Three-Dimensional Tether Awareness Trainer

by

David H. Manowitz

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 21, 1999

June 1999

Author_____

Department of Electrical Engineering and Computer Science
May 21, 1999

Certified by_____

Nathaniel I. Durlach
Senior Research Scientist
Thesis Supervisor

Accepted by_____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Three-Dimensional Tether Awareness Trainer
by
David H. Manowitz

Submitted to the Department of Electrical Engineering and Computer Science

May 21, 1999

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

It is important that pilots of underwater remotely operated vehicles (ROVs) learn an awareness of the position of their tethers as they pilot their vehicles. Thus, a three-dimensional tether model that is computable in real time is a vital part of a computer-based training system for ROV pilots. This thesis presents such a tether model and a discussion of its implementation. Due to the constraint of real-time computation, a full physical model of the tether is not developed. However, this model attempts to capture two important behavioral components of a real tether. First, it computes the position of the tether when it is only acted upon by a current. Second, it determines the effects of interactions between the tether and other objects in the undersea environment. At least in these regards, a reasonable approximation can be made.

Thesis Supervisor: Nathaniel I. Durlach
Title: Senior Research Scientist, M.I.T. Research Laboratory of Electronics

Acknowledgments

I would like to thank a number of people for their help on this thesis. First and foremost, I would like to thank my supervisor Nat Durlach for his support and ideas. Many of the ideas for this thesis were proposed by Nat. I am also grateful to Stewart Harris and Jason Fritz of Imetrix, Inc. and Barbara Fletcher, formerly of Imetrix, and currently of the Space and Naval Warfare Systems Center. They were invaluable for their ideas and feedback on the tether model. They were my link between the simulation and the real world. Additionally, Nick Pioch and Bruce Roberts of BBN have provided me with support, ideas, and not least, conference rooms to meet with the rest of the team. I would like to thank Tom Wiegand, Matt Esch, and Jonathan Zalesky for their work last year, which gave me a very good starting point for this thesis. Last, but certainly not least, I am indebted to the Department of Naval Research for funding this project so that I would not be financially indebted to anyone else.

I also want to recognize my family and friends. Although they did not help directly with this work, they provided me with support and encouragement when I needed it. They also provided me with a break from my work when I needed it, and sometimes even when I did not.

Contents

Figures

1 Introduction

1.1 Background

Underwater remotely operated vehicles (ROVs) have become an important tool in the exploration of the seas. These tools have been recently brought to the public's attention through their role in exploring the sunken remains of the RMS *Titanic* in real life and in the movie *Titanic*. Before these events brought ROVs to the public's eye and once they have faded from it, the ROV has been and will continue to be a valuable tool in underwater reconnaissance, inspection, and exploration.

Piloting an ROV is a skill that takes a great deal of training. A pilot must be able to handle the vehicle even when it is being driven off course by currents and must be aware of the vehicle's surroundings. For the most part, training to pilot an ROV requires time piloting a real ROV; however, since there are not many vehicles available, and they are often needed for real missions, it can be hard to arrange for training. As a result of this difficulty, using a virtual environment (VE) simulator can be extremely beneficial for training. In addition to providing a simulation, a VE trainer can also incorporate intelligent tutoring to give guidance to a novice pilot. Using a VE trainer, a pilot-to-be can learn the various skills needed: maneuvering, sonar reading, and tether awareness, among others [4].

All ROVs are connected to their point of origin—usually a ship or dock—via a cable or tether that carries power and control signals to the vehicle, and video, sonar, and other signals from it. Since radio waves do not travel well through water, the tether is the main link between an ROV and its launching point. As the ROV travels underwater, the

tether is affected by currents and can become entangled around objects. Additionally, the pilot should be aware of the length of tether in the water: too much tether makes entanglements likely; too little prevents the pilot from suitable exploration. Thus, in training people to pilot ROVs, awareness of one's tether is an essential skill that must be acquired.

A pilot should have a general idea of where the tether is located and be sensitive to signs of a tether entanglement. Thus, a VE trainer should have a model of the tether available to determine the behavior of the tether. This model of the tether must conform to at least the gross characterization of the tether's behavior. In particular, the model should exhibit two main behaviors. First, it should bend in the direction of the current when the current velocity is greater than zero. Second, it should wrap around objects when the tether collides with an obstacle and the ROV does not move away from the obstacle. The models developed in this thesis and the previous work all conform to these behaviors. Alternate methods for determining tether position based on a tether instrumented with physical sensors are discussed in [2]. However, all the methods developed so far use an analytical model to predict the tether position.

