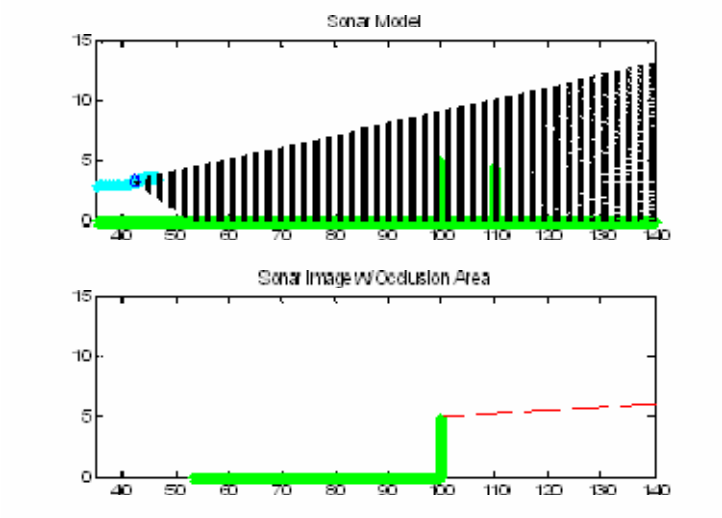

Figure 33. Height Determination for the Spline Method

After returning to its original altitude, a standard Gaussian path is then projected to clear the detected obstacle. As the vehicle reaches the peak of this Gaussian, it begins to pitch back downwards and gets an up-close look at the occlusion area. The sonar is now able to detect the previously unseen obstacle and the Gaussian is immediately altered with the use of a spline to provide a safe path.

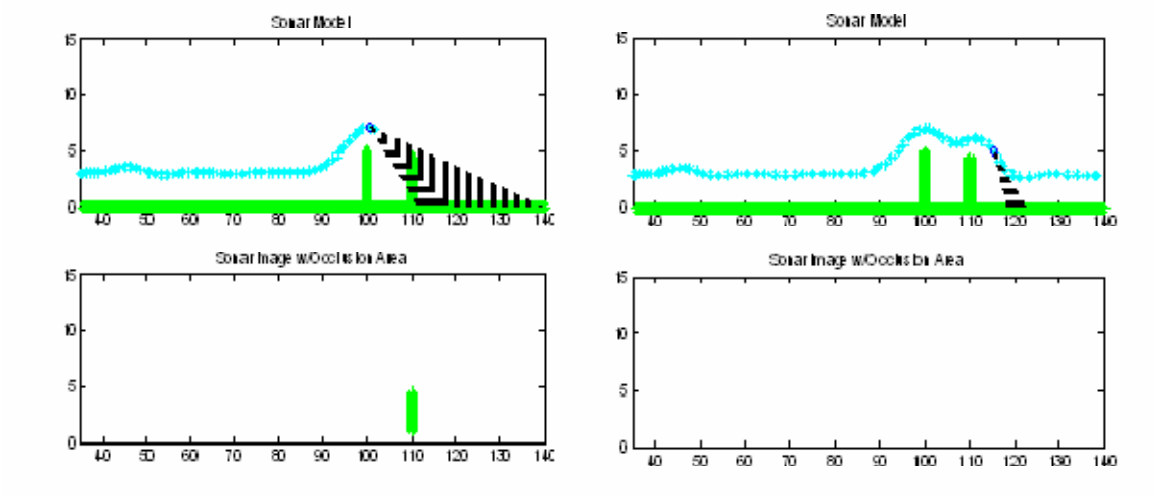

Figure 34. Occlusion Look and Obstacle Detection

The choice of a spline was based on its ability provide a continuous curve that can be varied based on its knot locations. The Matlab command "PP = SPLINE(X,Z)" provides a piecewise polynomial for the cubic spline interpolated to the values of "Z" at "X". "X" and "Z" are vectors, with "Z" containing two more values than "X," since the first and last value in "Z" can used as the end-slopes for the cubic spline. By including the end-slopes, a smooth equivalent slope can be assured when the path transitions from the Gaussian to the cubic spline, and then from the spline back to the ordered original altitude. Figure 35 shows how the "X" and "Z" vectors are calculated in the event another obstacle is detected.

