Optimal Path Planning and High Level Control of an Autonomous Gliding Underwater Vehicle

by

Anna M Galea

B.A.Sc., Engineering Science, Biomedical Engineering. (1997)

University of Toronto

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

September 1999

Signature of Author .... [illegible]
Department of Electrical Engineering and Computer Science
July 18, 1999

Certified by .... [illegible]
James G. Bellingham
Lecturer, Department of Ocean Engineering
Thesis Supervisor

Read by .... [illegible]
Knut Streitlien
~~Research Engineer~~, MIT Sea Grant AUV Lab

Accepted by .... [illegible]
Arthur C. Smith
Chair, Committee on Graduate Students

OPTIMAL PATH PLANNING AND HIGH LEVEL CONTROL OF AN AUTONOMOUS GLIDING UNDERWATER VEHICLE

by

Anna M Galea

Submitted to the Department of Electrical Engineering and Computer Science
on July 23, 1999 in partial fulfillment of the requirements for the Degree of
Master of Science in Electrical Engineering and Computer Science

ABSTRACT

Recent oceanographic interests have focussed on the littoral ocean, where regions of shallow water and high current are prevalent. Long the domain of remotely operated vehicles and, more recently, autonomous propeller-driven vehicles, glider vehicles are being designed and tested for these conditions. These vehicles face special problems in these environments because of their slow water speed and their depth-dependent energy function which requires them to use more energy to move the same horizontal distance through shallow water than in deeper waters. Missions run in shallow waters and with high-speed, time-variable currents thus face propulsion energy consumption difficulties as well as time efficiency concerns.

A solution to the optimal path for a typical glider vehicle was sought in a simulated environment without vehicle dynamics considerations, where the minimization criteria were time and energy required to travel between two waypoints. Dynamic programming, analytic methods, and numerical programming methods were applied to the problem, with numerical programming generating the most general results and the greatest insight into the minimization problem. Under the assumption of constant water speed, time-optimal paths were insensitive to depth variations, and were instead a compromise between a short path and one that accommodated high currents. For energy-minimization runs, the optimal paths deflected towards deeper regions, and were much less sensitive to current than to depth.

To run missions with these strategies, a layered control architecture was implemented for these vehicles. Special behaviors were written to accommodate glider operations, and a dynamic controller written which incorporated improvements suggested by working with its Odyssey model. The specific control strategy can be transported to any glider vehicle, regardless of the hardware used to affect dynamic changes.

Thesis Supervisor: James G. Bellingham
Title: Lecturer, Department of Ocean Engineering

Table of contents

List of Figures

List of Tables

Glossary of terms and acronyms

Definitions of Terms:

Thermocline: The natural stratification of water by temperature, with cooler water below.

Ping: A high pitched fixed frequency sound of narrow bandwidth used as an identifiable long-range signal underwater.

Acronyms Used:

AOSN	– Autonomous Oceanographic Sampling Network
AUV	– Autonomous Underwater Vehicle
CTD	– Conductivity, Temperature, and Depth sensor
GPS	– Global Positioning System
WRC	– Webb Research Corporation

Chapter 1

Introduction

One of the goals of implementing Autonomous Oceanographic Sampling Networks (AOSNs)[1,2] is to integrate the use of various underwater vehicles and devices in order to maintain an oceanographic presence in all parts of the ocean[3]. Traditionally, shallow waters were considered the domain of ships and ROVs[4] while deeper waters were seen as the rightful place of propeller-driven AUVs, buoyancy-driven drifters[5] and glider vehicles[6]. Although great strides have been made in populating the deep ocean with these latter devices[α] the recent push towards more detailed exploration of littoral waters[7] has led to the use of AUVs in these regions[8,9], and the design of a new glider for use in shallow waters.

Glider vehicles are a derivative of drifters that have been used for years to sample the ocean column. Drifters control their buoyancy by changing their volume (by filling and emptying a collapsible outer bladder from an internal reservoir) or mass (by intaking and expelling an amount of ambient water). Their horizontal motion is entirely dependent on the prevailing currents. Gliders are essentially drifters with the ability to control the pitch of a fin surface as they float and sink[6,10], thus converting some of the vertical motion into horizontal speed to effectively glide through the water and control their horizontal coverage. Since the buoyancy changes affected are slight, these vehicles tend to glide slowly, with a typical horizontal speed of about 37 cm/s[6]. Although they aren't as fast as propeller driven vehicles, their operational time is much longer (on the order of days instead of hours) and so they have their own niche in an AOSN. For example, gliders can be used as sentinel vehicles, constantly monitoring a region until an event of interest is recorded, at which time they might signal a faster AUV to explore the

[α] One of the goals outlined in Stommel's "1000 diving robots"[3] was achieved in January 1998 when Webb Research Corporation manufactured its 1000th float.

region in more detail and possibly track the phenomenon. Their inherent slow speed, however, raises questions about glider performance in regions of high currents, since their ability to conduct proper survey missions is integral to their contribution to oceanographic research.

Another potential problem that gliders face concerns their propulsion power in shallow waters. Due to the ambient pressure difference, the pump used to affect the buoyancy changes requires more power at greater depths[11]. Practical considerations, however, dictate that the multiple dives necessitated by shallower waters require more power for the same horizontal travel as fewer dives in deeper waters[11,12].

In order to study the feasibility of using glider vehicles in shallow water, high-current situations, a simulation was constructed to test a glider computer model in a dynamic-free environment. The results of these simulations were to provide strategies by which real gliders can be used most effectively in adverse physical conditions. In order to run missions with these strategies, a flexible control architecture had to be developed for use on the vehicle. A layered controller was chosen for its documented success on small AUVs. This will be tested in sea trials in the summer of 1999.

Chapter 2

Glider Vehicle Description

2.1. History

Throughout the history of oceanographic research, floats have been used to study the surface currents of the oceans and other bodies of water. Due mainly to increased pressure with depth, water in large bodies is denser further down, and so an object that sinks on the surface might be able to 'float' in midwater. This phenomenon can be exploited such that floats can be built which are ballasted for neutral buoyancy at a particular depth. This works since the floats are less compressible than the surrounding water. Floats such as these have been used with much success to shed light on internal currents. The first of these in-water floats were equipped with a drop weight which, at the end of a predetermined time, would be released so that the float surfaced and its end position could be recorded. These early floats could thus only provide information at one end point.

By 1985 buoyancy-controlling mechanisms started to come into use. A float equipped with means to make itself positively or negatively buoyant at will could surface multiple times before the end of its mission, providing many more data points than its more primitive ancestors[13]. Today, there are many hundreds such floats in active use around the world, many equipped with instruments such as CTD (conductivity, temperature, and depth) sensors and other devices so that they provide much more information about the internal workings of the oceans than simply current.

A vessel that can control its buoyancy can do so in either a fixed-mass or fixed-volume fashion. A fixed-mass vessel typically has a flexible external bladder (generally protected from the environment by being housed inside the vehicle's wet volume). This bladder is connected to an internal reservoir in which is stored a quantity of fluid. For positive buoyancy, the external bladder is filled, thus increasing the displaced volume, but not the vehicle mass [figure 2.1a]. Effectively, the vehicle becomes less dense, thereby becoming more buoyant. If the vehicle is properly weighted to be neutrally buoyant when the bladder is half-filled, then when it is filled the vehicle floats, and when it is emptied the vehicle sinks. The fluid can be air pumped from inside the dry volume to the bladder, however at high ocean pressures an air/water interface is undesirable, and so generally a non-compressible fluid is used. Conceptually, you can also have a vehicle which changes its size by extending a piston into its environment instead of inflating an external bladder. This works under the same principles outlined above.

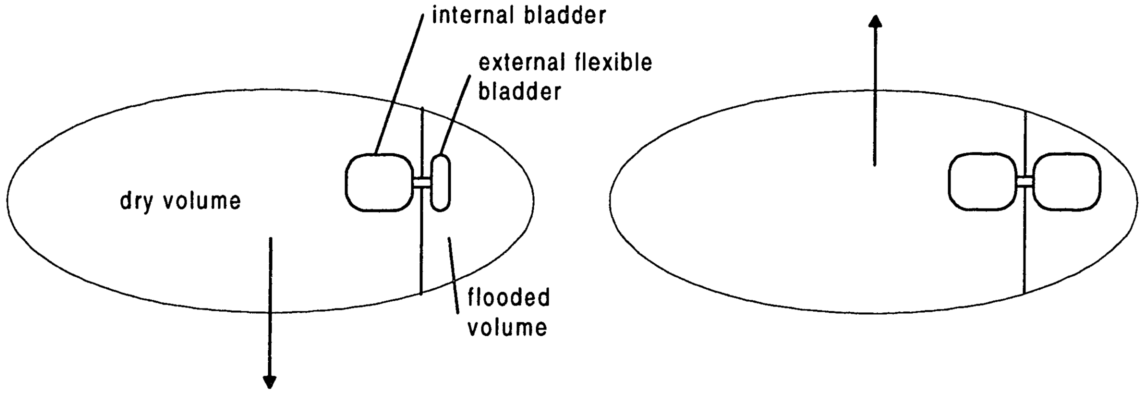


Figure 2.1a. A fixed-mass buoyancy controlled device.

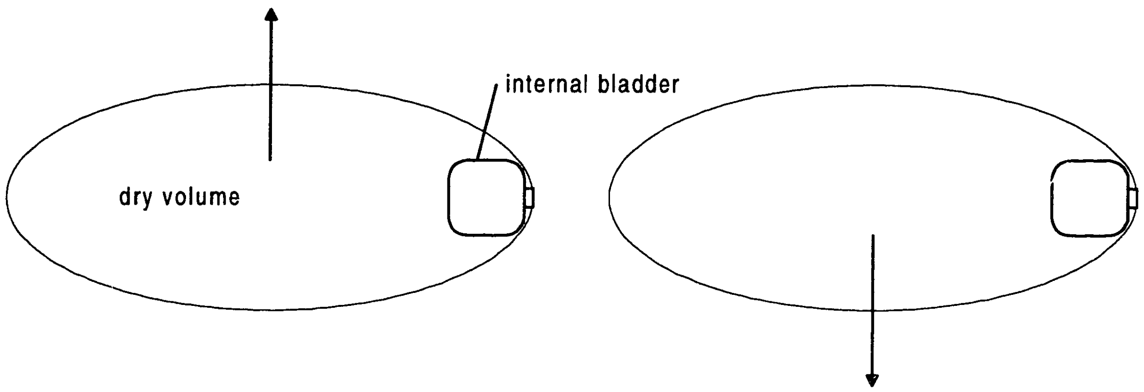


Figure 2.1b. A fixed-volume buoyancy controlled device.

Figure 2.1b illustrates the buoyancy principle of a fixed-volume vessel. To achieve negative buoyancy, the vessel takes in an amount of ambient water, thus increasing its mass without changing its volume. For adequate buoyancy control, the vehicle should be neutrally buoyant midway through the column of water it is intended to study when the reservoir is half-full. For either the fixed-mass or the fixed-volume scheme, only a small amount of fluid needs to be moved. A well-ballasted vessel will float or sink with a change in its displaced mass or volume of less than 1%.

2.2. Gliding

Gliding can be defined as the acquiring of horizontal motion solely from potential energy. The reader may be familiar with air gliders, which convert their potential (gravitational) energy, which acts purely downwards, into forward motion by means of airfoils which provide lift and serve to counteract drag. If the conversion from vertical to horizontal motion were perfect, the glider would never descend, continuing forever in its horizontal motion.

The story of underwater gliders is different in that underwater gliders have the advantage of being immersed in a much more dense fluid than air, and thus their lift need not hold up the entire weight of the vehicle. Besides the denser environment, the speeds at which these vehicles glide are also very different, and so underwater gliders experience very low Reynolds numbers (on the order of 10^5, compared to Reynolds numbers of at least 10^8 for their air counterparts). The analysis of air gliders, therefore, does not port over to underwater gliders. Still, the conceptual analogy serves well to introduce their cousins in the underwater realm.

2.3. Glider

A glider is essentially a float with pitch control of a fin surface with which it can convert some of the vertical velocity from floating or sinking into forward velocity. This leaves gliders with a very distinctive vertical zigzag profile through the water column, as shown in figure 2.2. Today, there are various gliders being built and used by different groups throughout the world[14], but all employ the same basic concepts to glide through the water column.

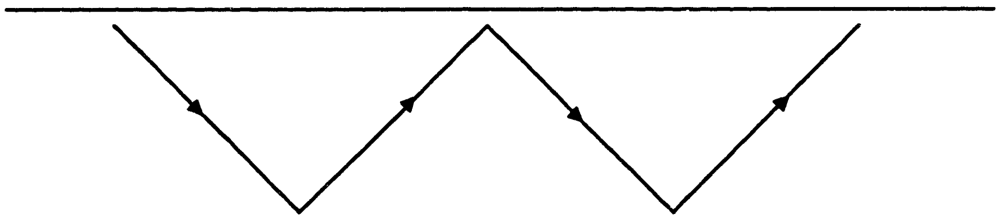


Figure 2.2. Characteristic glider motion pattern

Webb Research Corporation (WRC) has in the last few years delved into glider research and has designed and built a *Slocum*[10] glider which obtains its propulsion from a revolutionary thermal engine[5,15]. This type of engine is well suited for use on a glider, since it exploits passage through the thermocline for recharging, which is achieved naturally by the up-down motion of a glider. This type of engine will only provide power, however, in very specific ocean states with a suitable thermocline, and so a battery-powered glider was also designed for use where the thermal powered glider cannot function. It is this battery-powered glider that was used as the model for the work outlined in this thesis. The limited on-board energy calls attention to energy minimization, which combined with other factors to make this a particularly interesting test-case, and one which is a superset of the issues faced by other vehicles, both gliding and propeller driven.

2.4. The Slocum Electric Glider

Figure 2.3. WRC battery powered *Slocum* glider

Figure 2.3 shows a WRC battery-powered glider in one of its possible wing/tail configurations. The glider is 1.8m long, with a 1.2m long central cylindrical section of 21cm outer diameter, and elliptical nose and tail cones. Overall it displaces 51.8kg of water. Only the tail cone is a flooded volume, and this houses the sensor units, as well as an air bladder for surface buoyancy (the use of which will be discussed later). The nose cone houses a rolling diaphragm pump for ambient water, which is used for buoyancy changes in a fixed-mass context. A rolling diaphragm pump is used since only the rubber diaphragm is in contact with the water, avoiding the damage caused by corrosion of metal parts.

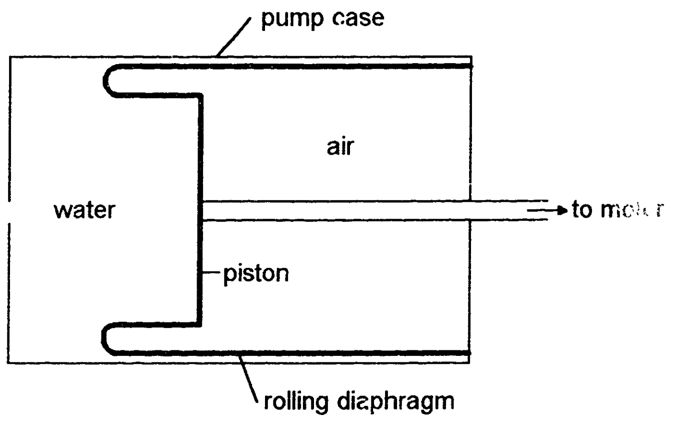


Figure 2.4. Schematic of a rolling diaphragm pump. The ambient water is in contact only with the rubber diaphragm.