1.2 Previous Work Performed

1.2.1 Outline of Work

Last year, our group developed a quasi-static model of a tether in open water [2] and a two dimensional model of a tether in a constrained environment [12]. The main concerns in designing the representation of the tether are faithfulness to the form of a real tether and the ability to make the necessary computations in real time. As one might

conjecture, it is not possible to get an exact representation of a real tether in real time or even near-real time, so the compromise implicit in the tether models is that they should be computable as fast as possible and should give an accurate depiction of the major features of the tether, such as general curvature and collision response.

1.2.2 Open-Water Model

In determining the model for the tether in open water, it is assumed that the motions of the ROV and tether are small compared to the velocity of the current, which is assumed to only have nonzero velocity in the standard Cartesian xy-plane. This is often a valid assumption, as ROVs are generally limited to relatively low speed. With this assumption, the tether is modeled as a static approximation that changes slightly every time-step of the simulation—a quasi-static approximation. Dynamic systems techniques can be used to construct a tether model, but, in general, these algorithms are too slow for the simulation.

Two classes of models studied to determine an appropriate one for the static approximation are finite-element models and assumed-equation models. The finite-element models attempt to determine the position of the tether by first dividing it into a number of small pieces. Then, given the tether's length and the forces acting upon each segment, the model iterates to find a solution that results in a force equilibrium. The assumed-equation models start by presuming that the tether will form a certain shape, and then, given the length of the tether and direction of the current, the model attempts to find values of the shape parameters such that the length of the tether will be as close to the