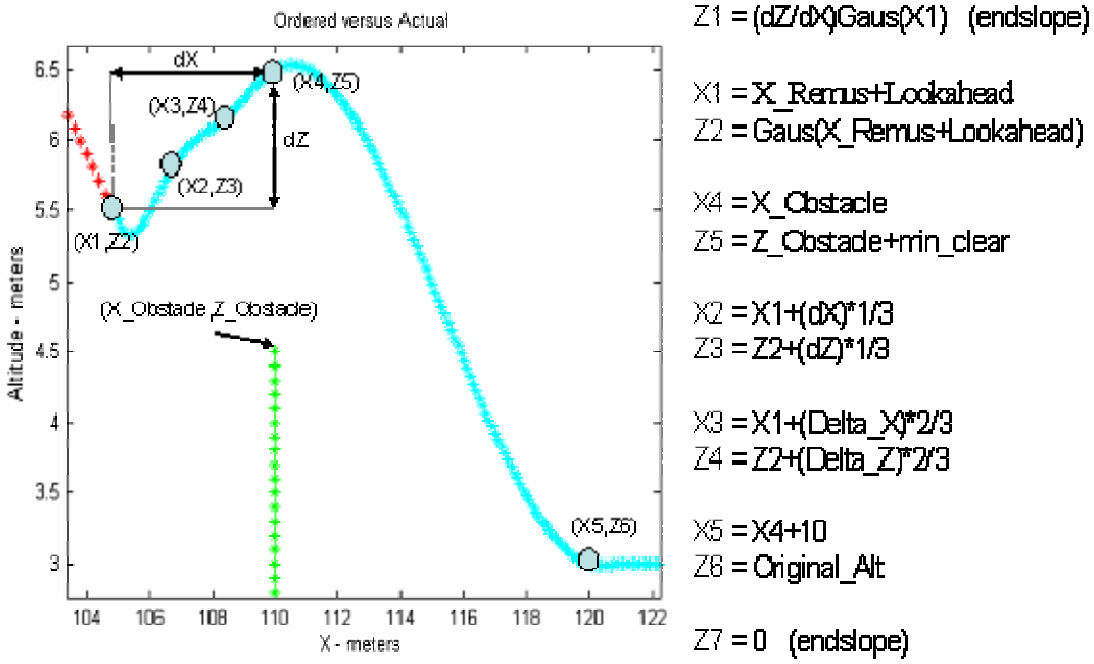

Figure 35. Spline Calculations

Once an obstacle is detected, the vehicle's location, X_Remus, is noted and the obstacles horizontal position and height are defined as X_Obstacle and Z_Obstacle. The spline's first knot or coordinate, (X1,Z2), is located at a horizontal look-ahead distance in front of X_Remus since the pitch commands are being calculated at that point and at an altitude equal to the Gaussian at that horizontal location. The first end-slope, Z1, is equivalent to the slope of the Gaussian at the adjoining location and found by taking the

derivative of the Gaussian at X1. Coordinates (X4, Z5) represent the peak of the spline and is located two meters directly above the second obstacle. Two knots are then placed between the first knot and the peak knot, which divide the horizontal and vertical distance between the two into thirds. The last knot brings the vehicle back down to the original three meter altitude, 10 meters behind the obstacle, with a slope of zero. Figure 36 shows the entire obstacle avoidance path generated using the spline method along with the response of the vehicle. Figure 37 provides a closer look of the spline addition portion of the path.

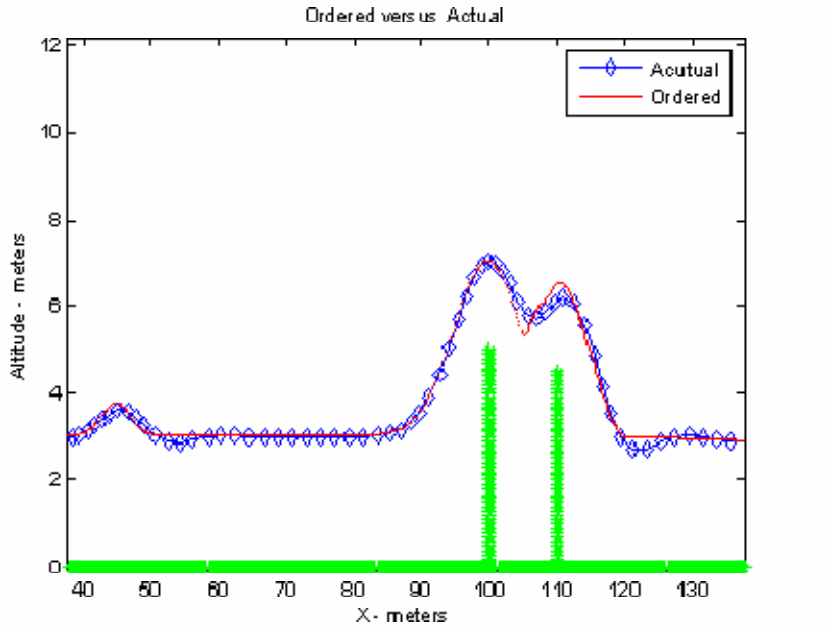

Figure 36. Ordered and Actual Vehicle Response Using the Spline Method

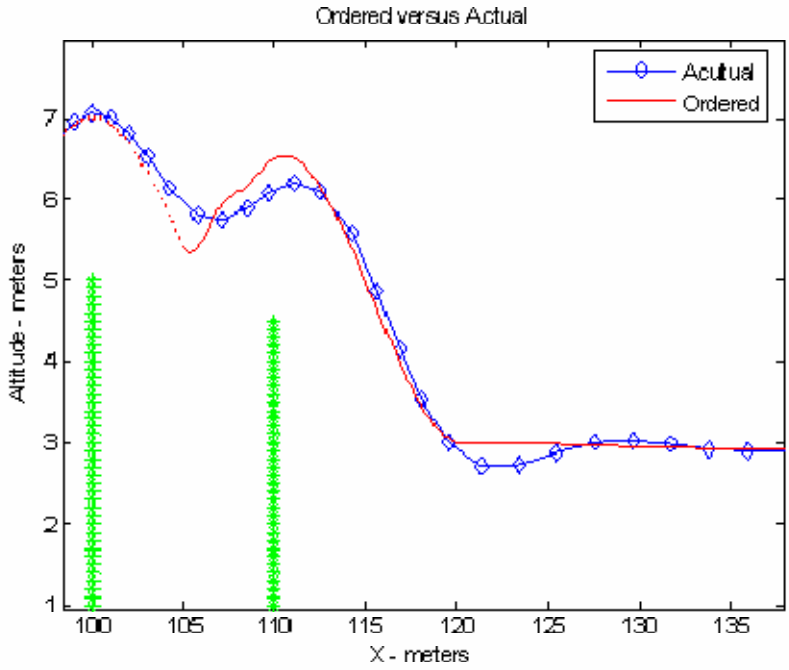

Figure 37. Enlarged Spline Portion

The results from the spline method were also compared to the defined constraints for optimal obstacle avoidance. The vertical path over the second obstacle was less then the desired "safe" two-meter minimum clearance, however, the navigation was more importantly collision-free. Also, this method is truly reactive and requires no prior knowledge of the obstacles since the spline can be implemented anywhere along the Gaussian, and the locations of its knots can be altered to provide even greater flexibility. This method is therefore robust and able to deal with a number of real-world situations in addition to the one simulated.

Continuity throughout the path was established by matching the start and end slopes of the spline with the end slope of the altered Gaussian and zero-slope of the original fixed altitude. The path's smoothness, however, is at the mercy of the knot locations creating the interpolating polynomial. There are combinations of knots will cause an undulating path or one that exceeds the maximum turning radius of the vehicle. The path created in the simulation contains sections that were not accurately followed by the look-ahead pitch controller, resulting in errors of 0.463 meters.