Figure 2 5 is a schematic of the glider used in the sea trials for this work. In this glider, the boom tail fin contains GPS, Argos, and RF modem antennas. When the vehicle tries to obtain a GPS fix or is in a communications mode, it surfaces using the water pump, then inflates the air bladder so that the tail is held slightly higher than the nose. The tail fin with its antennas is placed out on the boom to further elevate it from the water, since water, especially salt water, absorbs the transmitted and received signals.

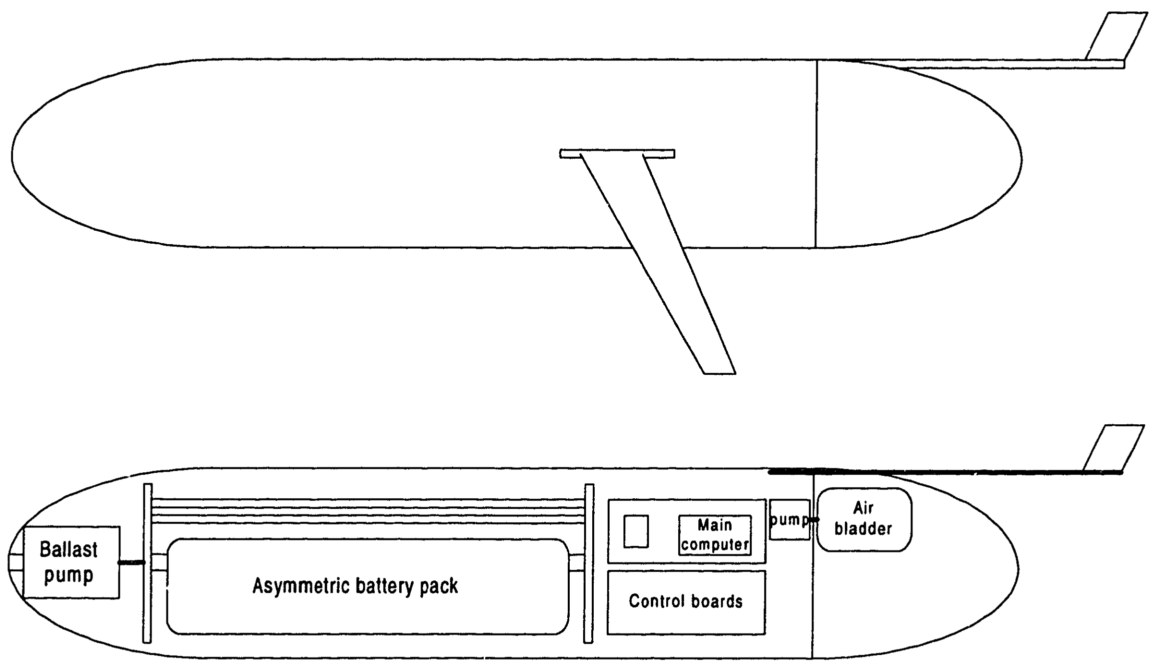


Figure 2.5a-b. A brief schematic of the glider used for this work. Sensors such as a CTD unit and a fluorometer are housed in the tail cone, while an attitude sensor is housed in the nose.

Typical payload for these vehicles includes a fluorometer, which is used to study phytoplankton content based on the cells' fluorescence, a CTD unit, which measures the conductivity, temperature, and pressure of the ambient water (which can be used to obtain the depth information, hence the 'D' in the acronym) and an altimeter, which sends a high pitched 'ping' towards the bottom at controllable intervals, and calculates the vehicle's distance from the bottom based on the echo time. These sensors all require water presence and are equipped with their own pressure housings, and so are housed in the tail cone. A 10MHz pinger used for vehicle location and depth telemetry is also housed in the tail, along with a drop weight, which is the last line of safety for the vehicle: if all else fails (ie the computer stops relaying commands and the main batteries run too low), then a burn wire is activated, the drop weight is ditched and the vehicle floats for retrieval. Internal sensors include an attitude sensor, housed in the nose cone, and various sensors which watch over the operations of the actuators, monitor internal states such as internal pressure and battery voltage, and ensure that communication is maintained between the computer and the hardware. The wings and tails on all WRC gliders are fixed, and thus pitch control of the wings and roll control of the vehicle requires pitching and rolling of the entire vehicle. This can be achieved by motion of the eccentric battery pack axially or rotationally. This will be discussed in more detail in the following section. Position of the battery pack and other devices can be seen in figure 2.5b.

The computer on these vehicles is a CF1 Persistor with an MC68CK338 processor and a incorporated 1MB solid state memory module. The 256KB of battery backed static RAM is augmented by 48MB of solid state memory (a Persistor Compact Card brand flashcard) for storage. Programs are written in C, compiled on a PC and run in PicoDOS on the Persistor.

Battery power on board, though variable with different vehicle configurations, consists of 100 alkaline D-cells and approximately 90 C-cells. These are wired in packs of 10, providing a nominal supply voltage of 15V. With careful consideration for which sensors and actuators can be powered down for certain intervals, this is expected to power the glider for 40-day non-stop missions.

2.5. Static Stability

The underwater glider, unlike its air counterpart, is inherently very stable when ballasted properly. Its axisymmetric shape yields a center of buoyancy that is at the center line, while the large weight of the eccentric battery pack ensures that the center of gravity is below this. These two points line up naturally when the vehicle is in water at zero velocity, and no matter how the vehicle is perturbed from its equilibrium position, it will always return to it by pure buoyant and gravitational forces. The Slocum electric glider has fixed wings and so uses the change in buoyancy both to sink and float, but also to obtain its glide angle. Figure 2.6b shows the glider configuration for when the water reservoir is half-full and the vehicle is neutrally buoyant. If the water is pushed out, as in figure 2.6a, then the center of gravity shifts slightly forwards since the piston is extended, while the center of buoyancy moves forward further since the vehicle displaces more water in its nose. The vehicle then pitches upwards as the buoyancy and gravitational vectors try to align. It also starts to float since the weight of the water it now displaces is greater than its own weight. The buoyant and gravitational forces for the Slocum electric glider are equivalent to approximately 471N each, with the gravitational force being larger by the small amount of water in the reservoir. This amount is only about 0.1 N, yet it suffices to throw the neutral buoyancy off and shift the relative positions of the two vectors, pitching the vehicle to about 25° nose-up and causing it to start floating. The amount of ballast pumped and its position within the vehicle determines the angle at which the vehicle will glide without other compensating mechanisms. In order to minimize the load on the finite amount of battery power it is undesirable to use extrenuous mechanisms, and also a minimal required ballast changed is desired. This latter condition fixes the ballast pump at the extreme nose or tail of the vehicle, and a desired glide angle (which will be discussed in section 2.8) fixes the amount pumped. In the Slocum electrical glider, a glide angle of approximately 26° is considered optimal, and so with a body of 1.8 meters in length and 48kg displacement requires a change in volume of approximately 90 cubic centimeters. The amount of ballast pumped, with its corresponding change in volume, also fixes the speed at which the vehicle glides, in this case to 41 cm/s in-line, or just under 37 cm/s horizontally.

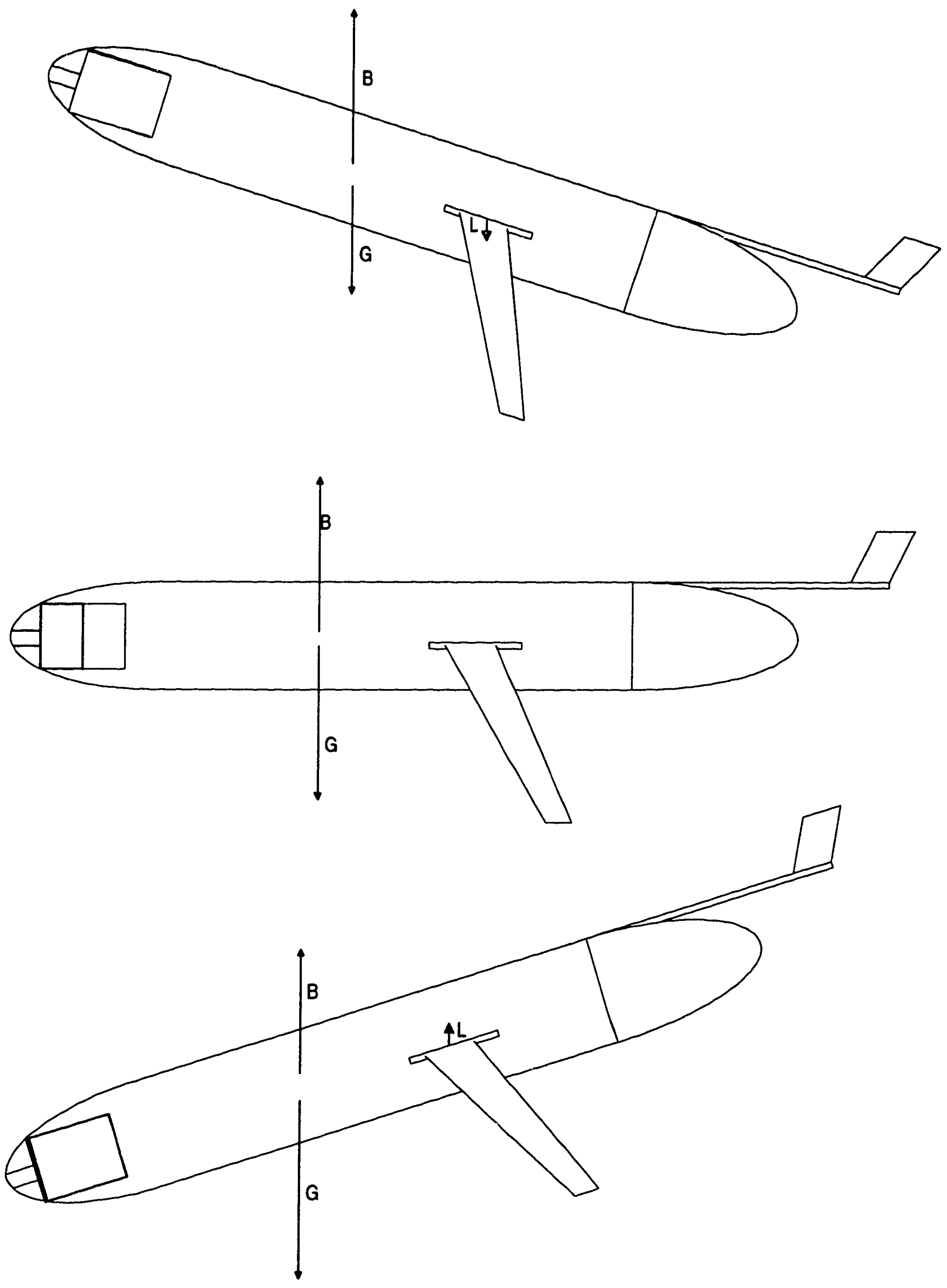


Figure 2.6a-c. Glider attitude and lift generation for a) an empty water reservoir, b) a half-full (neutral) reservoir, and c) a full reservoir. The vertical portion of lift is shown. The buoyancy and gravitational vectors are drawn much smaller than their proper scale in relation to the lift vector. In reality, the lift is less than 1% of either of these forces, hence the offset of the larger vectors is minimal.

2.6. Motion: Hydrodynamic Forces

The lift forces generated by the body and wings as the vehicle sinks are only so large as to offset the slight imbalance between the buoyant and gravitational forces. Although an approximately 48kg vehicle is flying through the water, most of its weight is supported by the water it displaces, and so essentially only 100 grams or so are contributing to the motion. If the vehicle was perturbed from its equilibrium descent position, it will automatically return to its 25° nose-down configuration, simply to align the two large forces which dominate so assuredly over any hydrodynamic forces.

A similar picture can be painted for its ascents, which are depicted in figure 2.6a. As mentioned above, the battery pack can be moved axially to provide pitch control. Typically, the water pumped into and out of the nose suffices to achieve the desired pitch angle, and the battery control of pitch is used to finetune this position or change it slightly under special circumstances dictated by the mission being run. Once again, movement of the battery pack serves to shift the center of gravity, and does not in any way change the stability of the vehicle.

2.7. Roll Stability and Banking

The vehicle is stable in roll, as in pitch, by the fact that it is statically stable. These two large forces will maintain the proper attitude in the two directions, and in fact the wings are present for their generated lift force and for banking, not for roll stability. The boom tail as well, is in fact only necessary for the antennas contained inside it. The glider would be stable without a boom tail, as in the configuration shown in figure 2.3, for example.

As in ascending, descending, and pitching, turning is also achieved by changing the relative position of the centers of buoyancy and gravity. This is realized by rotating the battery around the center axis of the glider, causing the whole vehicle to roll. As can be seen in figure 2.7, this causes a small portion of the lift force to point to one side. If the wings were aligned with the centers of gravity and buoyancy, then no moment would be generated and the vehicle would not feel any effect, however if the vehicle was designed with wings fore or aft, then this force serves to push that part of the vehicle to one side, causing the glider to rotate about its gravitational and buoyancy vectors. Since the lift changes direction on ascents versus descents, a different turn direction is realized for the same bank direction in the two cases. The picture is inverted as well depending on the location of the wings relative to the vehicle's center.

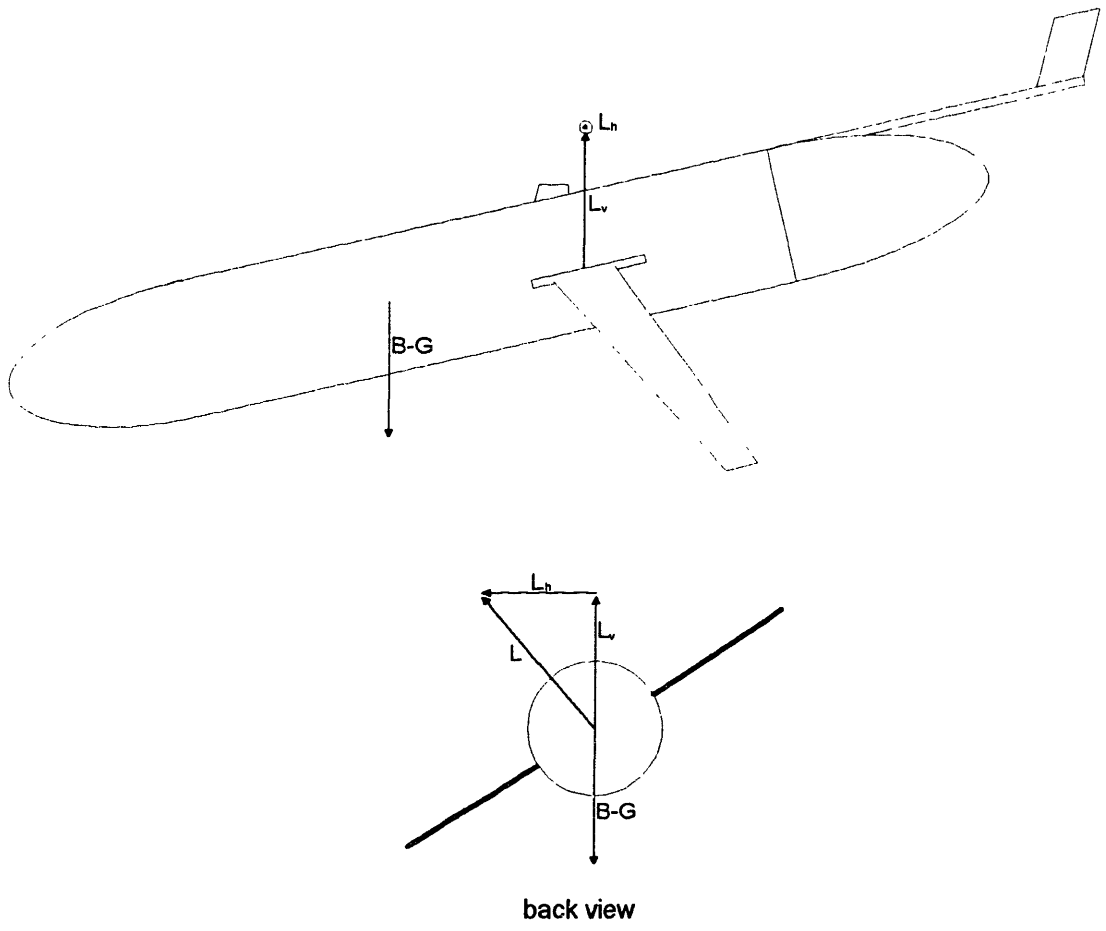


Figure 2.7. Descending glider banking left (with left wing down). The lift force serves to push the wings left, and so for aft-mounted wings, turns the glider to the right.