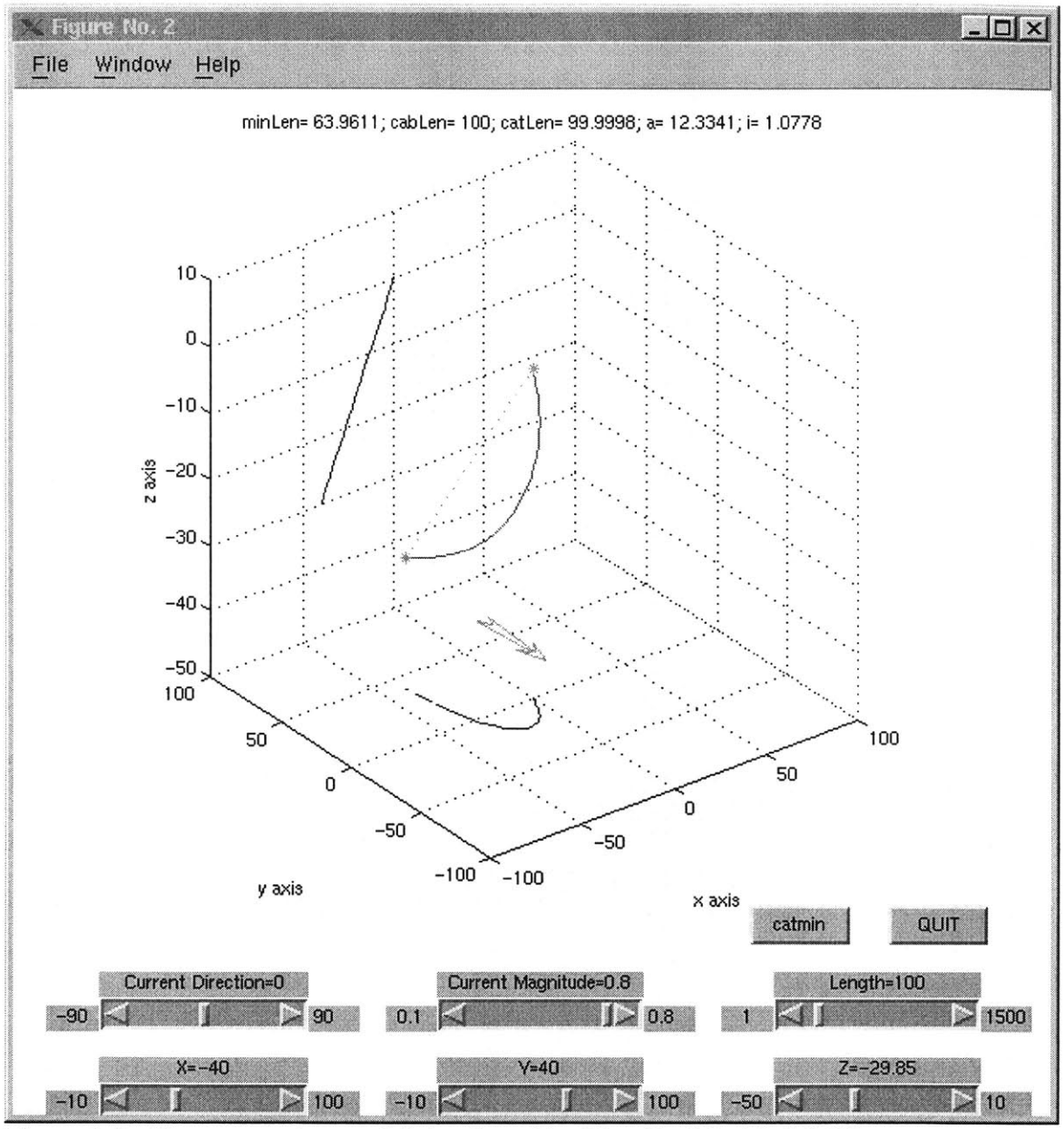

Figure 1-1: Catenary tether model developed by Matt Esch in Matlab [2]

given length as possible. It can be shown that a cable being acted upon by a constant force such as gravity or a uniform current will in fact form a catenary, with an equation of the form $f(x) = A\cosh\left(\frac{x - xs}{A}\right) + ys$ (for a two dimensional model), as illustrated in Figure 1-1. The catenary assumption is further confirmed by comparing results obtained

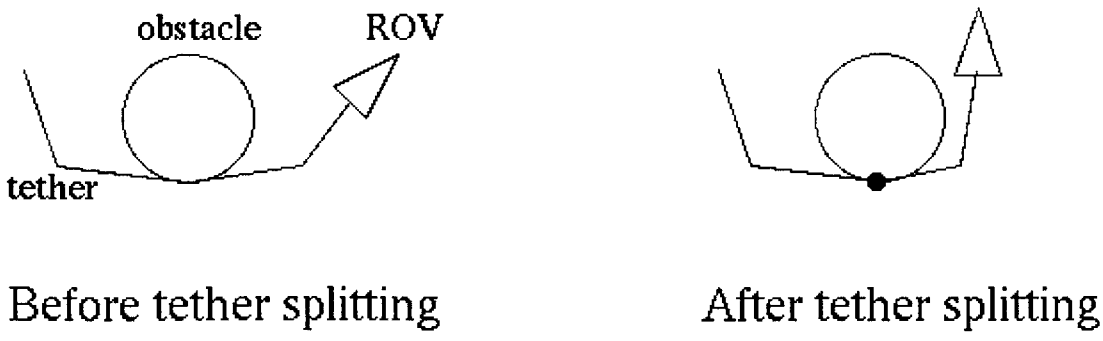

Figure 1-2: Illustration of tether splitting upon collision, showing point where first segment of tether is fixed in place.

with this model to results obtained with the finite-element model: the results are almost identical. Since the two models produce essentially the same results and the assumed-equation model is computable in many fewer operations than the finite-element model, the assumed-equation model has been chosen as the preferred static tether model [2].

1.2.3 Collision Model

The quasi-static tether model gives a good approximation to the shape of a tether in a current, but only when the tether is in open water. When there are other objects around with which the tether can become entangled, modifications must be made to the model. When the tether collides with an obstacle in the water, it is assumed that the tether breaks into two pieces, each like a separate tether, at the point of collision. However, only the piece between the ROV and the collision point can move; the other piece is fixed in place, as shown in Figure 1-2. This is not a realistic assumption; often the tether will slide once a collision has occurred, but implementing frictional effects can be quite complicated, especially when predicting the position of the tether without

information about the forces on segments of the tether, so they have not been included thus far.