Lastly, the spline method provided effective positioning for REMUS's sensors. The TVD from the optimal side scanning sonar orientation was calculated to be 79.386 meters[2] and the small pop-up keeps majority of the TVD to be caused by avoiding the obstacle. Also, by only altering the backend of the Gaussian keeps the vehicle flying at its fixed altitude for as long as possible. The spline method successfully dealt with the FLS orientation issues by pitching the vehicle upwards to determine the obstacle's height, and the second obstacle was detected within the occlusion area. Since the avoidance Gaussian can be altered mid-course, the vehicle is no longer required to gather information about the occlusion area before-hand. The vehicle can now get a complete view of a occlusion by waiting until it clears the first obstacle and is positioned above and near the previously unknown area. The mid-course alteration not only allows minimization of the uncertainties, but it also allows this method to be "reactive," creating a path that can be updated due to real-time information about the environment and obstacles detected.

C. APPROACH COMPARISON

A metrics was used to compare and weigh the advantages between the two methods with regards to the optimal obstacle path constraints. A scale of one to five was used to measure the significance of any advantage one method has over the other. A five represents an advantage of high significance, while a one is of little significance, and dashes were used if no advantage existed between the two methods

		Pop-up	Spline
Avoid Obstacle	Min Clearance	1	-
	Robustness	-	5
Navigation	Smooth	3	-
	Contiuous	-	-
	Controller Error	2	-
Vehicle Limitations	Pitch	3	-
	Plane Deflection	-	-
Sensor Orientation	Vertical Deviation	-	3
	Occlusions	-	5
	TOTAL	9	**13**

Table 5. Comparison Metrics

Even though the spline method did not always achieve the "safe" minimum clearance of two-meters, it met the more important requirement of providing a collision-free path from all obstacles. Also, unlike the pop-up, the spline is able to adjust to varying environments and not just the isolated cases that were simulated. The pop up method has the advantage of producing a smoother path that wouldn't order the vehicle to exceed its maximum turning radius. The spline method, however, excels in the optimal sensor orientation for both of REMUS's sensors, which support its primary mission as an environment information gatherer.

The use of a spline addition offers a robust method required when operating in a unknown environment. Not only is it able to react to real-time updates, it also positions the vehicle to maximize the information gathered about the environment, therefore, the spline method proves to be the superior choice for the "back-seat driver's" path planner.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

One day fully autonomous AUV's will no longer require human interactions to complete its missions allowing the Navy to keep divers and valuable resources out of harms way. This thesis built on the previous work conducted at NPS's Center for AUV Research to improve the autonomy of the REMUS class of AUVs with an implemented FLS. The goal was to design a reactive "back-seat driver" to coincide with the normal altitude control auto-pilot to safely navigate the vehicle in the presence of previously unknown obstacles.

For the first "back-seat" driver requirement, a modified LOS pitch controller with a look-ahead distance proved to be an accurate path follower by reducing the controller errors to acceptable levels. Using a model of the FLS, two additional sensor orientation constraints were discovered while conducting SIMULINK simulations in a hazardous environment. A Gaussian pop-up and a spine addition method were developed to over-come these issues while providing an optimal obstacle avoidance path. Comparing the two methods, the spline addition method proved to be the path planner of preference since it provided a robust avoidance path while optimizing the vehicle's information gathering sensors. Along with the look-ahead pitch controller, the spline addition path planner makes up a truly reactive "back-seat driver" that will improve REMUS's survivability in an unknown environment.

B. RECOMMENDATIONS

There are three areas in which further research can build of this thesis's effort to improve the autonomy of REMUS. First, this thesis was a first look at the use of splines for an obstacle avoidance path and further research is required to fully maximize the benefits behind its use. The addition of further knots will improve the smoothness of the curve and offer even greater flexibility by possibility replacing the entire obstacle avoidance path with one large multi-knot spline. The path could be altered by changing the locations of knots within the vicinity of any obstacle it encounters. Secondly, before implementing the "back-seat driver" into the vehicle for real-world testing, the spline method needs to be tested with current sonar imagery processing methods. Doing so will

verify that the processor is able to accurately classify and locate a obstacle while conducting a pop-up or in the middle of a Gaussian. Finally, since this thesis only involved motion in the vertical plane, future work should focus on defining optimal path planning constraints in the horizontal plane. By including horizontal plane motions, an optimal three-dimensional path can be generated, providing a more realistic solution in avoiding the obstacle.

APPENDIX. MATLAB CODE

VEHICLE DYNAMICS

```
% REMUS parameters developed by Chris Churan edited by Tyler Furukawa
clear
clc
z_g = 1.96e-2; x_b = 0; W = 299; buoy = 306;
I_z = 3.45; I_y = 3.45; I_x = 1.77e-1;
U = 1.5; to = 0; tf = 80;
m = 299/9.81; M_q = -6.87; M_qdot = -4.88;
M_w = 30.7; M_wdot = -1.93; M_d = -34.6;
Z_q = -9.67; Z_qdot = -1.93;
Z_w = -66.6; Z_wdot = -35.5; Z_d = -50.6;

% Dynamics ------------------------------------------------------------
% modified for [wr; q; theta]
M = [m-Z_wdot -Z_qdot 0; -M_wdot I_y-M_qdot 0; 0 0 1];
A_0 = [Z_w m*U+Z_q 0; M_w M_q -z_g*W; 0 1 0];
B_0 = [Z_d; M_d; 0];
A = inv(M)*A_0
B = inv(M)*B_0
C = [0 0 1]

%Check open loop poles
open_loop_poles = eig(A)

%Check Controlability
Control=[B,A*B,A^2*B];
Controllable=rank(Control)

%Check Observability
Observe=[C',A'*C',A'^2*C'];
Observability=rank(Observe)

p = [-.5+.866i, -.5-.866i, -1] %using butterworth pattern
K = place(A,B,p) %[-1.6807, 0.2935, 0.0889]
```