In the current configuration, the wings are kept well aft, so the bank directions for desired turning are as follows:

Table 2.1. Turn directions for aft wings.

	bank direction	turn direction
ascending	right (right wing down)	right
	left	left
descending	right	left
	left	right

2.8. Gliding Efficiency

The glider can be modeled as a basic fuselage as shown in figure 2.5a, fitted with an appropriate set of wings for lift. This lift has components normal and parallel to the wings and glide path. The normal component counters the controllable buoyancy while the parallel component serves to counter the drag which is generated along the glide direction. These relations are shown in figure 2.8, from which fundamental equations can be garnered.

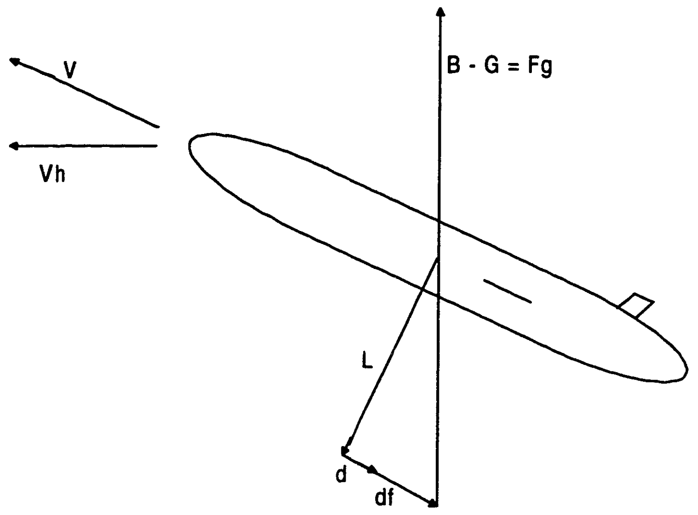


Figure 2.8. Force and velocity diagram of upward gliding. B = buoyant force, G = Gravitational force, F_g = Net drive buoyancy, L = Normal wing lift, d = Wing drag, d_f = Fuselage drag. The glide angle θ is the angle made to the horizontal.

The following analysis ignores the moments of the generated forces, since these do not affect the propulsive efficiency and performance due to the non-dissipative nature of steady-state moments. A square drag law can be used since the Reynolds number of the flow past a moving glider is generally quite constant. The assumption is also made that body lift is negligible compared with the wing lift and all drag from the body and external sensors can be lumped into a fuselage drag.

Comparisons between gliders and vehicles of differing propulsion types is usually hindered by the very different ways in which these vehicles move through the water. To this end, a useful benchmark and normalizing factor can be found in analyzing the power required to propel a wingless glider fuselage horizontally at a constant speed. This can more easily be compared with the power required of other vehicles under the same conditions, as the elimination of vertical motion and wing dissipation result in the theoretical minimum propulsion power required. To that end, for a given wing lift to drag ration $n = l/d$, the wing power ratio can be defined as the ratio of the power dissipated in the wings during gliding to this baseline fuselage horizontal power. From figure 2.8 this can be found as follows:

Using the square drag law, the ratio of fuselage drag at a horizontal speed V_h (d_{Fh}) to the gliding fuselage drag d_F is

$$\frac{d_{Fh}}{d_F} = \frac{V_h^2}{V^2} \frac{(V \cos\theta)^2}{V^2} = \cos^2\theta . \qquad 2.1$$

Geometrically from figure 2.8 we can see that the ratio of the total vehicle drag (wing drag d plus fuselage drag) to wing lift is

$$\frac{d + d_F}{l} = \tan\theta . \qquad 2.2$$

Using the wing lift to drag ratio $n = l/d$ this can be rewritten as

$$\frac{d}{d_F} = \frac{1}{n \tan\theta - 1} . \qquad 2.3$$

The above relations can be combined to yield the wing to horizontal fuselage drag ratio, or:

$$\frac{d}{d_{Fh}}=\frac{d}{d_F\cos^2\theta}=\frac{1}{(n\tan\theta-1)\cos^2\theta} \qquad 2.4$$

which when multiplied by the appropriate speeds yieds the *wing power ratio*:

$$\frac{dV}{d_{Fh}V_h}=\frac{1}{(n\tan\theta-1)\cos^3\theta}. \qquad 2.5$$

The *fuselage power ratio* is the ratio of the power dissipated in the fuselage to the baseline of horizontal motion. This can be readily obtained from equation 2.1 and trigonometry as:

$$\frac{d_FV}{d_{Fh}V_h}=\frac{1}{\cos^2\theta}\frac{1}{\cos\theta}=\frac{1}{\cos^3\theta}. \qquad 2.6$$

The total power ratio is the sum of the wing and fuselage power ratios. It is therefore a good non-dimensional benchmark with which to glean operational-optimal design information about the glider in any wing/fuselage configuration. Since it is in ratio to the baseline of horizontal motion, this is as well a good mark against which to compare glider performance to that of vehicles with different propulsion strategies, which don't necessarily move in the typical glider vertical profile shown in figure 2.2. The total power ratio is plotted in figure 2.9 for various values of wing performance (n = l/d).

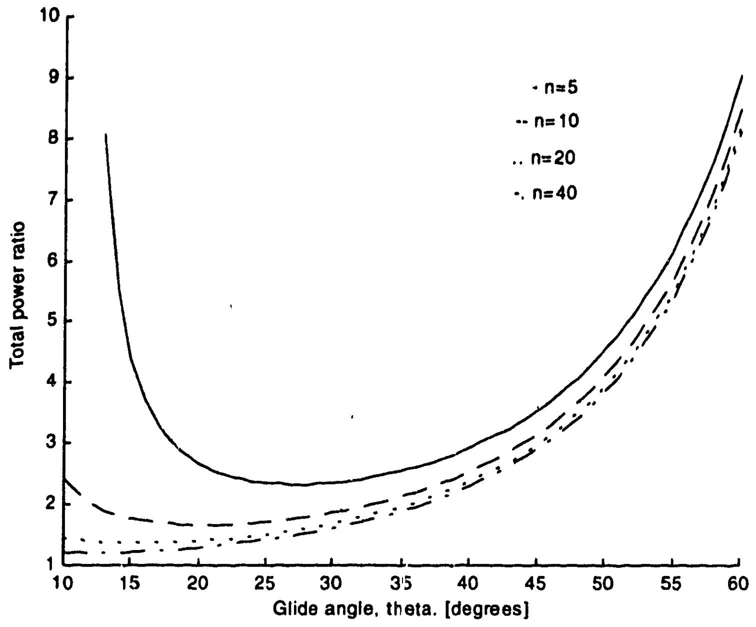


Figure 2.9. Total power ratio for various wing lift/drag ratios

Wings with a lift-to-drag ratio of greater than 10 are readily realized with flat plate wings, though wings with an l/d of 40 which can operate at the low Reynolds numbers gliders experience are difficult to operate successfully[10]. In light of this, the lower three plots should be considered for study, although the analysis holds for poorer wing performance. From these plots we can see that although the minima in the power ratios are wide, the best performance is obtained when the glide angle is kept to within +/- 5° from optimal. Practically, the optimal glide angle is aimed 5° higher than the theoretical minimum obtained from the above graph, since wing stall, with its vastly increased power ratio, onsets rapidly for smaller glide angles, as can be seen from the steep increase in the slope for n=5 at glide angles below its ideal angle.

Although a decrease in the total power ratio is obtained for wings with a higher l/d ratio, attempting to construct and use wings of a ratio of 40 garners a gain of only 30% over wings with a lift to drag ratio of 10. Although this seems a large gain at face value, it is a far more difficult method for obtaining a smaller gain than by working to reduce the body drag. Reducing drag of the hull and external appendages such as antennas and sensors can show a gain of greater than 50% with much less effort.

The wings used on the Slocum electrical glider are flat plate wings with an l/d ratio of approximately 7. This calls for a glide angle of about 21° for maximum efficiency. In order to avoid the stall zone and maintain a margin of safety in relying on ballast alone to achieve the desired glide angle, the glide angle of operation is set at 26°. As discussed in section 2.5, this, combined with the hull length, fixes the amount of ballast pumped which in turn fixes the glide speed.

2.8.1. Efficiency comparison

The full usefulness of the total power ratio is realized as a good benchmark for comparing glider efficiency to that of vehicles with different propulsion methods. A typical propeller-driven AUV, for example, has a propeller efficiency of 0.7[16], yielding a horizontal power ratio of 1.4[14]. If this AUV is used to perform water-column observations in a vertical pattern similar to a glider's natural motion, then its power ratio becomes $1.4/\cos^3\theta$, or approximately 2 for a typical glide angle of $\theta = 26°$. This is very similar to the power ratio of the gliders as presented here, if not a bit worse. This is aggravated in reality due to losses, further to the propeller, of other motors, actuators, and the ubiquitous presence of stabilizing fins in propeller driven vehicles.

2.9. Propulsion Energy

As stated by Webb [10] "the operational performance goal for an electric powered glider is to minimize the energy expended per meter of horizontal travel at a given horizontal speed". Besides operating at an appropriate glide angle and reduction of drag, there are many factors that must be equated for this to be realized. To increase the horizontal distance covered per amount pumped the angle of ascent and descent should be a minimum. To save energy, internal actuator use should be kept to a bare minimum. Since most of the operational pitch can be achieved by the change in pumped ballast, this is feasible for the pitch actuator, however achieving a small angle with a ballast change alone implies pumping less ballast, which decreases the overall speed of the vehicle. Although the vehicle would remain stable for most speeds, a vehicle speed that is less than prevailing currents is impractical since it would make the mission time even longer, negating the effects of energy savings. In reality, however, these limits are not reached, since the angle is limited by the above-mentioned practical wing considerations. As discussed above, this is limited to approximately 26°, and so the ballast pumped is set to be the minimum required to achieve this angle without the use of internal attitude corrections. Placing the ballast reservoir at the furthest point forward on the vehicle ensures minimum pumping requirements.

Using empirical relations for a typical pump used on these vehicles, we obtain the following closed form for the total pump energy (in Joules) required for a descent and ascent to and from a depth d (in meters), for d <100m:

$$E = 60V\frac{3+0.04d}{140-0.2d} \qquad 2.7$$

where V = volume of oil pumped for turnaround (cm^3)

2.9.1. Environmental effects on propulsion energy

2.9.1.1. Effect of depth

Due to the nonlinearity of the energy as a function of depth, diving twice as deep does not require twice as much energy, although it results in twice the horizontal coverage (see figure 2.10).

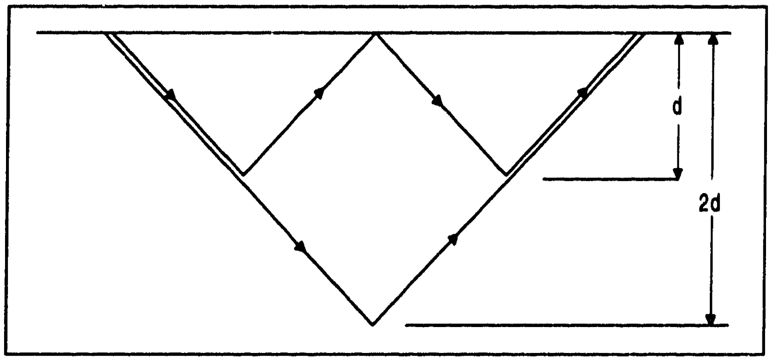
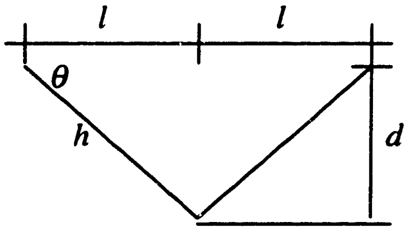


Figure 2.10. Two dive configurations for the same horizontal coverage

Using equation 2.6, we can calculate the relative propulsion energy consumption for an equal amount of horizontal travel for depths less than 100 meters and no current. Each dive at a depth *d* uses as much energy as indicated by equation 2.6 and this can be calculated per unit of travel as E/*l* by the following geometric logic:



$$l = h\cos\theta \qquad d = h\sin\theta$$

$$l = d\cos\theta\,/\sin\theta$$

The energy per unit of travel thus becomes $\frac{E}{l} = \frac{60V}{d}\frac{3+0.04d}{140-0.2d}\frac{\sin\theta}{\cos\theta}$. This is plotted in figure 2.11 for a constant glide angle of 26° and $V = 200\text{cm}^3$. For depths of 30 meters or less, which is the range of interest to this study, the propulsion energy required is more than 80% greater than that required at depths of 100 meters.

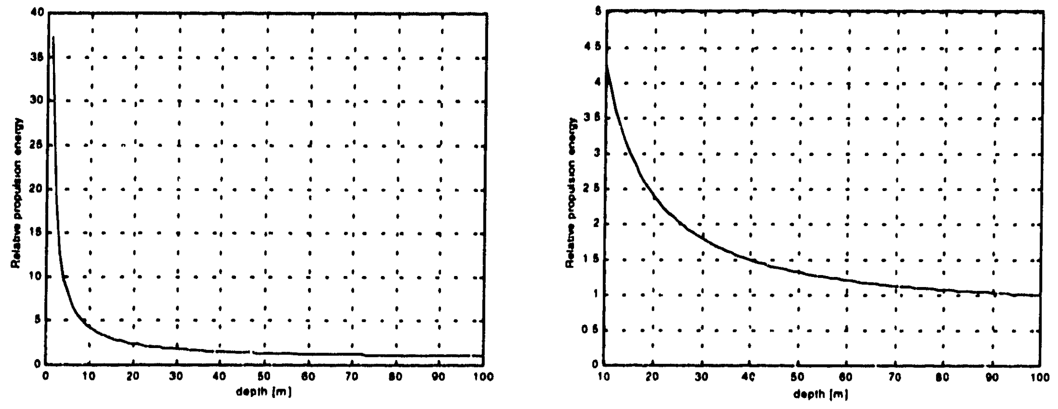


Figure 2.11. Propulsion energy for equal travel at varying depths, normalized to 100m depth. All runs performed at a glide angle of 26°.

2.9.1.2. Effect of current

Figure 2.12 shows the effects of current strength and direction on propulsion energy. The axes indicate the strengths of the easterly and northerly currents imposed on the vehicle as it traverses one unit eastwards in a straight line. For a given current, the vehicle's effective easterly velocity was calculated. This is inversely proportional to the amount of energy the vehicle uses, since depth is maintained constant. The runs were performed at the constant depth of 20m, and the results were normalized to the energy used at zero current. This is nearly equal to the vehicle's horizontal speed of 36.85 cm/s.