Once collisions have occurred and the tether has begun to wrap around an object, the pilot should be able to free the vehicle by driving the vehicle back around the obstacle or obstacles on which the tether is caught. Thus, the simulation must allow for the tether to unwrap from these objects. This factor has also been incorporated into the tether model. When the angle between two segments (measured with normal towards the object wrapped around) is greater than 180 degrees, the tether "de-collides" at that point and the two segments are rejoined, thereby allowing the pilot to free the vehicle.

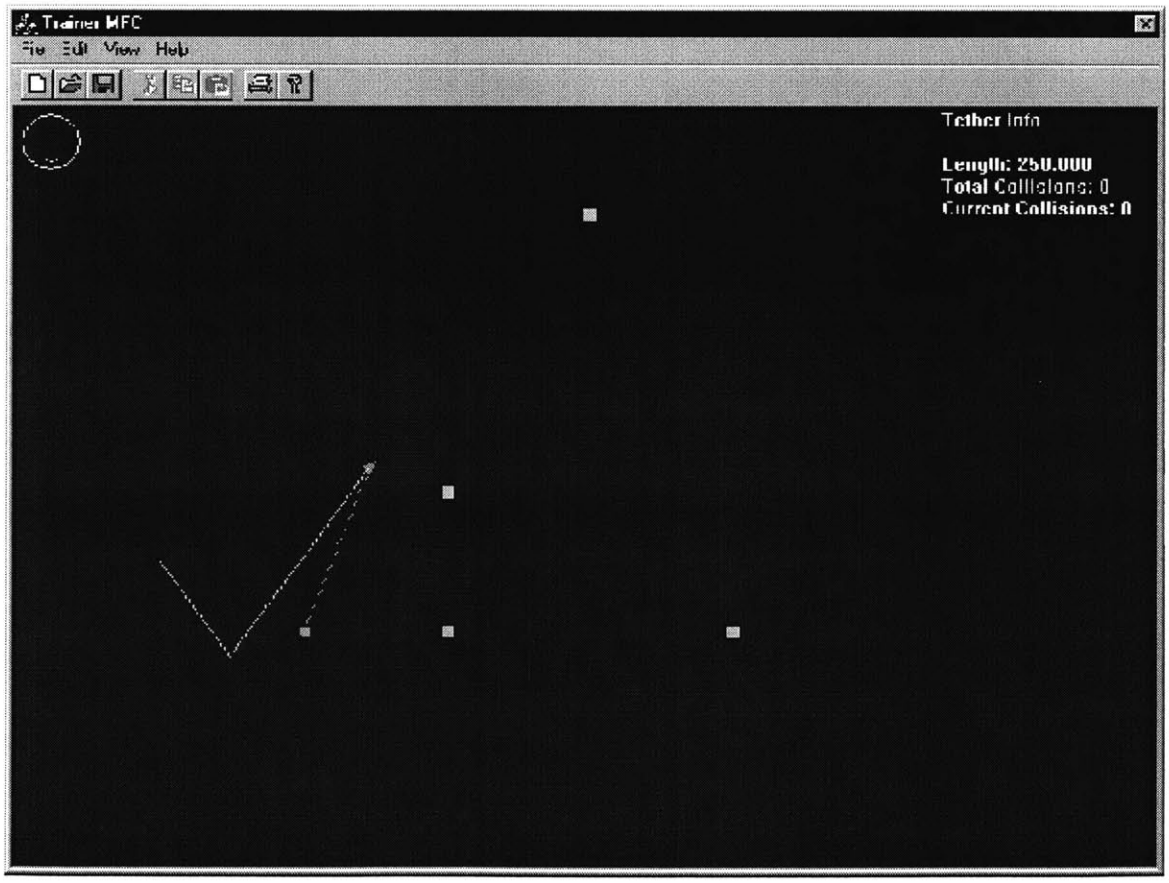

Figure 1-3: Two-dimensional tether trainer developed previously (showing V-shaped model)

Along with the model of the tether in an unconstrained environment, a trainer using a two dimensional model incorporating collisions was developed last year and is depicted in Figure 1-3. It embodies the tether-splitting and joining idea, but it restricts the tether collisions to specific bounding points predefined for each object in the simulation.