TRACKING

```
function command=tracking(x,z)

aim = 4; %aims 4 meters ahead

%%%%%Normal Altitude Command%%%%%%%%%
Z_mine=35;
min_clear=2;
X_mine = 60;
sigma=5; %sigma^2=25
```

```
%Gaussian Function
x_g1 = x; %x-position
z_g1 = Z_mine-min_clear*exp(-(X_mine-x_g1)^2/(2*sigma^2));% z-posit
x_g2 = x+0.1; %creates a delta x of 0.1 meter
z_g2 = Z_mine-min_clear*exp(-(X_mine - x_g2)^2/(2*sigma^2));

%Tangent
T2=[(x_g2-x_g1);(z_g2-z_g1)];

%Normal
N2=[-(z_g2-z_g1);(x_g2-x_g1)];

%Position
P2=[(0);(z-z_g1)];

%cross track error
CTE=P2'*N2/sqrt(N2'*N2);

command=-atan2(-CTE,aim);
```

TRACKING WITH SLOPE

```
function command=tracking_with_slope(x,z)

aim = 4; %aims 4 meters ahead

Z_mine=35;
min_clear=2;
X_mine = 60;
sigma=5; %sigma^2=25

%Gaussian Function
x_g1 = x; %x-position
z_g1 = Z_mine-min_clear*exp(-(X_mine-x_g1)^2/(2*sigma^2)); % z-posit
x_g2 = x+0.1; %creates a delta x of 0.1 meter for slope calculation
z_g2 = Z_mine - min_clear*exp(-(X_mine - x_g2)^2/(2*sigma^2));

%Tangent
T2=[(x_g2-x_g1);(z_g2-z_g1)];
%Normal
N2=[-(z_g2-z_g1);(x_g2-x_g1)];
%Position
P2=[(0);(z-z_g1)];
%cross track error
CTE=P2'*N2/sqrt(N2'*N2);

slope=atan2(-(z_g2-z_g1),(x_g2-x_g1));

command=slope-atan2(-CTE,aim);
```

TRACKING_LOOKAHEAD

```
function command=tracking_lookahead(x,z)

aim = 4; %aims 4 meters ahead

Z_mine=35;
min_clear=2;
X_mine = 60;
sigma=5; %sigma^2=25
lookahead = 4.58 %ideal lookahead = 4.58 m

%Gaussian Function
x_g1 = x+lookahead; %modified x-position
z_g1 = Z_mine-min_clear*exp(-(X_mine-x_g1)^2/(2*sigma^2)); %mod z-posit
x_g2 = x_g1+0.1; %creates a delta x of 0.1 meter
z_g2 = Z_mine - min_clear*exp(-(X_mine - x_g2)^2/(2*sigma^2));

%Tangent
T2=[(x_g2-x_g1);(z_g2-z_g1)];
%Normal
N2=[-(z_g2-z_g1);(x_g2-x_g1)];
%Position
P2=[(0);(z-z_g1)];
%cross track error
CTE=P2'*N2/sqrt(N2'*N2);

command=-atan2(-CTE,aim);
```

CONTROLLER ERRORS

```
%calculates and plots controller errors

x_sim=(1:1:120);
[num_points, columns] = size(x_pos);

E=zeros(num_points,1);
for k=1:1:num_points
    E(k,1)=sqrt(((ocean_depth-z_pos(k))-(Mine_altitude+min_clear*exp(-
(x_pos(k)-X_mine)^2/(2*sigma^2))))^2);
end
figure(1);clf
plot(x_pos,E), title('Vertical Error')
xlabel('Horizontal Position (m)')
ylabel('Vertical Error (m)')

%Calculates and plots sum of error
sum_error=zeros(num_points,1);
for l=2:1:num_points
    sum_error(l,1)=sum_error(l-1)+E(l-1);
end
figure(2);
plot(x_pos,sum_error), title('Total Vertical Error')
```

```
xlabel('Horizontal Position (m)')
ylabel('Total Vertical Error (m)')

max_error = max(E)
avg_error = sum(E)/num_points
total_error =sum(E)
```

OCEAN MODELS

```
Wall_Z = 5; %height of obstacle
Wall_X = 100;
Wall_Z2 = 4.5;
Wall_X2= 110;

%Floor 1
X_Floor1 = 0:Interval:Wall_X-Interval;
Z_Floor1 = zeros(1,length(X_Floor1));
%Floor 2
Z_Floor2 = 0:Interval:Wall_Z;
X_Floor2 = Wall_X*ones(1,length(Z_Floor2));
%Floor 3
X_Floor3 = Wall_X+Interval:Interval:2*Wall_X;
Z_Floor3 = zeros(1,length(X_Floor3));
%Floor 4
Z_Floor4 = 0:Interval:Wall_Z2;
X_Floor4 =Wall_X2*ones(1,length(Z_Floor4));

%plots the floor
subplot(2,1,1)
plot(X_Floor1,Z_Floor1,'g*'), hold on
plot(X_Floor2,Z_Floor2,'g*')
plot(X_Floor3,Z_Floor3,'g*')
plot(X_Floor4,Z_Floor4,'g*')
axis ([0 120 0 60])
```