Although in reality the speed increases at the beginning of each ascent or descent, the assumption of constant water speed is justified by two reasons: First of all, a terminal velocity is in practice reached for most of a glide. As well, it should be kept in mind that for smooth transitions between ascents and descents, the vehicle is commanded to start pumping before it has completed the previous leg. This also helps to create a speed profile that is more constant than that obtained by the idealized straight line dive profiles of figure 2.10. Thus, with the omission of vehicle dynamics, it is sufficiently accurate to model the vehicle water speed as a constant. This assumption was used throughout the simulations run for this study.

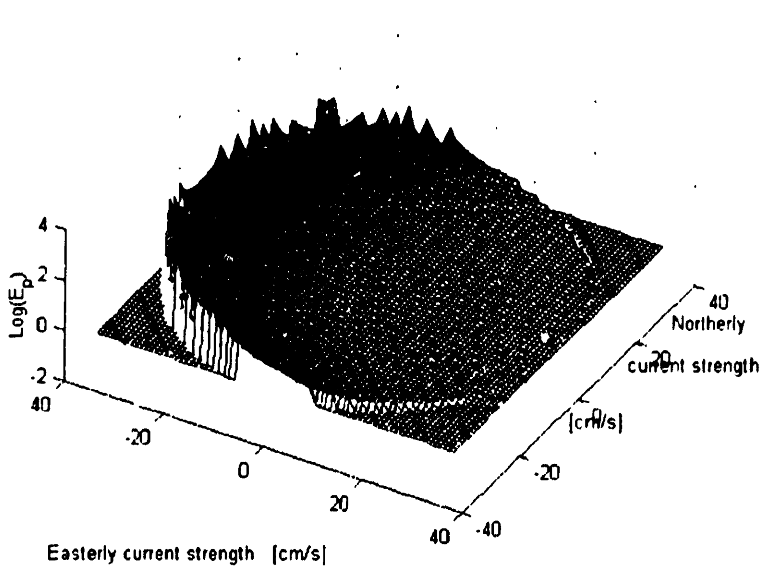


Figure 2.12. Propulsion energy used for easterly travel with constant bathymetry, normalized to the propulsion energy for no current. The maximum current used is 36cm/s. The vehicle's horizontal speed is 36.85 cm/s at a glide angle of 26°.

2.10. Hotel Load

Power consumption on a vehicle comprises the depth-dependent propulsion portion as well as a constant hotel load. This is the power consumption of all sensors, computers, and devices not used directly for propulsion, and so adds a term directly proportional to the total time. Although this load may be small, it serves to heighten even further the effect highlighted in figure 2.12 in that there is an across-the-board increase in power by a fixed percentage. For example, with the parameters used to create figure 2.12, a 1 Joule hotel load adds a 41.5% increase to the power used. Clearly, the longer the mission and the greater the overall power, the greater the absolute hotel power required.

Typical hotel loads on the vehicle are shown in table 2.2. Clearly, a 1 Watt hotel load is well within the operational range for the sensors and accessories on board, with typical operational hotel loads running as high as 2.5J. This results in a hotel that is on par with the propulsion energy consumption when the vehicle is operated in shallow water.

Table 2.2. Energy Consumption for Hotel Loads. To conserve energy, each device is only turned on for a certain percentage of the mission. Most missions operate at load levels less than the ones planned for in this table.

	% Mission on	Amps	Volts	Power [W]
Pitch & Roll	14.67	0.08	15	0.18
Piston	13.33	1	15	2.00
Air Pump	2.00	0.2	15	0.060
Altimeter	2.00	0.05	15	0.015
ARGOS	0.40	0.4	15	0.024
Pinger	0.08	10	15	0.12
CTD Profile	20.00	0.035	15	0.105
Fluor Prof	20.00	0.03	15	0.090
RF	3.33	0.18	15	0.090
GPS	4.00	0.24	5	0.048
Ocean Pressure	2.00	0.005	5	0.000
Attitude	3.33	0.012	5	0.002
Persistor	78.48	0.003	5	0.012
			Total	2.746

2.10.1. Comparison to propeller-driven

The efficiency of gliders was compared to the efficiency of propeller-driven vehicles in the previous section, and found to be comparable or better. In terms of overall power consumption, gliders fare better as well, in that their total propulsion energy usage is less than that of their propeller-driven counterparts at practical depths of greater than 10 meters. The hotel load of the existing gliders examined above is comparable to the propulsion power usage. It has been shown[17] that propeller-driven vehicles perform optimally when their speed is set such that the hotel load is twice the propulsion requirements. In reality, of course, the hotel load is dictated by the sensors on board, and propeller driven vehicles have traditionally carried hotel loads that are far greater than optimal.

2.11. Slocum Electric Glider Spec Summary

A summary of the specifications for the glider used in this study is as follows:

Length	1.8 m	
Diameter	21 cm	
Total air weight	51.505 kg	
Displacement	51.841 kg	
Position of Origin	geometric center of main hull	
Position of center of Gravity:	x (forward +)	4.06 cm
	y (side to side)	0
	z (up +)	-0.655 cm
Position of center of Buoyancy:	x	3.64 cm
	y	0
	z	-0.051 cm
Gravitational offset	0.594 cm	
Wing lift-to-drag ratio	6.9 – 12	
Glide angle	26°	
Moment required for 25° pitch	14.36 kg-cm	
Ballast piston extension	10 cm	
Maximum ballast pumped	200 cm^3	
In-line water speed	41 cm/s	
Battery pack weight	14kg	
Battery gravitational offset	-4.2 cm [z]	
Drop Weight buoyancy	-581 g	

Chapter 3

Optimal Path Planning

3.1. Introduction

In Oceanographic studies, it is often desirable to obtain a snapshot data set for particular body of water. Since it is not always feasible to gather multiple geographically distinct data points at the same time, survey missions, whether by manned boat or autonomous vehicle, are often run as reasonably quickly as possible, and errors are calculated based on the time elapsed for the mission. Another common requirement for oceanographic data involves collecting boundary value data points along a particular line in the ocean. This is useful to establish such information as the temperature of inflow and outflow in a bay. These missions are generally run on a very strict path, and errors are calculated based on deviation from this path as the data is extrapolated to fit the desired mission course.

To these ends, traditionally, autonomous underwater vehicles have generally been operated in fixed-path missions[18], and missions run for short term survey optimization[17], without much study into vehicle-optimal survey paths. This works reasonably well for fast vehicles when energy considerations are negligible, however concerns arise when dealing with slower vehicles in comparatively high currents[19], energy-exhaustive missions, or vehicles whose energy function depends on the depth of operation.

Vehicles such as gliders fall into this category, and have their use in oceanographic study as outlined in chapter 1. The concerns that arise from attempting to use such vehicles in high-current, shallow-water situations force a rethinking of the survey constraints: Error estimation has improved drastically[17] in the last few years, and ocean predicting models[19], which use the survey data as inputs, can often extrapolate data sufficiently to lessen the snap-shot data constraint. The constraint for a specific data collection path can also be relaxed to include a particular corridor of a finite width, since these vehicles can collect continuous data instead of a discrete set of points.

With the snap-shot and data-path constraints modified, vehicle paths can be fitted to a vehicle-optimal routes, or routes with a combination of vehicle- and survey-optimal constraints. Vehicle-optimal usually refers to energy optimization, whereas survey-optimal constraints include mission time and allowable corridor width.

3.2. Motivation

In recent years, interests in oceanography have moved towards more littoral waters, where underwater vehicles face high currents and shallow waters. Depths of less than 100m and currents greater than 15 cm/s are commonly found in many coastal areas[20], and areas of geographic interest[21]. Missions for glider vehicles have in the past relied on preset paths, with little regard to prevailing bathymetry or currents. Under these more adverse conditions, however, the energy and time required for completion of a mission can be prohibitively high, as follows the results in section 2.9. With the relaxed survey constraints, however, it might be possible to contrive paths wherein the savings gathered would make the use of these slow vehicles with a depth-dependent energy function more feasible. As shown in section 2.9, at the depths and currents in question, a change in a few meters or a slight change in current angle or strength makes a large difference in the vehicle performance, and mission paths might be devised to make use of this.

The problem was formulated to study the feasibility of using these vehicles in shallow waters and in currents comparable to their operational speed. These vehicle constraints are a superset of the constraints for most vehicles and so would yield the most general vehicle-optimal results. Survey optimal mission paths should also be studied under the new constraints (for example, under a particular current configuration, it might be more time expedient to deviate from a direct route, so long as the geographical deviation was not too large). Clear strategies can then be set down under which vehicle- and survey-optimal missions can be run. The problem to be studied was thus outlined as finding the time or energy optimal path between a particular startpoint and endpoint, under variable depth, time and spatially variable current, or a combination thereof.

3.3. Background

To date, there has been research pursued in the path planning of vehicles in static[22,23] and dynamic[24] environments with a priori[25] and real-time[26,27] information. Most of this literature has been in the fields of robotic manipulators[27] and land vehicles[28] in an obstacle-filled environment[26,27]. Although some work has been done on minimal time path planning, little work has been pursued on path planning in a general current field whilst taking the vehicle energy consumption into account.

In conducting a literature search in the field of path-planning between two fixed points, a few methods were commonly cited and so were explored for this work. Since current and depth data was available at discrete gridpoints, methods which worked well with a discretized path were examined first. Dynamic programming was studied, specifically British Museum and A* searches.

A British Museum search is a depth-first algorithm very useful for such tasks as finding a path through a maze. The possible paths can be represented as a tree with the root as the start point and branches for each of the directions for motion available at each node. Each branch is evaluated in turn until the end position is reached. All remaining paths are then examined and their length compared, keeping the shortest path found as the solution.

With a modification to the path length, such that the energy or time used for traversal was used to replace the physical distance as a cost, this method worked effectively for the problem presented. Unlike a maze situation, however, there are *mn* paths possible in an obstacle-free m by n grid, and so this method was very slow and cumbersome. Severe tree pruning and path selection ameliorated the problem, but the solution space remained too large for effective implementation, and so an A* search[25] was explored.

The variant of an A* search that was used for the problem at hand essentially followed the following steps for find the cost-optimal path from a point A to a point B, where the cost can be any combination of time or energy:

1. Start from point A.
2. Isolate the cardinal directions to B from A (e.g. – North and East, etc).
3. Levels are expanded in concentric diamonds – diagonal layers towards the goal.
4. Starting from level 1 (point A) each subsequent level n has n children.
5. Calculate the cost to each child from each of its possible parents. (There are always either one or two parents, depending on whether the gridpoint in question is internal or external to the grid outlined with points A and B as its opposite corners.)
6. For each gridpoint, its associated cost is that of getting to it from its better parent, plus the accumulated cost to get to the parent from point A (in other words, at each level you only have to solve for costs from the previous level - the costs aggregate, so for each position all you have to remember is its cost and its best parent). The 'better' parent is that which makes the total cost for the child the lesser.
7. When point B is reached, the path is determined by jumping back to the subsequent better parents, gathering the steps in the path.

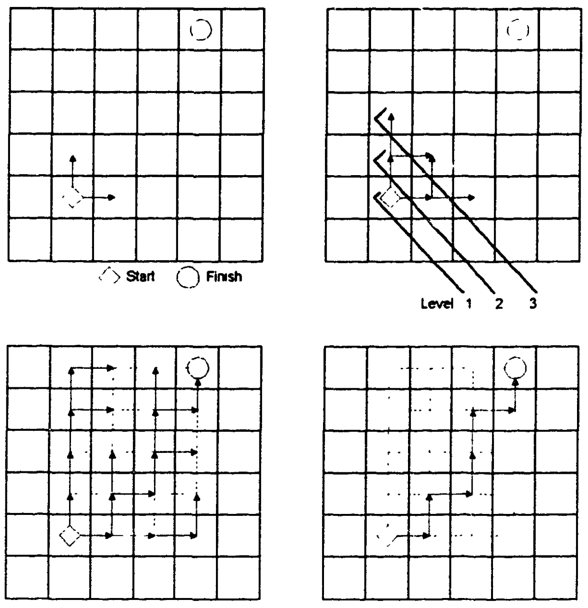


Figure 3.1. A dynamic programming approach to optimal path generation. From top left: a: The cardinal directions from the startpoint to the endpoint are identified. b: Each level is expanded along the diagonal towards the endpoint. c: Each point on a level records the better path to it, until the final point is reached. d: The path is traced back from the endpoint, giving a unique cost-optimal path.

Although this method was efficient and yielded some good strategies, the generality of the path and the size of the solution space were only adequate for certain special cases. If the start and end points were aligned along a cardinal direction, there was only one possible path (see figure 3.2). True generality (within the grid constraint) only occurred when the two points were diagonally opposite each other.

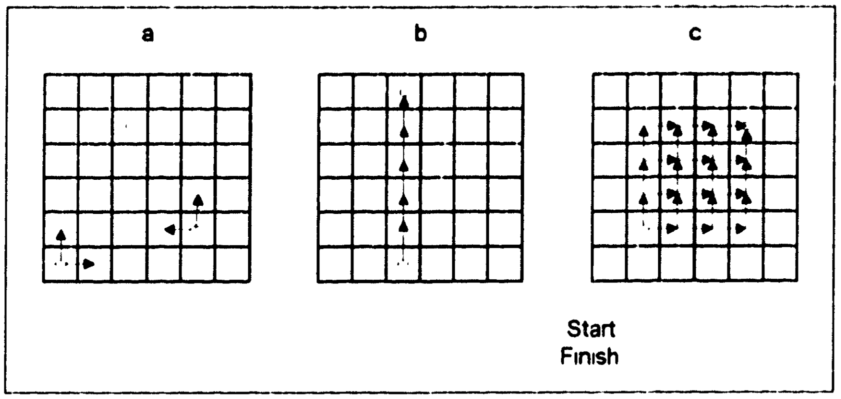


Figure 3.2. Dynamic Programming solution space extremes. a) Implemented method of level expansion; b) Collinear Start and Finish; c) Diagonal Start and Finish

Methods exist by which these problems can be avoided, such as allowing for a temporal overlap in the tree expansion (expanding on the diagonal, for example, such that there are up to three directions for expansion, and the cardinality of the grid reduces in importance) or rotating the data before using this method, such that the start and end points are always diagonally opposite each other. These were compounded by the overriding problem that each path point is a point on a discrete grid. Although grid size can be decreased, resulting in smoother paths, computationally this was judged to be too intensive to be practical. These problems urged the search for a different solution which avoids these problems and calculated paths that were free of the grid constraint and thus more true to the actual optimal path.

3.4. Methods

The method finally settled upon was a gradient descent method. It is not difficult to prove analytically that the most general solution to the zero current and constant depth problem is a straight line[29]. This result was used to initialize the algorithm, and numerical programming was applied to the problem at hand. A straight line path between start and end points was divided into a number of discrete points. The cost was calculated for this path, and then each point was perturbed one unit north and one unit east. For each perturbation, the cost of moving from the previous point in the path to this new point was calculated, so that a gradient field for the points on the path was established. Since a shorter path would demand less cost, the gradients had a major component pointing backwards to the start point. This component was eliminated, and the points were each perturbed along the resultant gradient proportional to the magnitude of the gradient (figure 3.3). To keep the paths smooth, the amount of perturbation was also proportional to the square of the distance from either end of the chain. This method converged to within a 20 meter perturbation by less than 10 iterations, with most missions converging in only three or four iterations.

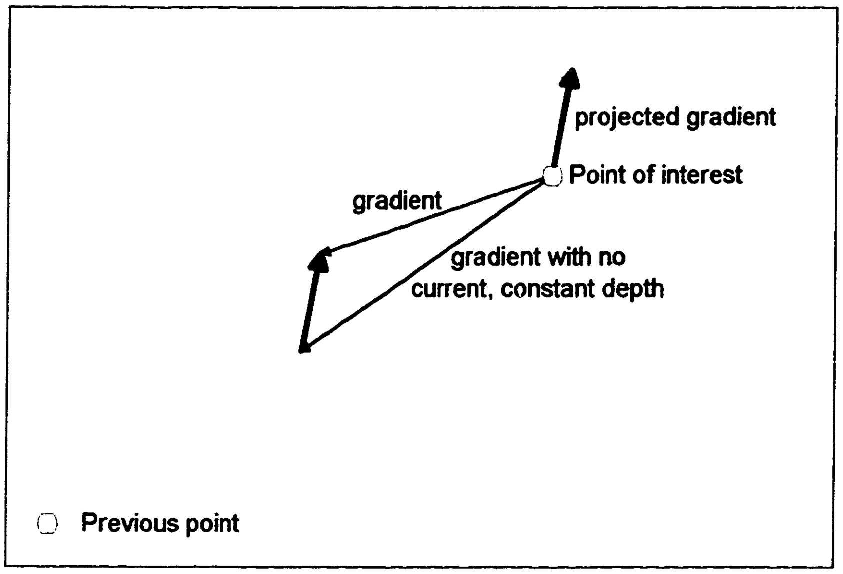


Figure 3.3. Eliminating the undesirable portion of the gradient for gradient descent.

As with all of the methods implemented, the cost function used to determine the optimal path was a linear combination of the energy and time used to travel from a fixed starting point to a fixed end point. The relative weights of time and energy were flexible to allow study of the two parameters independently.

Typical areas of interest are Massachusetts Bay and other tidal coast area with currents of about 20 cm/s and depths between less than 30 meters and up to 100 meters[20,21]. To simulate such environments, the current field for the experiments was set up with a current that sweeps predominantly southwards, with eastward and westward components as shown in figure 3.4. A 12-hour sinusoidal tidal cycle was imposed on the current, and so a three-dimensional interpolation was performed for calculations that involved the current. To do this for every calculation was very time-intensive, however, so modified current fields were constructed for typical start points. The current at every point in the grid was the value of the current at that point at a time after the start of the mission when the vehicle would have reached that point, travelling straight from the start. The accuracy of this modification was established only after the optimal paths were calculated. It will be shown that the paths did not deviate very much from a straight line, and so the currents used were correct within the error of the prediction software used as the source for the current data, which is an acceptable error.

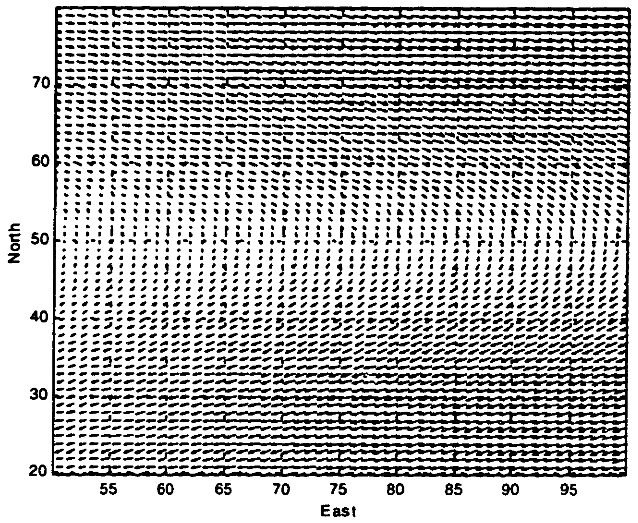


Figure 3.4. Portion of realistic current field implemented for simulation, showing the easterly and westerly portions as found in Massachusetts Bay. The highest current has a magnitude of 43 cm/s, while the entire field is on a sinusoidal tidal cycle.

The depth field implemented consisted of one smooth hole and bump, with varying steepness and a background depth of between 20 and 30 meters (figure 3.5). To test the effects of depth gradients and current intensity separately, linear depth profiles of varying gradients and uniform westerly currents (i.e.: moving to the east) of varying intensities were also implemented. These were all compared with the standard of 20 meters constant depth and zero current.

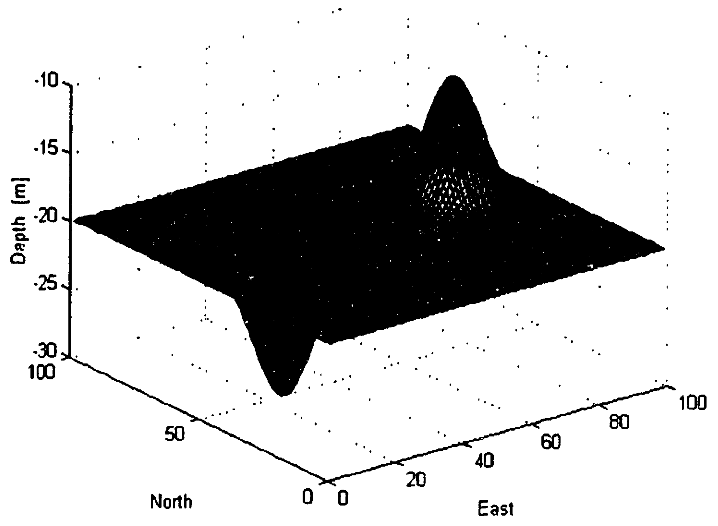


Figure 3.5. Simulation depth field implemented. The steepness and magnitude of the bump and hole were varied to study the effects of realistic gradients.

3.5. Results

Test runs were made using either propulsion energy or time as a minimization criteria. In order to study the effects of current and depth variations on each of these constraints, simple time and energy optimization missions were run on a simple one-dimensional current or depth gradient.

3.5.1. Savings based on current strength

A constant Westerly current was set up and the start and end points fixed directly North-South. For all of the runs, savings were accrued by moving east with the current for the first half to two-thirds of the mission and then turning westwards at a steeper angle for the remainder of the trip. The savings thus garnered were surprisingly large (see figures 3.6 and 3.7) over the initializing direct route, with a percentage savings approaching 50% in propulsion energy for when the current was set to 35cm/s with a vehicle horizontal water speed of 36.85 cm/s.

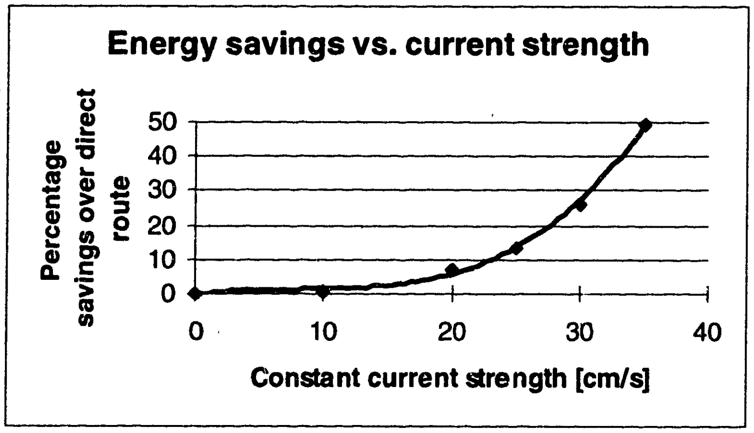


Figure 3.6.

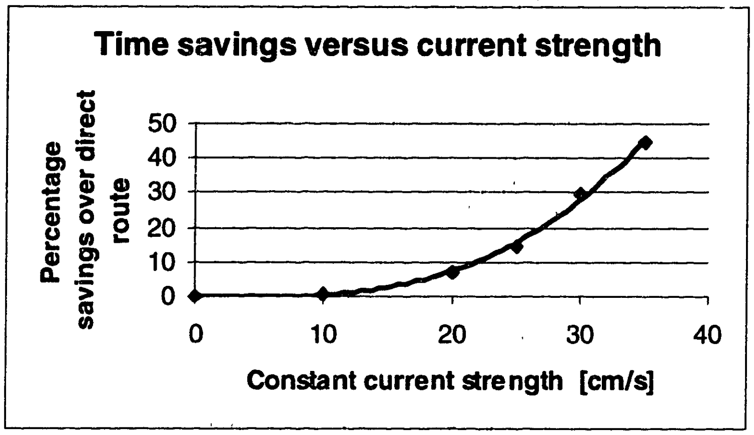


Figure 3.7.

3.5.2. Savings based on depth gradient

Easterly depth gradients of 1, 5, and 10 meters per gridspace (generally assumed to be 50 meters) were used. For any orientation of start and end points, the optimal path for time minimization was a straight line. This stands to reason from the assumption of constant horizontal water speed and glide angle, since with these assumptions the vehicle will have the same horizontal speed regardless of depth or stage of glide.

Depth gradient, however, did affect the energy optimization, as is expected with a depth-dependent energy function. The paths deflected towards the deeper regions in every case, constrained by the fixed endpoints. For various gradients, the deflection in the path (as a percentage of total path length) was:

Gradient Strength [m depth / 50 m east]	% Deflection
1	2.33
5	1.5
10	0.4

Table 3.1. Depth gradient strength versus percentage path deflection

The optimal path, therefore, is seen as a balance between a path that is short and a path that deflects to take advantage of deeper waters. With the steeper gradients the path did not need to deflect as much as with the shallower gradients in order to accrue the same benefits. All of the paths were limited in their amount of deflection by the fact that a further deflection would have made the overall path longer, so the benefits amassed from moving to the deeper waters would have been negated by the loss from the total travel time.

The total savings in proportional energy savings reached a plateau with steeper gradients. With a gradient of one meter of depth for every grid space (50 meters) in the East direction, and a depth of 35 meters at the start and end points, the deflected path showed a savings in propulsion power of just over 5%. When the gradient was increased to 10 meters of depth per 50 meters easterly, this saving increased to just over 11%. Further increases of the gradient did not show a marked improvement, since from figure 2.11 we can see that energy usage is affected more at shallower depths.

3.5.3. General results

To obtain some qualitative results, test runs were performed on the more realistic depth and current profiles. Each mission was run with a variety of depth variations, current intensities, or both. Some cases are highlighted to illustrate the strategies learned:

Figure 3.8 shows the energy-optimal path for those specific start and end points. As predicted, the path curves to incorporate as much of the deeper regions as possible, without making the path too long to negate the accrued benefits. This path changed by less than 5% when the current was increased from zero to the point where the maximum current found during the run was equal to the vehicle's speed. This indicates that with low hotel loads, runs that use energy minimization as a criteria are far more dependent on depth than on current.

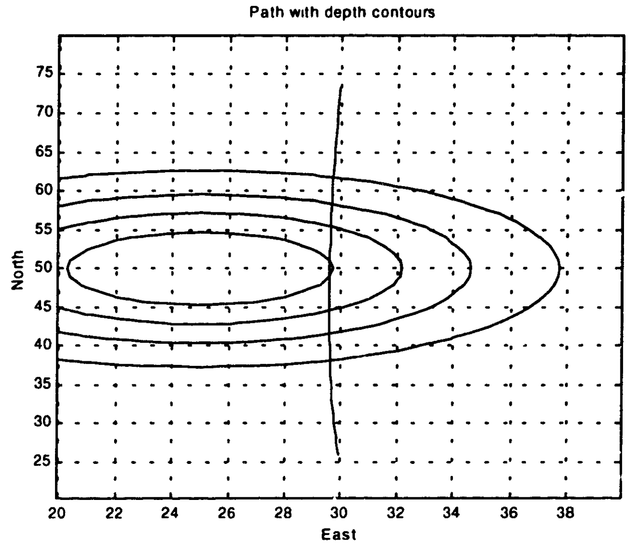


Figure 3.8. Energy minimization path for indicated waypoints. (The path is unchanged for starting at the top or bottom) Current made negligible difference to the optimal path. Depth contours indicate a gaussian-shaped hole with a maximum depth of 30m and a base of 20m.

The example shown in figure 3.8 reaps a savings in propulsion power of less than 2.3% over the straight line path between the waypoints. It was surmised that this was due to the fact that most of the depth gradient occurs only in the central portion of the path, and certainly the savings found in the previous section are larger. Although it is generally true that to minimize energy consumption it is better to deflect the path to deeper depths, a trade-off between shallow area avoidance and path length is a delicate balance. For example, when the start and end points were placed on either side of a bump, a change in the average depth of the bump from 22.1 meters to 26.6 meters (where the background depth was 30 meters) negated the effect of diverting around the bump altogether.

In contrast to energy minimization, as discussed in section 3.5.2, time minimization is dependent only on current. Figure 3.9 shows the path obtained on a time minimization run, where the start and end points were such that the vehicle experienced a strong easterly current first, then a strong westerly current, all at a constant depth. The optimal path in this scheme was a curved path that best accommodated the two currents without greatly extending the total path length. The time-optimal path showed a savings of just over 10% in cost.

Figure 3.10 is the same run with a bump placed in between the start and end points. The optimal path did not shift from that found under constant depth, proving that with the assumption of a constant horizontal water speed, time minimization runs are insensitive to depth variations.

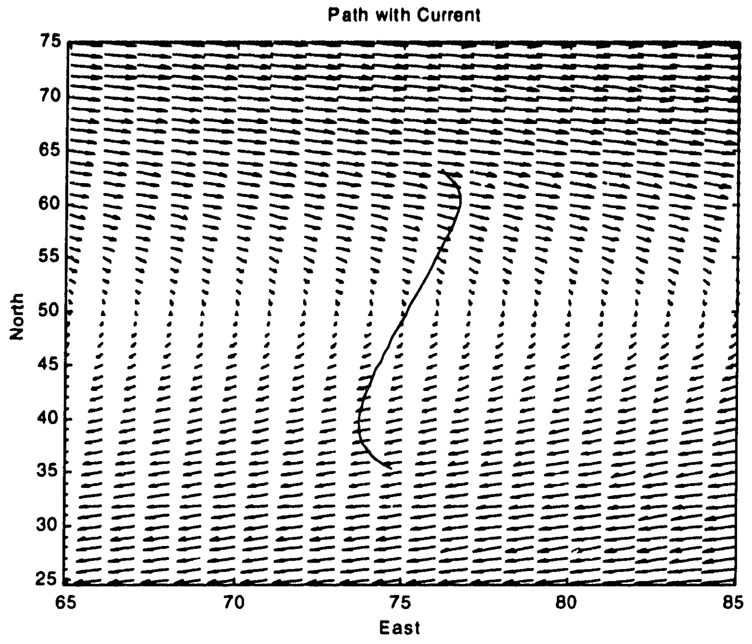


Figure 3.9. Time optimal path with indicated current. Once again, the path is unchanged whether starting from the top or the bottom, due to the prevailing symmetry. The current indicated is that which the vehicle encounters at each point, when starting from the indicated start point and moving in a straight line. The maximum current magnitude shown is 43 cm/s.

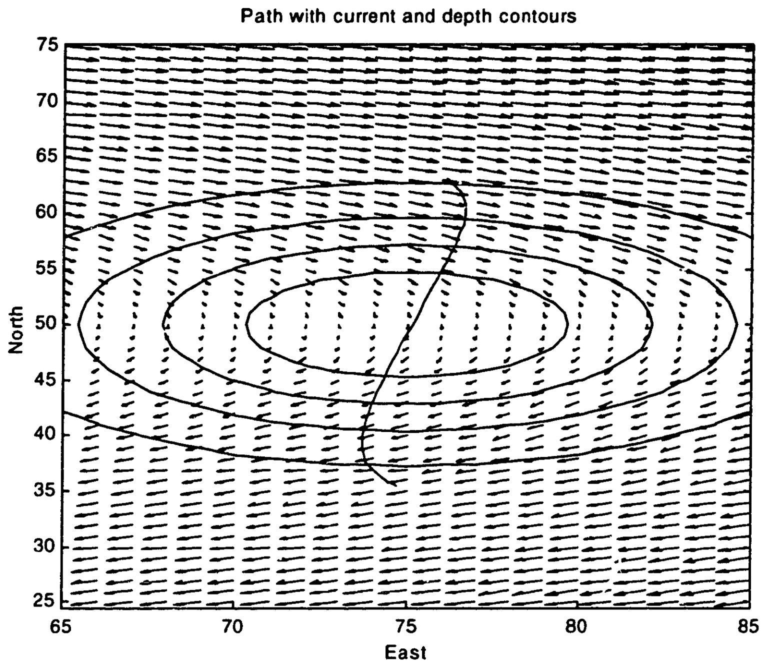


Figure 3.10. Time optimal path with depth contours and current. Maximum current magnitude is 43 cm/s. Depth contours indicate a bump with minimum depth of 10m and base depth of 20m.