SONAR MODEL

```
Wall_Z = 5; %height of obstacle
Wall_X = 60;
Interval = 0.1;

%Floor 1
X_Floor1 = 0:Interval:Wall_X-Interval;
Z_Floor1 = zeros(1,length(X_Floor1));
%Floor 2
Z_Floor2 = 0:Interval:Wall_Z;
X_Floor2 = Wall_X*ones(1,length(Z_Floor2));
%Floor 3
X_Floor3 = Wall_X+Interval:Interval:120;
Z_Floor3 = zeros(1,length(X_Floor3));
```

```
%plots the floor
subplot(2,1,1)
plot(X_Floor1,Z_Floor1,'g*'), hold on
plot(X_Floor2,Z_Floor2,'g*')
plot(X_Floor3,Z_Floor3,'g*')
axis ([0 120 0 60])
title('Sonar Model')

%plots ordered path
x_sim=(1:1:120);
for j=1:1:length(x_sim)
    plot(x_sim(j),Mine_altitude+min_clear*exp(-(x_sim(j)
X_mine)^2/(2*sigma^2)),'r*');hold on
end

X_Remus = 0; Z_Remus = 5; %Remus Position (Can take from SIMULINK)
plot(X_Remus, Z_Remus,'bo')
Theta = 0; %Remus Pitch in degrees(taken from SIMULINK model)

Max_Sonar_Range = 100; %in meters

%Calculates Slopes of Sonar Beam
sonar_angle =-22.5+Theta:-.5:-45+Theta; %interval determines # of beams
for a=1:length(sonar_angle)
    sonar_slope(a)=tand(sonar_angle(a));
    %plots sonar lines
    Beam_X=[X_Remus, X_Remus+Max_Sonar_Range*cosd(-sonar_angle(a))];
    Beam_Z=[Z_Remus, Z_Remus-Max_Sonar_Range*sind(-sonar_angle(a))];
    plot(Beam_X,Beam_Z,'k:')
end

%finds the highest beam that intercepts the wall
for b=1:length(sonar_angle)
    Wall_Intercept(b)=sonar_slope(b)*(Wall_X-X_Remus) + Z_Remus;
    if Wall_Intercept(b)<=Wall_Z & Wall_Intercept(b)>=0
        Beam_Intercept = b;
        Highest_Beam_Z = Wall_Intercept(b);
        break
    else Highest_Beam_Z=0;
    end
end

%find occlusion area
if Highest_Beam_Z>0
    Occlusion_X=[Wall_X, Wall_X, Wall_X, Wall_X+(Max_Sonar_Range-
(Wall_X-X_Remus))*cosd(-sonar_angle(Beam_Intercept))];
    Occlusion_Z=[0, Highest_Beam_Z, Highest_Beam_Z, Highest_Beam_Z-
(Max_Sonar_Range-(Wall_X-X_Remus))*sind(-sonar_angle(Beam_Intercept))];
else Occlusion_X = [0, 0]; Occlusion_Z=[0, 0];
end
subplot(2,1,2)
plot(X_Floor1,Z_Floor1,'g*'), hold on
plot(X_Floor2,Z_Floor2,'g*')
plot(X_Floor3,Z_Floor3,'g*')
plot(Occlusion_X,Occlusion_Z,'r--')
axis ([0 120 0 60])
```

```
title('Occlusion Area')
 %y=mx+b  for the sonar b = Z_remus and Global_X = x +X_Remus

%Find where bottom beam intercepts the bottom 0=mx+Z_remus or x = -
Z_remus/m
Lowest_Beam_X = -Z_Remus/sonar_slope(length(sonar_angle));
if sqrt(Z_Remus^2 +Lowest_Beam_X^2) < Max_Sonar_Range &
Lowest_Beam_X+X_Remus > X_Remus %check range & if x intercept < X_Remus
    Left_X = Lowest_Beam_X +X_Remus;
else Left_X = 0;
end

%if bottom beam doesn't intercept bottom, find its "Z" at wall
if Left_X<=0 & abs(Wall_X-X_Remus)<=Max_Sonar_Range
    %y = mx+b where x = (Wall_X-X_Remus), and b = Z_Remus
    Lowest_Beam_Z=sonar_slope(length(sonar_angle))*(Wall_X-X_Remus) +
Z_Remus;
    if Lowest_Beam_Z<=Wall_Z;
        Low_Wall_Intercept = Lowest_Beam_Z;
    else
        Low_Wall_Intercept = 0;
    end
end

%if X intercept is behind wall, find its "Z" at wall
if Left_X>Wall_X & abs(Wall_X-X_Remus)<=Max_Sonar_Range
    %y = mx+b where x = Wall_X, and b = Z_Remus
    Lowest_Beam_Z=sonar_slope(length(sonar_angle))*(Wall_X-X_Remus) +
Z_Remus;
    if Lowest_Beam_Z<=Wall_Z;
        Low_Wall_Intercept = Lowest_Beam_Z;
    else
        Low_Wall_Intercept = 0;
    end
end
```

SONAR_MOVING

```
clf
Wall_X = 100;  Wall_Z = 5; %height of obstacle
Wall_X2= 110;  Wall_Z2 = 4.5;

Interval = 0.1;
Max_Sonar_Range = 100; %in meters

Gaus2_offset = 35;
Gaus2_range = Wall_X-Gaus2_offset;
Orig_alt = 3;

%Floor 1
X_Floor1 = 0:Interval:Wall_X-Interval;
Z_Floor1 = zeros(1,length(X_Floor1));
%Floor 2
Z_Floor2 = 0:Interval:Wall_Z;
```

```
X_Floor2 = Wall_X*ones(1,length(Z_Floor2));
%Floor 3
X_Floor3 = Wall_X+Interval:Interval:2*Wall_X;
Z_Floor3 = zeros(1,length(X_Floor3));
%Floor 4
Z_Floor4 = 0:Interval:Wall_Z2;
X_Floor4 =Wall_X2*ones(1,length(Z_Floor4));

x_sim=(0:.2:2*Wall_X);

figure(1),clf;
for k=1:1:length(x_sim)
    if x_sim(k)<Gaus2_range
        %plot orig alt
        plot(x_sim(k),Orig_alt,'r*');hold on
    else if x_sim(k)<(Wall_X+Gaus2_offset)
        %plot gaussian
        plot(x_sim(k),Orig_alt+(min_clear+Wall_Z-Orig_alt)*exp(-
(x_sim(k)-Wall_X)^2/(2*sigma2^2)),'r*')
        else
            plot(x_sim(k),Orig_alt,'r*')
        end
    end
end

%plot(X_mine,Mine_altitude,'ro');
plot(x_pos,(ocean_depth-z_pos));
axis([0 2*Wall_X 0 15]), title('Ordered versus Actual')
xlabel('X - meters'), ylabel('Altitude - meters')

mov = avifile('sonar_Movie.avi','Compression','Cinepak','FPS',1);

for c=3:2:length(x_pos);

    set(gcf,'doublebuffer','on');

    figure(c)
    %plots the floor
    subplot(2,1,1)
    plot(X_Floor1,Z_Floor1,'g*'), hold on
    plot(X_Floor2,Z_Floor2,'g*')
    plot(X_Floor3,Z_Floor3,'g*')
    plot(X_Floor4,Z_Floor4,'g*')
    %plots actual course
    %plot(x_pos,ocean_depth-z_pos,'c*')
    axis ([30 130 0 15])
    title('Sonar Model')

    X_Remus = x_pos(c);
    Z_Remus = ocean_depth-z_pos(c); %Remus Posit (Taken from SIMULINK)
    plot(X_Remus, Z_Remus,'bo')
    Theta = Remus_theta(c); %Remus Pitch in degrees(taken from SIMULINK

    %Calculates Slopes of Sonar Beam
```

```
    sonar_angle =(-2.5+Theta):-Interval:(-25+Theta); %determines # of
beams
    for a=1:length(sonar_angle)
        sonar_slope(a)=tand(sonar_angle(a));
        %plots sonar lines
        Beam_X=[X_Remus, X_Remus+Max_Sonar_Range*cosd(-
sonar_angle(a))];
        Beam_Z=[Z_Remus, Z_Remus+Max_Sonar_Range*sind(sonar_angle(a))];
        plot(Beam_X,Beam_Z,'k:')
    end

    %finds the highest beam that intercepts the wall
    for b=1:length(sonar_angle)
        Wall_Intercept(b)=sonar_slope(b)*(Wall_X-X_Remus) + Z_Remus;
        if Wall_Intercept(b)<=Wall_Z & Wall_Intercept(b)>=0
            Beam_Intercept = b;
            Highest_Beam_Z = Wall_Intercept(b);
            break
        else
            Beam_Intercept=-1;
            Highest_Beam_Z=-1;
        end
    end

    %find if "sees" second obstacle
    %if intercepts 1st wall
    Sonar_Wall_Z2=-1;%initialize each time
    Sonar_Wall_X2=-1;%intiialize each time
    if Beam_Intercept>1;
        Wall2_Intercept = sonar_slope(Beam_Intercept)*(Wall_X2-X_Remus)
+Z_Remus;
        if Wall2_Intercept <= Wall_Z2
            Sonar_Wall_Z2=Wall2_Intercept:Interval:Wall_Z2;
            Sonar_Wall_X2=Wall_X2*ones(1,length(Sonar_Wall_Z2));
        end
    elseif Beam_Intercept<1;
        %check to see if intercept 2nd wall
        for d=length(sonar_angle):-1:1 %cycle from lowest beam
            Wall_Intercept2=sonar_slope(d)*(Wall_X2-X_Remus) + Z_Remus;
            if Wall_Intercept2<=Wall_Z2 & Wall_Intercept2>=0 &
X_Remus<Wall_X2
                %Beam_Intercept2 = d;
                %Lowest_Beam_Z2 = Wall_Intercept2(d);
                Sonar_Wall_Z2=Wall_Intercept2:Interval:Wall_Z2;
                Sonar_Wall_X2=Wall_X2*ones(1,length(Sonar_Wall_Z2));
                break
            else
                Sonar_Wall_X2=-1;
                Sonar_Wall_Z2=-1;
            end
        end
    end

    if Highest_Beam_Z>0 & X_Remus<Wall_X
        %find occlusion area
```

```
        Occlusion_X=[Wall_X, Wall_X+(Max_Sonar_Range-(Wall_X-
X_Remus))*cosd(-sonar_angle(Beam_Intercept))];
        Occlusion_Z=[Highest_Beam_Z, Highest_Beam_Z-(Max_Sonar_Range-
(Wall_X-X_Remus))*sind(-sonar_angle(Beam_Intercept))];
    else Occlusion_X = [0, 0]; Occlusion_Z=[0, 0];
    end
    subplot(2,1,2)
        plot(Occlusion_X,Occlusion_Z,'r--'), hold on
    axis ([30 130 0 15])
    title('Sonar Image w/Occlusion Area')

    %y=mx+b  for the sonar b = Z_remus and Global_X = x +X_Remus

    %Find where top beam intercepts the bottom 0=mx+Z_remus or x = -
Z_remus/m
    Top_Beam_X=-Z_Remus/sonar_slope(1);
    if sqrt(Z_Remus^2+Top_Beam_X^2)<Max_Sonar_Range &
Top_Beam_X+X_Remus<Wall_X & Top_Beam_X+X_Remus>X_Remus%check range and
if x intercept < Wall_X
        Right_X=Top_Beam_X+X_Remus;
    else Right_X=-1;
    end

    %Find where bottom beam intercepts the bottom 0=mx+Z_remus or x = -
Z_remus/m
    Lowest_Beam_X = -Z_Remus/sonar_slope(length(sonar_angle));
    if sqrt(Z_Remus^2+Lowest_Beam_X^2)<Max_Sonar_Range &
Lowest_Beam_X+X_Remus<Wall_X & Lowest_Beam_X+X_Remus>X_Remus%check
range and if x intercept<Wall_X
        Left_X = Lowest_Beam_X +X_Remus;
        if Right_X>0
            X_Sonar_floor1=Left_X:Interval:Right_X;
        else
            X_Sonar_floor1=Left_X:Interval:Wall_X-Interval;
        end
        Z_Sonar_floor1=zeros(1,length(X_Sonar_floor1));
        Z_Sonar_floor2=0:Interval:Highest_Beam_Z;
        X_Sonar_floor2=Wall_X*ones(1,length(Z_Sonar_floor2));
    elseif abs(Wall_X-X_Remus)<=Max_Sonar_Range
        %if bottom beam doesn't intercept bottom or is behind wall,
find its "Z" at wall
        Lowest_Beam_Z=sonar_slope(length(sonar_angle))*(Wall_X-X_Remus)
+ Z_Remus;
        if Lowest_Beam_Z<=Wall_Z;
           Low_Wall_Intercept = Lowest_Beam_Z;
           Z_Sonar_floor1=Low_Wall_Intercept:Interval:Highest_Beam_Z;
           X_Sonar_floor1=Wall_X*ones(1,length(Z_Sonar_floor1));
           Z_Sonar_floor2=-1;
           X_Sonar_floor2=-1;
        else Z_Sonar_floor1 = -1; X_Sonar_floor1=-1; Z_Sonar_floor2=-1;
           X_Sonar_floor2=-1;
        end
    end
    %subplot(3,1,3)
    plot(X_Sonar_floor1,Z_Sonar_floor1,'g*')
    plot(X_Sonar_floor2,Z_Sonar_floor2,'g*')
```

```
    plot(Sonar_Wall_X2,Sonar_Wall_Z2,'g*')
    axis ([30 130 0 15])

    F = getframe(gcf);
    mov = addframe(mov,F);
    pause(.1);
end

mov = close(mov);
```