3.6. Sources of Error

Although these simulations provide some reliable strategies for running missions with real gliders, there are some inherent difficulties that must be addressed before the results are strictly applied. First of all, any minimization algorithm such as the one implemented here is subject to the problem of local minima. If in one of its iterations the algorithm found a local minimum, the algorithm would not necessarily find the global optimum. In general, the algorithm would most likely settle in the local minimum closest to the straight line path. This is not an issue with the constant current and simple gradient tests implemented to study the effects of current and depth, since the linear inputs cannot yield more than one minimum, if any at all. So, the fact that a minimum was found can be safely taken as that the global minimum was found. In more complicated cases, such as the more realistic cases presented here, it is conceivable that more than one minimum is possible. Since the deviations of these paths so closely resembled the deviations of the simple cases presented aforehand, however, one can rely on the results presented herein as being global minima.

The way in which the path is perturbed, is another possible source of error. In order to fix both endpoints, the path was constrained from both directions, with the result that some paths had a noticeable jog in the middle of the path points. Although all of these paths were rejected as the optimal, it brought to attention the fact that the way in which the path was perturbed was not the most general way possible, although it can be argued that it is effective given the two-sided constraint enforced in this study. The true optimal path could very well have a different perturbation pattern, and better paths might have been found if the constraint had been softened to include start and end regions, instead of fixed points.

The final problem is associated with the way in which time-variant currents were handled. This method is perfectly acceptable for the simulations run here, and is also acceptable when running the simulation with current data obtained from an ocean prediction model, since the error in the data itself would be larger than the error incurred by the geographic and temporal displacement of the currents in our data manipulation. If the simulation were run on actual current data, a slight error would be incurred, although the results found by this algorithm would still provide a good benchmark for comparing actual energy and time expenditures on a mission to a theoretical optimum.

3.7. Conclusions

Each of the paths found did not vary greatly from the straight line path which is ideal with no current and constant depth. The largest variation was found in time minimization trials with currents on the order of the vehicle speed, such as that depicted in figures 3.9 and 3.10. In these cases, total path length increased up to 10%. In any trial, it was evident that the savings of an overall short path had to be balanced by savings garnered from diverting towards deep water or moving with a strong current.

Results from dynamic programming were similar in strategies to the results highlighted above. For energy minimizations, current was less of a factor than depth considerations, whereas time minimization runs were insensitive to depth variations. Due to the discretization of the grid, however, all paths found by this method were the same length, and so the balance between savings gained from moving with a current or around a bump and savings lost by increasing path length was not present. In this way, numerical programming proved superior to the dynamic programming scheme applied.

It is important to keep in mind that these simulations were performed in a dynamics-free environment with full a priori information. Although in the scale of a long mission the dynamics do not play a large role, real runs do not often have the benefit of full a priori information. When any current or depth information is available, it should be used in the path planning for the mission. The strategies learned here can also be incorporated into the vehicle intelligence, and gradients in current and depth utilized as the information becomes available.

Chapter 4

Control

4.1. Introduction

All autonomous vehicles require intelligence in order to operate successfully. Knowledge of their task, environment, and a way to relate to their environment are fundamental constructs that, although instinctual to humans, are nontrivial to implement. There have been countless methods proposed and implemented to provide machines some semblance of autonomous operation[30], with varying degrees of success. Schemes have been executed which mimic the apparent reasoning of creatures from insects[31] to humans[32]. Other methods have involved new ways of thinking based solely on computer logic[33], sometimes using parallel systems[34].

Missions for an autonomous underwater vehicle are usually either one or a combination of two types: survey or goal oriented. A survey oriented mission will have as the prescribed task a desired track for the vehicle to follow, gathering data as it goes. This might be a simple lawnmower type survey of the ocean bottom, or a more complicated three dimensional survey of the support structure of an oil rig. A goal oriented mission might be to follow a moving feature such as an ocean front. Missions which are a combination of the two generally include surveying a particular track or region until an event which meets the goal of the mission. This might involve surveying a region for mines, for example, and changing operating modes when one is located.

4.2. Background

To run these types of missions on the same vehicle with the same control architecture, a very flexible design has to be implemented lest the coding become terribly cumbersome in trying to deal with all possible operational modes. Control architectures for underwater vehicles are nearly as varied as the vehicles themselves[35,37,37]. The Sea Squirt[31] and Odyssey vehicles[38] designed and built by the MIT Sea Grant AUV Laboratory have successfully used a layered-control based architecture[39,40,41] developed especially for use on small autonomous underwater vehicles[37,42]. The initial architecture involved a sliding-mode controller for the three control modes (the heading, speed and depth of the vehicle) working in conjunction with a layered control scheme to determine the appropriate commands. Layered control in this implementation was used to cascade from one simple behavior to another, to determine the next safe command in each of the control modes. Each behavior was thus only required to control one relatively small aspect of a complicated mission, with the higher level decisions left to the resolution of the layered control architecture. This allows for flexible planning of relatively complicated missions from simple behaviors[41,43].

This architecture was honed into a three-level controller which comprises the following parts:

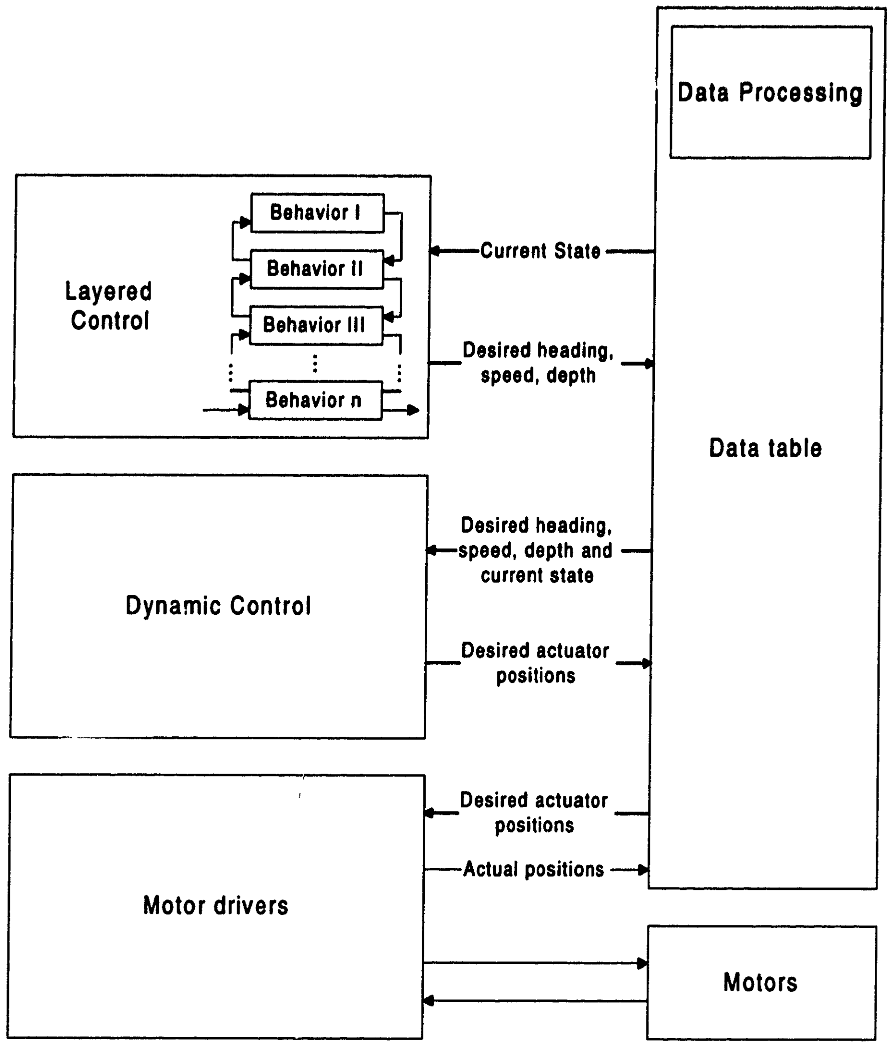


Figure 4.1. Control software architecture for use on small autonomous underwater vehicles.

4.3. Motivation

Based on the success of this architecture on small autonomous underwater vehicles, it was desired to implement such a scheme on the gliders. A direct port was not possible since the software written was specifically developed for the Odyssey class vehicles. Spurred by the desire to write working code for the gliders, the issue also arose of writing such a control architecture in a vehicle-independent manner. It was thus decided to incorporate the successful aspects of the existing code, as well as the original programmers' and new ideas for improvements, into a vehicle-independent platform and use the Webb Research Corp. electric Slocum glider as the test bed. These gliders have had only simple timing mechanisms in place, and so have not been able to run modular missions. New, glider-specific code would of course be necessary, especially at lower levels of control, however this was to be written with vehicle-independence in mind, and so should remain portable, modular, and easy to alter.

4.4. Glider Control Architecture

The architecture as shown in figure 4.1 was maintained, and in fact the resulting code is more true to the theoretical design than its predecessor. The main control levels (layered control, dynamic control, and motor drivers) communicate solely via the data table. The work described in the following sections outlines the higher level of control, through to dynamic control. This includes the data table, sensor processing, layered control with its associated behaviors, and dynamic control. Details of direct actuator control are entirely vehicle specific, and even specific to vehicle configurations, and so were not dealt with in this work.

4.4.1. Data Table

The data table comprises any variable or constant that are:

- used as interaction flags between the various control levels,
- indicate a vehicle state (e.g. heading, position, battery voltage..),
- vehicle-specific operational constants (e.g. factor to convert desired roll angle into actuator position), and
- vehicle specific hydrodynamic constants

A specific naming protocol was implemented to facilitate user manipulation of the variables required for vehicle changes. This was found to work quite well as it has been used successfully by a few people with no prior knowledge of the vehicle code.

4.4.2. Sensor Processing

Derived quantities such as heading rate (which is calculated from compass and time data) are calculated from available data in sensor processing. Besides implementing the calculations for these quantities, a function that converts GPS latitude and longitude data into the vehicle's base Universal Transverse Mercator coordinates was written. This function is used whenever the vehicle has successfully obtained a GPS fix. During control cycles between fixes, a dead reckoning algorithm calculates the coordinates based on elapsed time and compass readings.

4.4.3. Layered Control

Layered control is the part of the code which resolves conflicting commands sent by different aspects of a mission, based on a predetermined hierarchy. Each task is coded as a behavior, so that different behaviors can be combined in almost any order to achieve a unique mission. The most basic unit of operation for a glider vehicle is one down-up cycle. Most survey missions involve a combination of underwater waypoint-traversals (all the while performing the typical glider cycles), along with surface intervals between cycles to obtain GPS fixes, transmit collected data, and obtain new instructions. Behaviors were written to achieve such missions, to test various hardware components, and also to maintain a layer of safety during testing and while running missions. Behaviors are programmed into a mission by calling them by name, and specifying as many behavior arguments (b_arg) as desired. The behaviors are listed in the master data list, so that their arguments' default value is recorded, and, if a different value is not specified, used at run time.

4.4.4. Behaviors Implemented

4.4.4.1. Mission Abort

```
Arguments:                  Name (units)default
                                        value

    b_arg:            max_time (    s) 0
    b_arg:   battery_min_power (    V) 0
    b_arg:           max_cycle ( int) 0
    b_arg:           max_depth (    m) 0
    b_arg:           air_bladd (bool) 1
    b_arg:   abort_on_pressure (    s) 0
```

This behavior monitors the time, ambient pressure, battery voltage, total number of cycles and present depth of a mission. If a set maximum time (max_time) or number of cycles (max_cycle) is surpassed or the measured battery voltage dips lower than battery_min_power before the end of the programmed mission, this behavior aborts the mission by pumping out all of the water, turning on the air pump and rf modem and waiting until it reaches the surface before timing out. A harsher abort is called if the vehicle depth exceeds the max_depth, wherein besides the aforementioned actions, a dropweight is released and the pinger is activated to try to telemeter the vehicle's location underwater. If any of these parameters are not set (ie: left to default) the corresponding safety check is not performed. If the abort_on_pressure flag is set to a value greater than zero, the mission is aborted (by the softer abort) if no change in ambient pressure is recorded within its specified number of seconds (and the vehicle is not on the surface). This latter precaution is to guard against getting tangled in underwater obstacles (such as lobster pots). Upon aborting, the vehicle makes itself positively buoyant for recovery. All of the ballast is pumped out, and if an air bladder exists (air_bladd = 1) it is pumped full.

4.4.4.2. Glider_yo

```
    b_arg:          duration (    s) 0
    b_arg:            cycles ( int) 0
    b_arg:         min_depth (    m) 0
    b_arg:         max_depth (    m) 0
```

```
b_arg:          min_altitude (   m) 0
b_arg:       abort_when_done (bool) 0
```

The glider_yo behavior controls the vehicle's most basic vertical cycles for a set amount of time (`duration`) or number of cycles (`cycles`). The maximum and minimum depths and a minimum altitude from the ocean bottom can be set to obtain different profiles at various points in a mission. This behavior will run for a `duration` amount of time, a number of `cycles`, or whichever comes first. If either of these arguments is not set its corresponding exit is ignored. This behavior has the ability to end a mission. If the `abort_when_done` flag is set high, than when it has completed its run the mission aborts. In this way the vehicle is assured to be on the surface at the end of a mission.

4.4.4.3. Depth Envelope

```
b_arg:             max_depth (   m) 0
b_arg:             min_depth (   m) 0
b_arg:        alt_env_active (bool) 0
b_arg:          min_altitude (   m) 0
b_arg:  depth_cutoff_active (bool) 0
b_arg:          cutoff_depth (   m) 0
```

This behavior was modeled closely on an existing Odyssey behavior and is used as a safety margin over the glider_yo behavior. It corrects the commanded depth if for some reason the vehicle has strayed from the prescribed depth envelope or the commanded depth is out of these bounds. This envelope is bounded above by `min_depth` and below by `max_depth` and, if the `alt_env_active` flag is set and the altitude sensor is active, also `min_altitude`. If `depth_cutoff_active` is set greater than zero (1 = true) then the mission is aborted if the vehicle surpasses the `cutoff_depth`.

4.4.4.4. Waypoint

```
b_arg:          wpt_n (    m)  0
b_arg:          wpt_e (    m)  0
b_arg:          speed (  m/s)  1 #default <0 ie: nominal
b_arg:        capture (    m)  1
```

```
b_arg:      circle_mode ( bool)  1
b_arg:          heading (  rad) -1
b_arg:             gain (rad/m)  0.0175
b_arg:          timeout (    s)  0
b_arg: abort_on_timeout ( bool)  0
```

Waypoint is also modeled on a well-tested Odyssey behavior. It allows for various modes of waypoint capture by means of the `circle_mode` flag and `heading` parameter. If `circle_mode` is set to greater than zero, then the vehicle must arrive at the waypoint (specified by `wpt_n` and `wpt_e` in the internal Universal Transverse Mercator units) to within a radius set by `capture`. Otherwise, the waypoint is considered captured only when the vehicle has passed it along its heading. If `heading` is set to an allowable heading value (nonnegative) then the vehicle will attempt to capture the waypoint along the prescribed direction. The behavior automatically times out if `timeout` is reached before the waypoint is captured, and the mission aborts if the abort flag is set.