TRACKING_POPUP

```
function command=tracking_popup(x,z)

aim = 4; %aims 4 meters ahead

X_mine = 100;
Z_mine=35; %Altitude of 5 (when ocean_depth=40)
org_depth = 37; %altitude of 3 (when ocean_depth=40)
%for popup Gaussian
ho=5;
sigma1=5;
%for Obstacle Avoidance Gaussian
min_clear=3;
sigma2=12;
clear = (min_clear+org_depth-Z_mine);
gaus1_offset=80;
gaus1_range=X_mine-gaus1_offset;

gaus2_offset = 40; %do gaussian 40m before and after obstacle
gaus2_range = X_mine-gaus2_offset;

lookahead = 4.5; %"look" ahead 4.5
delta_x = .1; %for slope calculation

if x<(gaus1_range-lookahead)
    %do original altitude
    x_g1 = x+lookahead; %modified x-position
    z_g1 = org_depth;
    x_g2 = x_g1+delta_x; %creates a delta x of 0.1 meter
    z_g2 = org_depth;
elseif x<(gaus2_range-lookahead)
    %do 1st gaus
    x_g1 = x+lookahead; %modified x-position
    z_g1 = org_depth - ho*exp(-(gaus1_range+25-x_g1)^2/(2*sigma1^2));
    x_g2 = x_g1+delta_x; %creates a delta x of 0.1 meter
    z_g2 = org_depth - ho*exp(-(gaus1_range+25-x_g2)^2/(2*sigma1^2));
elseif x<(X_mine+gaus2_offset)
    %do gaussian
    x_g1 = x+lookahead; %modified x-position
    z_g1 = org_depth - clear*exp(-(X_mine-x_g1)^2/(2*sigma2^2));
    x_g2 = x_g1+delta_x; %creates a delta x of 0.1 meter
    z_g2 = org_depth - clear*exp(-(X_mine - x_g2)^2/(2*sigma2^2));
```

```
else
    x_g1 = x+lookahead; %modified x-position
    z_g1 = org_depth;
    x_g2 = x_g1+delta_x; %creates a delta x of 0.1 meter
    z_g2 = org_depth;
end


%Tangent
T2=[(x_g2-x_g1);(z_g2-z_g1)];
%Normal
N2=[-(z_g2-z_g1);(x_g2-x_g1)];
%Position
P2=[(0);(z-z_g1)];
%cross track error
CTE=P2'*N2/sqrt(N2'*N2);

command=-atan2(-CTE,aim);
```

TVD CALCULATION

```
Original_Depth=37; %altitude =3m m
Deviation=zeros(1,length(x_pos));
Area=Deviation;
for n=2:1:length(x_pos);
    Deviation(n)=(Original_Depth - (z_pos(n)+z_pos(n-1))/2); %avg
deviation between 2 pts
    Area(n)=Deviation(n)*(x_pos(n)-x_pos(n-1)); %tapezoid rule
end

Max_Deviation=max(Deviation)

TVD=sum(Area)
```

TRACKING SPLINE

```
function out=tracking_spline2(x,z)

aim = 4; %aims 4 meters ahead
X_mine = 100;
Z_mine=35; %Altitude of 5 (when ocean_depth=40)
org_depth = 37; %altitude of 3 (when ocean_depth=40)
%for modified popup
ho=.75;
sigma1=2;
%for obstacle avoidance Gaussian
min_clear=2;
sigma2=5;
clearance = org_depth-(Z_mine-min_clear);
gaus1_offset=80;
gaus1_range=X_mine-gaus1_offset;
```

```
gaus2_offset = 20; %do gaussian 20m before and after obstacle
gaus2_range = X_mine-gaus2_offset;

lookahead = 4.58; %"look" ahead 4.58
delta_x = .1; %for slope calculation

%second obstacle
Wall_Z2 = 35.5;
Wall_X2= 110;
clear2=org_depth-(Wall_Z2-min_clear);
Orig_alt = 3;

%for spline
Interval=0.1;
spline_offset=.1;
spline_range=X_mine+spline_offset;
X_slope1=spline_range+lookahead;
spline_X_int=Wall_X2-X_slope1;

X_spline1=X_slope1;
X_spline2=X_spline1+(spline_X_int)/3;
X_spline3=X_spline2+(spline_X_int)/3;
X_spline4=Wall_X2;
X_spline5=Wall_X2+10;
X_spline6=X_spline5+1;
X_spline7=X_spline6+1;
X_spline8=2*X_mine;
X_spline=[X_spline1 X_spline2 X_spline3 X_spline4 X_spline5 X_spline6
X_spline7 X_spline8];

Z_spline1=clearance*exp(-(X_mine-X_slope1)^2/(2*sigma2^2))*(100-
X_slope1)/-(sigma2^2); %slope of spline
Z_spline2=org_depth-clearance*exp(-(X_mine-
X_slope1)^2/(2*sigma2^2));%height at gaussian
Z_spline3=Z_spline2-(Z_spline2-(Wall_Z2-min_clear))/3;
Z_spline4=Z_spline3-(Z_spline2-(Wall_Z2-min_clear))/3;
Z_spline5=Wall_Z2-min_clear;
Z_spline6=org_depth;
Z_spline7=Z_spline6;
Z_spline8=Z_spline6;
Z_spline9=Z_spline6;
Z_spline10=0.0;
Z_spline=[Z_spline1 Z_spline2 Z_spline3 Z_spline4 Z_spline5 Z_spline6
Z_spline7 Z_spline8 Z_spline9 Z_spline10];

%xx=X_slope1:Interval:X_spline8;
%yy=spline(X_spline,Z_spline,xx);

if x<(gaus1_range-lookahead)
    %do original altitude
    x_g1 = x+lookahead; %modified x-position
    z_g1 = org_depth;
    x_g2 = x_g1+delta_x; %creates a delta x of 0.1 meter
    z_g2 = org_depth;
elseif x<(gaus2_range-lookahead)
    %do 1st gaus
```

```
    x_g1 = x+lookahead; %modified x-position
    z_g1 = org_depth - ho*exp(-(gaus1_range+25-x_g1)^2/(2*sigma1^2));
    x_g2 = x_g1+delta_x; %creates a delta x of 0.1 meter
    z_g2 = org_depth - ho*exp(-(gaus1_range+25-x_g2)^2/(2*sigma1^2));
elseif x<(spline_range)
    %do 2nd gaussian
    x_g1 = x+lookahead; %modified x-position
    z_g1 = org_depth - clearance*exp(-(X_mine-x_g1)^2/(2*sigma2^2));
    x_g2 = x_g1+delta_x; %creates a delta x of 0.1 meter
    z_g2 = org_depth - clearance*exp(-(X_mine - x_g2)^2/(2*sigma2^2));
else
    x_g1 = x+lookahead; %modified x-position
    z_g1 = spline(X_spline,Z_spline,x_g1);
    x_g2 = x_g1+delta_x; %creates a delta x of 0.1 meter
    z_g2 = spline(X_spline,Z_spline,x_g2);
end


%Tangent
T2=[(x_g2-x_g1);(z_g2-z_g1)];
%Normal
N2=[-(z_g2-z_g1);(x_g2-x_g1)];
%Position
P2=[(0);(z-z_g1)];
%cross track error
error=P2'*N2/sqrt(N2'*N2);

slope=atan2((z_g2-z_g1),(x_g2-x_g1));

command= -atan2(-error,aim);
x_g1=x_g1;
z_g1=z_g1;
out=[command,x_g1,z_g1];
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Ackleson, Steven, "Office of Naval Research Investments in Unmanned Underwater Vehicles (UUV)," Office of Naval Research, [http://www.onr.navy.mil/about/conferences/rd_partner/2005/docs/past/2002/2002_ackleson_investments_uuv.pdf], June 2006.

[2] Coleman, Jack," Undersea drones pull duty in Iraq hunting mines," *Cape Code Times*, 2 April 2003.

[3] Hydroid Inc., Homepage, [www.hydoridinc.com], June 2006.

[4] Blue View Technologies Homepage, [www.blueviewtech.com], June 2006.

[5] Healey, A. J., "Obstacle Avoidance While Bottom Following for the REMUS Autonomous Underwater Vehicle," Proceedings of the IFAC-IAV 2004 Conference, Lisbon, Portugal, July 5-7, 2004. [http://web.nps.navy.mil/~me/healey/papers/IAV2004.pdf], June 2006.

[6] Churan, C., *Obstacle Avoidance Control for the REMUS Autonomous Underwater Vehicle*, MSME Thesis, Naval Postgraduate School, Monterey, California, September 2003.

[7] Heminger, Dan, *Vertical Plane Obstacle Avoidance and Control of the REMUS Autonomous Underwater Vehicle Using Forward Looking Sonar*, MSME Thesis, Naval Postgraduate School, Monterey, California, June 2005.

[8] Healey, A. J., "Dynamics and Control of Mobile Robotic Vehicles (MA-4823)," Class Notes, Naval Postgraduate School, Monterey, California, Winter 2001.

[9] Prestero, Timothy, *Verification of a Six-Degree of Freedom Simulation Model for the REMUS Autonomous Underwater Vehicle*, M.S. Thesis, Massachusetts Institute of Technology, September 2001.

[10] Ogata, Katsuhiko, *Modern Control Engineering,* Fourth Edition, Prentice Hall, 2001.

[11] Mathworld Homepage, [www.mathworld.com], June 2006.

[12] Horner, D. P., Healey, A. J. and Kragelund, S. P., "AUV Experiments in Obstacle Avoidance," Proceedings of the OCEANS 2005 MTS/IEEE Conference, Washington D.C., September 18-23, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Distinguished Professor Anthony J. Healey
   Department of Mechanical and Astronautical Engineering
   Naval Post Graduate School
   Monterey, California

4. Research Associate Douglas P. Horner
   Center for AUV Research
   Naval Post Graduate School
   Monterey, California

5. Research Associate Professor Oleg Yakimenko
   Department of Mechanical and Astronautical Engineering
   Naval Post Graduate School
   Monterey, California

6. Dr. Tom Swean
   Office of Naval Research
   Arlington, Virginia

7. Chris Von Alt
   Woods Hole Oceanographic Institute
   Woods Hole, Massachusetts

8. Dr. Kerry Commander
   Office of Naval Research
   Arlington, Virginia

9. Ken Kruger
   U.T. Applied Research Lab
   Austin, Texas