4.4.4.5. Communicate

```
b_arg:           duration (   s) 0
b_arg:         num_cycles ( int) 0
b_arg:   cycle_multiplier ( int) 10
b_arg:      when_transmit ( int) 0
b_arg: next_surface_cycle ( int) 10
b_arg:    abort_when_done (bool) 0
b_arg:    time_for_listen (   s) 120
b_arg:       time_for_gps (   s) 60
b_arg:   time_for_surface (   s) 120
b_arg:        air_bladder (bool) 1
b_arg:          top_depth (   m) 5
b_arg:      surface_depth (   m) 1
b_arg:         comm_state (flag) 0
```

This behavior is responsible for obtaining GPS fixes at various points in a mission, and transmitting some of the gathered data. It also ‘listens’ for any commands sent by a remote operator, to allow for a change in commanded mission, a premature abort, or some other remote control. The behavior activates after a set number of cycles (`next_surface_cycle`) to obtain a GPS fix and listen for commands, and every `cycle_multiplier` thereafter for its duration. It transmits a set batch of data every when_transmit surfacings. The behavior waits until the top of a cycle to surface, with the allowable depth for activation being set by top_depth. If an air bladder exists it uses its extra inflation to obtain maximum buoyancy. The GPS unit and RF modem are turned on when the vehicle has reached the surface_depth.

Each communicate behavior is active for either a duration amount of time, or num_cycles number of surfacings. During communication periods, this behavior overwrites all other behaviors except safety behaviors so that the vehicle is not commanded to perform any tasks except communicate. This behavior was by far the most complex to implement, in that it requires a different mode of operation than other behaviors and was wholly unlike any other behavior previously written. Behaviors do not directly control the actuators or devices, but instead rely on flags in the data table to pass information on to the device drivers. This is the best structure for a layered control approach, so that behavior commands can be resolved by level of importance and precedence. Communicate, however, requires bypassing of all other behaviors and a fairly direct command of the various actuators since its tasks are highly sequential. For example, it must command the air pump to be activated and wait until a top depth is reached before the GPS is turned on. This was achieved by using internal states (comm_state) in this behavior. The commands from one state are passed onto the device drivers and the next state is not entered until the appropriate conditions are met. Different avenues are taken if a preset time is exceeded for any of the commands (for example, if getting a GPS fix takes longer than time_for_gps seconds). Like glider_yo, this behavior can command the end of a mission upon its completion, thus ensuring that the mission ends on the surface and immediately after a GPS fix is obtained (or at least attempted).

4.4.4.6. Set_roll, Set_pitch, Set_ballast

These three diagnostic behaviors have essentially the same structure. The parameters for set_roll are:

```
b_arg:                duration (   s) 0
b_arg:                    roll ( rad) 0
b_arg:                   reset (bool) 0
```

```
b_arg:              reset_roll ( rad) 0
```

This behavior commands a roll for `duration` time, after which, if `reset` is set to true (=1) it commands a `reset_roll` before timing out.

4.4.5. Dynamic Control

The dynamic controller implemented maintains the separation into three control modes (Speed, Heading, and Depth). Breaking down control into these three modes separates the control into the horizontal and vertical planes (which decouple naturally in vehicle dynamics) and the vehicle water-speed. Although the actual conversions and calculations within these control modes are somewhat vehicle-specific, this breakdown is appropriate for any underwater vehicle yet designed.

In each control mode, higher level controls calculate a lower level command, on down until the final lowest level command (an actuator position) is calculated. The higher level calculations (such as converting a desired waypoint into a desired heading) are not vehicle specific, with specialization occurring further down the control chain (for example calculating the exact amount of battery roll to achieve a certain roll to turn the vehicle). The control schemes for each control mode are as follows:

4.4.5.1. Speed Control

The relation between vehicle speed and amount of ballast pumped for the glider configuration used in this work is:

$$\text{desired\_ballast} = 1107.3 * (\text{desired\_speed})^2 + 7.7691 * \text{desired\_speed}$$

where desired_ballast is in cm^3
desired_speed is in m/s

This relation was obtained from water trials, and used to determine the required ballast when a water speed sensor is not available. If speed control is entered by a commanded speed, then this calculation is executed, and the desired ballast is passed to the ballast control. The ballast control can be controlled directly, or it takes the result from the higher level speed control. In either case, it makes sure that the desired ballast does not exceed operational limits, and commands an acceptable ballast. This final command is the only quantity from dynamic control that is used by the actuator drivers.

4.4.5.2. Depth control

Since ballast is controlled both automatically (based on the glider's continuous vertical cycles) and by speed control (based on how fast the vehicle is commanded to float and sink) the depth control for a glider becomes a vertical attitude controller, ensuring that during its cycles the vehicle maintains the appropriate pitch. Most of the proper pitch is established by the change in ballast, and the depth controller has only to finetune that attitude. To achieve this, and to maintain the maximum number of control levels, the controller is structured as follows:

```
If we aren't in a turnaround section (at the top or bottom of a cycle)
and if all of the commanded ballast has been pumped (to a reasonable
error)
check to see if the pitch is correct.

If the pitch is off by more than a reasonable error,
an ideal pitch is commanded.

The pitch is corrected by moving the battery an appropriate amount
forward or backward, determined by a simple proportional calculation (cut
off at operational safety limits)

If we're in a turnaround
set battery position to neutral since the pumping ballast should take care
of the pitch.
```

All of the constants for these calculations, including the acceptable error ranges, are easily changed by the mission programmer. With this structure, the pitch is automatically corrected as necessary, but can be overridden by direct pitch or battery position commands.

4.4.5.3. Heading control

The heading control is more complicated than the other two control modes in that it can take as inputs a waypoint, a heading, a roll or a battery roll position. When a waypoint is commanded it is compared to the current position and converted into a desired heading. This calculated heading, or a directly commanded heading, is fed into the following logic:

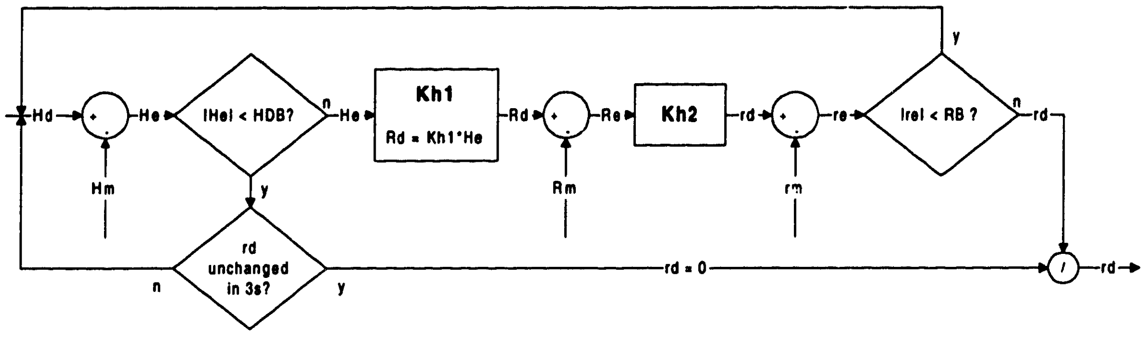
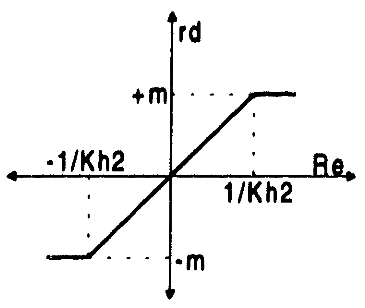


Figure 4.2. Heading autopilot block diagram

In this figure, the symbols are as follows:

Hd	is	desired heading
Hm		measured (actual) heading
He		heading error
HDB		heading dead band
Kh1		heading to rate constant
Rd		desired heading rate
Rm		measured heading rate
Re		error in heading rate
rd		desired roll
rm		measured (actual) roll
re		roll error
RB		Roll Band

Kh2 parameter that determines the following curve:



where m is maximum actuator roll

During an ascent, the above picture holds and a positive roll corresponds to a positive heading rate. During a descent, the graph is inverted and a positive roll (bank right) results in a negative roll rate (turn left). Computationally, this was implemented by assigning the following values to vertical motion and desired turn:

ascending	1	right turn (increase heading value)	1
descending	-1	left turn (decrease heading value)	-1

To calculate the appropriate bank direction, the two relevant quantities are multiplied. For example, on ascending (+1), a left turn (-1) requires a left bank (+1 x -1 = -1), or a dip towards the left wing. On descending, however, a left turn calls for a right bank (-1 x -1 = +1). The sign of the quotient corresponds with the vehicle's internal coordinates, in which vehicle roll increases clockwise when viewed from the back of the vehicle.

4.5. Implementation

Missions are programmed by adding together functionalities of the basic behaviors, with safety behaviors layered at the highest priority. Behaviors that control different aspects of the vehicle can run at the same time, while similar behaviors are run sequentially. For example, in order to run a mission which commands the vehicle to go to waypoint A, then B, while gliding between 5 and 100 meters of depth, the following is input:

```
behavior: mission_abort
    b_arg:             max_time (   s) 300

behavior: glider_yo
    b_arg:             duration (   s) 240
    b_arg:            min_depth (   m) 5
    b_arg:            max_depth (   m) 100

behavior: waypoint
    b_arg:            wpt_n (     m)  point B North
    b_arg:            wpt_e (     m)  point B East
    b_arg:          capture (     m)  50
    b_arg:      circle_mode ( bool)  1
    b_arg:          timeout (     s)  120

behavior: waypoint
    b_arg:            wpt_n (     m)  point A North
    b_arg:            wpt_e (     m)  point A East
    b_arg:          capture (     m)  50
    b_arg:      circle_mode ( bool)  1
    b_arg:          timeout (     s)  120
    b_arg: abort_on_timeout ( bool)  1
```

This mission calls for two consecutive waypoints, A and B, with 120 seconds to capture each. If waypoint A is not captured then the mission aborts. As a safety measure, if the mission is found to be running at 300 seconds, it is aborted and the vehicle should then float to the surface. The behavior list is read bottom-up, so the logic followed by the layered controller is as follows:

<u>until waypoint A is reached:</u>

Waypoint A: nothing has been commanded so far, so wapoint A is commanded

Waypoint B: a horizontal command has already been issued, so this behavior is not yet active

Glider_yo: no vertical command has been issued, so this behavior controls the depth.

Mission_abort: only commands if the mission time exceeds 300 seconds when waypoint A is reached:

Waypoint A is deactivated, Waypoint B is activated, and the top three behaviors command a mission.

A more detailed description on complete mission programming can be found in the appendices.

4.6. Conclusion

The control architecture as described above was implemented in a PC environment to run on a Persistor CF1[44] on a battery-powered glider vehicle. Lessons learned from the use of the parent code in Odyssey were incorporated into the new software, with an eye on generality of code, in order to make the code as vehicle-independent as possible. While working with the vehicles and their usual handlers, the need for a simple mission programming protocol became apparent, and so one was written and is being honed, allowing any user to program even complicated missions.

The behaviors implemented allow a range of testing and survey missions, as well as more complicated surface communication procedures. Although written with a glider vehicle in mind, most can be used successfully on any vehicle to run similar missions. Water testing was begun in July, and relatively complicated missions (for example, involving a series of glider_yos with communication at various points, all the while trying to obtain a particular heading) were carried out successfully.

Chapter 5

Concluding Remarks

5.1. Conclusions

The work done on theoretical optimization of path planning for a slow vehicle with a depth-dependent energy function gleaned strategies for effectively running these vehicles under physically adverse conditions. In certain circumstances, savings in time or propulsion energy of up to 50% can be realized with a few simple alterations to standard missions. The suggested changes keep missions within the range of acceptable oceanographic surveys and path-running, especially given recent advances in ocean prediction and interpolation technology. This helps prove the usefulness of glider vehicles to all ocean sampling networks, both those in the deep ocean and those in littoral waters.

The control architecture chosen for the vehicles proved sufficiently modular to implement the missions suggested by the path-planning work. Layered control schemes have been shown to work effectively for AUVs in the past, and so work done for gliders was modeled on existing software for an Odyssey IIb propeller-driven vehicle. Lessons learned on Odyssey were used to modify the code and the majority of the code was rewritten in order to be more vehicle-independent. The dynamic controller leaves control access at all levels of the vehicle architecture, from the highest levels of abstract waypoints to the lowest levels of actuator controls. Real missions were run on the vehicles and sea were started successfully.

5.2. Future Work

Within the body of the work outlined in this thesis, the groundwork for a functional vehicle-independent simulator was laid. With the appropriate hydrodynamic coefficients input into the program, it is possible to use the same code used to run actual missions to simulate the same missions on the bench-top, thereby avoiding possibly costly errors in the field. The original simulator written for the Odyssey vehicles was hardwired into the code and so was not vehicle independent. The necessary parameters were extricated from the code and are now available as easily manipulated variables in the main data table. It would be of great benefit to run the simulator for the gliders, as it would provide even more insight than the dynamics-free simulator written for the first part of the work outlined here.

Work is currently being undertaken at the MIT Sea Grant AUV lab to create a Graphical User Interface for the Odyssey architecture. Incorporation of this work with the control architecture detailed above would provide for a tool that is more powerful in its ease of use and wider availability.

5.3. Acknowledgements

The author would like to thank Jim Bellingham, who managed an ingenious mix of support and constructive criticism; Doug Webb without whose vision none of this work would have been possible; Clayton Jones and Jim Bales for their unique perspective on the two distinct parts of this work; and also to Knut Streitlien for helping out a distressed grad student despite his own busy schedule. Many thanks also to the MIT Sea Grant College Program, and especially the researchers at the Autonomous Underwater Vehicles Laboratory and at Webb Research Corporation.

This work was funded by ONR grants N0001498C-0281 and N0001497C-0322 and AOSN grant N00014-95-1-1316.

/

References

1 Curtin, T. B. et al. "Autonomous Oceanographic Sampling Networks" *Oceanography*, Vol. 6, No. 3, 1993

2 Kunzig, Robert "A thousand diving robots" *Discovery Magazine* April 1996 pp60-71

3 Stommel, Henry "The Slocum Mission" *Oceanography Magazine*, April 1989 pp22-25

4 Bellingham, James G. and Chryssostomidis, Chryssostomos "Economic Ocean Survey Capability with AUVs" *Sea Technology* April 1993 pp12-18

5 Simonetti, Paul "Low-Cost, Endurance Ocean Profiler" *Sea Technology*, February 1998 pp 17-20

6 Simonetti, P., *Slocum Glider: Design and 1991 Field Trials* .Woods Hole Oceanographic Institute, September 1992

7 Schmidt, H., Bellingham, J.G., and Elisseeff, P. "Acoustically Focused Sampling in Coastal Environments", Rapid Environmental Assessment SACLANTCEN Conference Proceedings Series. 1997

8 Schmidt, Henrik et al. "Real-Time frontal mapping with AUVs in a Coastal Environment", Proceedings of MTS/IEEE Oceans '96, vol 3 pp 1094-8, Florida 1996

9 Schmidt, Henrik et al. "Haro Strait Frontal Dynamics Primer Experiment", Summary and Cruise Report, Feb 1997

10 Webb, Douglas C. and Simonetti, Paul J., *A Simplified Approach To The Prediction And Optimization Of Performance Of Underwater Gliders.* Tenth Int'l Symposium on Unmanned Untethered Submersible Technology. September 1997

11 Galea, A M, *Energy Usage by Glider Vehicles.* Sea Grant: October 1998

12 Galea, A M, "Various methods for obtaining the optimal path for a glider vehicle in shallow water and high currents", UUST conference, 1999

13 Davis, R.E. et al. The Autonomous Lagrangian Circulation Explorer. Journal of Atmospheric and Oceanic Technology, Vol 9, No.3: June 1992

14 For example, the glider project at the University of Washington.

15 Webb, Douglas C. and Simonetti, Paul J. UUST conference, 1999

16 Poole, Patrick K., and Clower, Penn. "A Systems Approach to Autonomous Underwater Vehicle Propulsion Design", MTS Journal, Summer 1996 Vol 30, No 2, pp8-16.

17 Bellingham, James G., and Willcox, Scott J. *Optimizing AUV Oceanographic Surveys.* AUV 1996 Proceedings.

18 Touhy S.T., et al. *AUV Navigation using Geophysical Maps with Uncertainty.*8th Int'l Symposium on Unmanned, Untethered Submersible Technology. Univ. of New Hampshire, Sept 1993: pp 265-276

19 HOPS: Harvard Ocean Prediction System: User's Guide and Manual. Harvard 1997

20 Geyer, W. R., et al. "Physical Oceanographic Investigation of Massachusetts and Cape Cod Bays", Report to the Massachusetts Bays Program. October 1992.

21 "Tidal Flow Field at Haro Strait": Environmental Publication

22 Zelinsky, Alexander, *Using Path Transforms to guide the search for findpath in 2D.* Int'l Journal of Robotics Research, Vol 13 No 4, August 1994

23 Pellazar, Miles B., Vehicle Route Planning with constraints using genetic algorithms. Northrop Corp B-2, 1994

24 Fiorini, Paolo, Shiller, Zvi, *Time optimal trajectory planning in dynamic environments.* IEEE Int'l Conference on Robotics and Automation. April 1996

25 Stentz, Tony, The D* Algorithm for route planning in unknown terrain. Carnegie Mellon University, 1997

26 Papageorgiou M., Steinkogler A. Real-time optimal control of moving vehicles in changing environments. 1993

27 Field, Glen, Stepanenko, Yury. Iterative Dynamic Programming: an approach to minimum energy trajectory planning for robotic manipulators. IEEE Int'l Conference on Robotics and Automation. April 1996

28 Suzuki, Makato, et al. Geographical route planning based on uncertain knowledge.

29 Marion, Jerry B. Classical Dynamics of Particles and Systems. New York, Academic Press: 1970

30 Grefenstette, John J., ed., Genetic Algorithms for Machine Learning. Kluwer Academic Publishers: Boston: 1994.

31 Loch, John et al. "Software Development for the autonomous submersible program at the MIT sea grant and Draper Laboratory". 6th Int'l Symposium on Untethered, Unmanned Submersible Technology. June 1989

32 Cacciabue, Pietro C. Modelling and Simulation of Human Behavior in System Control. Advances in Industiral Control XXV: Berlin, New York, 1998.

33 Hisdal, Ellen. Logical Structures for Representation of Knowledge and Uncertainty. Studies in Fuzziness and Soft Computing, vol. 14. New York: 1998.

34 Hewitt, Carl. *Design Issues in Parallel Architectures for Artificial Intelligence.* External Memo, MIT Artificial Intelligence Laboratory: 1983.

35 Yoerger, Dana R., et al. "The Autonomous Benthic Explorer (ABE): An AUV Optimized for deep seafloor studies", Seventh Int'l Symposium on Unmanned Untethered Submersible Technology, Durham, MH, Sept 1991 pp60-70.

36 Bellingham, J.G. et al, "A second generation survey AUV", IEEE Conference on Autonomous Underwater Vehicles, Cambridge, MA 1994.

37 Smith, S.M. et al. "The Ocean Explorer AUV: A modular platform for coastal oceanography", Ninth Int'l Symposium on Unmanned, Untethered Submersible Technology, September 1995.

38 Bellingham J.G, et al "Demonstration of a High-Performance, Low-Cost Autonomous Underwater Vehicle" MIT Sea Grant Autonomous Underwater Vehicles Laboratory Report MITSG 93-28 1993

39 Bellingham James G. and Consi, Thomas R. "State Configured Layered Control" Proceedings of the IARP 1st Workshop on Mobile Robots for Subsea Environments. October 1990

40 Bellingham, James G, and Leonard, John J. "Task configuration with Layered Control" IARP Mobile Robots for Subsea Environments 1994 Monterey CA

41 Bellingham, James G., et al. "Keeping Layered Control Simple" Symposium on Autonomous Underwater Vehicle Technology June 1990 pp3-8

42 Loch, John LeRoy. "Design of a reflexive hierarchical control system for an Autonomous Underwater Vehicle" MIT, 1989

43 Consi, T.R. et al. "Exploration of Turbulent Odor Plumes with an Autonomous Underwater Robot". Biological Bulletin, Vol 189 pp231-2 October 1995

44 for further information on the Persistor CF1 the reader is referred to the web site at www.persistor.com

Appendices

Practical Glider User Manual

Updating the glider code using CVS

- Open CVS
- In the left window, find the appropriate folder (c:/CVS_Glider_Code/Glider)

To update everything:

- In the 'CVS folders' menu, choose update folder.
- The window that pops up should have the correct folder open (if not, find it)
- Choose OK (**with** recursion into subfolders)

To update one file at a time:

- Choose 'update file' (either from the 'CVS files' menu or by right-clicking the name)

To see the changes in the file numbers, press F5 for a refresh.

Now look at "Compiling and downloading the software"

Compiling and downloading the software

Once you have all the files (via CVS or a straight download) you need to open edit_struct.mcp and glider.mcp in Codewarrior.

Make edit_struct (in Codewarrior)

Run edit_struct.exe (doulbe-clicking on its icon should work)

- If the DOS window that opens says that a –f option is required, go to the edit menu, choose edit_struct settings, and in the C/C++ Language set the prefix file to precomp_dos.pch
- Make edit_struct again
- Run edit_struct.exe again. This time you should be able to just Press Enter to Continue.

Back in Codewarrior:
Choose the target for the glider.mcp (or just choose 'all')

Make glider.mcp

If we're directly connected to the glider:
- Open motocross, go to the Motocross menu, choose 'Post link and load'.
- Select the proper .bin file (glider to run missions, srtest, etc..)

If we're connected via the rf modem:
- Open motocross, and in the Motocross menu, choose 'Post link'.
- Select the proper .bin file (glider to run missions, srtest, etc..). This will create a .run file by the same name. It is this file that you now have to load to the persistor.
- To download this file, see "downloading to the glider"

To download missions, follow the procedures outlined in "Uploading from the glider"

Now you can run missions, etc…

How to program missions:

Mission text files consist of the following, in order:

1. Mode identification
2. Sensor start-up values
3. Behaviors

Mode Identification

The available modes are found in `load_mission.c`. They are:

```
TimingDebugMode
CommsDebugMode
DebugMode
Countdown
MonitorLayeredControl
SimMode
```

Sensor Start-up Values

Sensor values in `Masterdata` are the default values, and are the initializing values. If a sensor is desired to start a mission with a value different from that in `Masterdata`, it should be specified in the mission file

Behaviors

Behaviors are run from the bottom of the stack upwards, so the first behaviors executed should be at the bottom of the list. Typically, `mission_abort` (or `mission_timer`) appears first, as it has the highest priority, followed by other safety behaviors such as `depth_envelope`. The order of independent behaviors (such as one that controls the depth and one that controls the heading) does not matter, but for readability should be placed somewhat in chronological order from the bottom up.

To program behaviors, copy the behavior name and arguments from the behavior's explanation lines found with each behavior's code. Any sensors that are to have a value different from their default should be kept in the mission.

Description of arguments for various behaviors:

xy_waypoint:

wpt_n:	North coordinates of where you want to go, in UTM meters
wpt_e:	East coordinates of where you want to go, in UTM meters
speed:	set this to -1 to maintain nominal speed. Otherwise the software will calculate how much ballast to pump.
capture:	How close do you want to get to the waypoint?
circle_mode:	If this is set to 1, then the vehicle must get to the waypoint within a circle of radius set by capture. If it's set to zero, then the vehicle must pass the waypoint... this is useful when you're running things like a lawnmower mission. My suggestion is for now, use circle_mode = 1.
heading:	Set this to a value between 0 and 6.28 if you want the vehicle to get to the waypoint on a particular heading.
gain:	The vehicle will attempt to achieve this gain is the heading argument is set. Don't worry too much about this argument
timeout:	How long will the vehicle try to achieve this waypoint. After the timeout time has passed, this behavior is deactivated
abort_on_timeout:	If timeout is reached, do you want the mission to end? If so, set this to 1. If you want the mission to continue, then leave it at 0.

communicate:

duration:	Total time from the beginning of the first communication to the expected end of the last communication
num_cycles:	How many communicates the mission will have
cycle_multiplier:	The number of yos between communicates in the body of the mission
when_transmit:	Every when_transmit surfacings the vehicle will transmit something. Right now this doesn't actually transmit any data.
next_surface cycle:	After how many yos does the FIRST communication occur?

abort_when_done:	If the last communicate should also be the last thing in the mission, set this to 1.
time_for_listen:	How long should the vehicle stay at the surface waiting for a human to talk to it?
time_for_gps:	How long should the vehicle try to get a GPS signal? This is entirely uncoupled from the time_for_listen.
time_for_surface:	How long the vehicle has to get from the top_depth to the surface_depth.
air_bladder:	If an air_bladder exists and should be used, then this should be set to 1. Otherwise it should be set to 0
top_depth:	The depth at which communicate takes over control and turns on the air pump. ALWAYS set this greater than or equal to the min_depth in the glider_yo.
surface_depth:	The depth at which it starts trying to get a GPS fix.
comm_state:	This is an internal recording state... can be used for fancier missions, but generally left to zero.

mission_abort:

This behavior should never actually activate in a mission.. It is entirely a safety behavior. That said, though, here are the safeties it involves: Note that if any of the arguments are left at default, then that parameter is not looked at for abort.

max_time:	The maximum time (in seconds) allowed for the entire mission. Set this to greater than the estimated time for the entire mission.
battery_min_power:	The battery voltage at which the mission aborts. Currently it assumes that an abort will be called for a voltage greater than 3V.
max_cycle:	The maximum number of cycles allowed for a mission. This is in place just in case something goes wrong with glider_yo and it continues commanding yos after the desired number has been passed.
max_depth:	The maximum depth allowed for the vehicle. If this abort is triggered, it goes into the harshest abort, with a release of the dropweight.

air_bladd:	This is set to 1 if an air_bladder exists and should be used on abort, otherwise it should be set to 0
abort_on_pressure:	If this is set to a value greater than 1, then the abort is triggered if there is no recorded change in the pressure reading for that amount of seconds.

Miscellaneous

How to get the UTM coordinates:

In the glider/geodetic directory, there should be a file called geo2utm.exe. Run this file (by double clicking on it) and a DOS shell will open and prompt you for the latitude and longitude. Use decimal degrees. It will calculate and print to screen the corresponding UTM coordinates.

IN CASE THIS FILE ISN'T THERE: In codewarrior, open the file geodetic.mcp and make it (F7). This will create the .exe file.

If you change a value in masterdata:

Run edit_struct.exe (by double-clicking on it in the cvs_glidercode/glider directory), then open the glider.mcp file in codewarrior (or you probably already have it running) and MAKE it again (F7). Make sure that the target is set to glider (that's the little window at the top of the glider window in codewarrior). Now you can post link and load the glider.bin file and your masterdata changes have been added.

Missing sensor in the parsed files:

If there is a variable missing in the parsed files, then it must be missing in the parse.cfg file. Open the parse.cfg file (it's a text file) and check the [all] config (ie the list of sensor names under the keyword [all]). You can add the sensor and reparse... you don't have to rerun the mission.

PreMission CheckList

Run md_test

check the vacuum:

```
g m_vacuum
ret
```

check the altimeter:

```
g m_altitude
ret
```

check the battery:

```
g m_battery
ret
```

Programming the mission

the communicate behavior should have a top_depth greater than the min_depth in glider_yo

ALWAYS have a mission timer in mission_abort, and as many abort watches as you can.

Downloading, Uploading, and Parsing

Downloading to the glider:

- make sure you're in glider options. (see: setup menu: setup: glider options)

In procomm:

- to send files:
  - delete any files with the same name as the one you want to send [<del filename>]
  - type "`yr`" (for y-modem receive)
  - click the 'send file' button on top
  - make sure it is in y-modem mode (small window on the right)
  - choose the file
- to see a file on the persistor, write "`type filename`"

Uploading from the glider:

- type "`ys filename`"
- click on the 'receive file' button
- if you were in glider options, then the file should be in the `glider/parse` directory

Parsing and viewing missions:

- One of the last statements printed at the end of a successful mission indicates the .obd file the mission data was printed to. These files are titled "`g+date+mission_number`" with the appropriate extension. Upload that .obd file from the persistor.

- In the parse directory there is a `parse.cfg` file that contains lists of sensors available for parsing. The list title is indicated by [brackets]. Check to see which, if any, of the lists contains the variables you are interested in ('glider' being the most general. 'basic' and 'all' are also available) or write up your own list of sensors you want parsed.

- To parse the sensors you want from that mission, go to the parse directory in a dos window and type "`parse -o listname gmissionnumber.obd`".
- If you want to fill in the empty data, then instead of –o type –fo.
- This will create a `gmissionnumber.m` and a `gmissionnumber.dat` file

- run the .m file in matlab by going to the parse directory and typing `gmissionnumber` without an extension
- Typing "review1" in matlab will run a basic plotting program.
- To access any of the sensors, use the construct: `data(:,sensor_name)`
- This means that to plot sensor y versus sensor x in red you would type:

```
        plot( data(:,sensor_x), data(:,sensor_y), 'r' )
```

- Use `help plot` to see more options

To fill in the 'missing' data on a preparsed (but not filled) dataset:

After parsing:

- run the .m file in matlab by going to the parse directory and typing `gmissionnumber` without an extension
- run filldata, by typing 'filldata' at the matlab prompt
- type ' save g99etc' without a suffix
- From subsequent data processing, to work with this filled data from this mission, type 'load g99etc' and then you can review the data as normal

Viewing tools available:

Review1: Produces the basic plots of a mission:

Actuators And Physical Manifestation:
Figure 1 : heading and battery roll
Figure 2 : depth and ballast

DYNAMIC CONTROL Checks
Figure 3 : heading rate and roll
Figure 4 : vehicle pitch and battery position

Figure 5 : vacuum and depth
Figure 6 : cycle number and altitude

Plotctd: creates the three CTD-relevant raw data plots: Time versus each of water conductivity (water_cond), temperature (water_temp), and raw CTD pressure (m_waterpressure)

Plothead( ): Essentially this is a function that is called as 'plothead(heading of interest)' where typically the heading of interest (in radians) is the commanded heading. This function takes the heading data, which is all positive values from 0 to 2pi, and centers it around this heading of interest.

Plotwhere: plots the two-dimentional position of the vehicle (where it thinks it went by dead reckoning, and updated by GPS)

Plotwhere3: plots the three-dimentional position of the vehicle: essentially the same as plotwhere, but with depth information. See its help file for information about altering the view angle.

All of these programs except for plothead have to be run after review1, as they rely on the timestamp data generated by that program. For any of these files, one can always use 'help filename' for more information about the file.