In addition to the catenary model described earlier, two simpler models can be used in this trainer. The simplest model is just a straight line between the starting point and the ROV, which is divided upon collisions like the other models. This model can be used when there is no current—a situation that may prevent the other models from converging to a solution—and is simple to use in debugging the collision responses. However, it is the most unrealistic in that it does not have a constant length. It can grow and shrink up to the point when the ROV reaches the tether length.. More realistic than the straight line model, but less so than the catenary model, is a V-shaped model. This model uses two straight lines to form a V-shape that is constrained to have the correct length. It can be thought of as a limiting case of the catenary. It gives the general shape that the tether will attain in a current given its length, yet it is still a very simple model to compute. The last type of model that can be used in this program is the catenary model, but the implementation developed last year tends to be unstable in that as the position of the vehicle changes, the predicted position of the tether sometimes changes drastically.

Finally, in this simulation, when the tether collides with an obstacle, the sub-tethers formed are taken to be the same shape as the single tether segment had been [12]. As a result, this implementation has been useful in providing a basis on which to build the three dimensional model, but it still lacks many of the features needed in the full implementation.

1.3 Design Overview

1.3.1 Project Goals

The overriding goal of this project has been to develop a three-dimensional graphical and analytical model of an ROV tether that approximates the behavior of a real tether and is computable in real time. Since some trade-off is needed between the computational complexity of the model and the degree to which it matches a real tether, two main features of a tether have been focused upon: the general shape in open water in the presence of a current and the wrapping of the tether around an obstacle when the ROV drives around the object. Since it is hard to determine exactly the position that a real tether forms as an ROV drives about, the accuracy of the model is judged by the opinions of experienced ROV pilots. Although experiments could be done in which a tether is instrumented with various types of sensors and measurements of its position can be made, no such experiments have been done as of yet (at least not to the author's knowledge), and performing such an experiment would be a considerable undertaking. As a result, only the general behavior of the tether can be measured, but it is felt that more detailed conformity may not be needed for substantial training benefit.

1.3.2 Thesis Goals

The original goal of this thesis was to extend the part task trainer to a three-dimensional model. However, in creating a full three-dimensional version of the model, it was determined that a number of the assumptions in the existing model were problematic in that they were too unrealistic. As a result, the goal of this thesis has shifted to creating a three-dimensional tether model that attempts to remove some of the

shortcomings of the previous version, with the eventual goal of incorporating the model into a part-task trainer.

1.3.3 Thesis Constraints

The main constraints of this tether model design are that it be computable in real time, that it be implemented in a portable fashion, and that it be easily integrated with the existing core of the TRANSoM system. The first constraint has been realized by implementing a quasi-static, rather than a fully dynamic design. Additionally, even though the catenary is treated as an equation model, it is implemented as a piecewise linear graphical form, rather than an implicit surface or other similar model. As a consequence of the second restriction, the design is implemented in such a way that it avoids reliance on features of the computer system on which it is implemented. Furthermore, the software is implemented in C++ using Open Inventor, V-Collide, and other libraries that are available on or easily ported to various computer systems [11, 6]. Lastly, to ensure that the model is easily added to the core testbed system, a small, well-defined interface to the tether model has been designed into the Tetherdisplay class and a few methods in the TTether3D class (see Appendix A for more detailed class reference). Since the existing TRANSoM core has been implemented using Scheme, rather than C++, the interface has been constructed so that the tether functions can be used either from another C++ program or from Scheme code by the use of the Header-to-Scheme program [4, 8]. Thus, these considerations are reflected in the implementation of the tether model.

1.4 Thesis Scope

There are two main components to the thesis work. First, the extension of the current tether model to three dimensions is discussed in chapter 2. Along with the new collision resolution system, problems with the existing collision model and possible solutions are described in chapter 3. Finally, in chapter 4 a summary of the thesis and several ideas for extending the work performed in the thesis are presented.

2 Three-Dimensional Conversion

2.1 Introduction

Although converting the tether model from two to three dimensions may seem like a trivial task, this is not necessarily the case. Since the straight line (rubber band) model only uses the starting point of the tether and ROV position as endpoints and can, therefore, change length as the ROV moves, it has been trivial to change it by adding a third coordinate to the model. However, both the V-shaped and catenary models use the starting point, ROV position, current direction, and tether length to determine the tether shape, so additional factors must be considered.

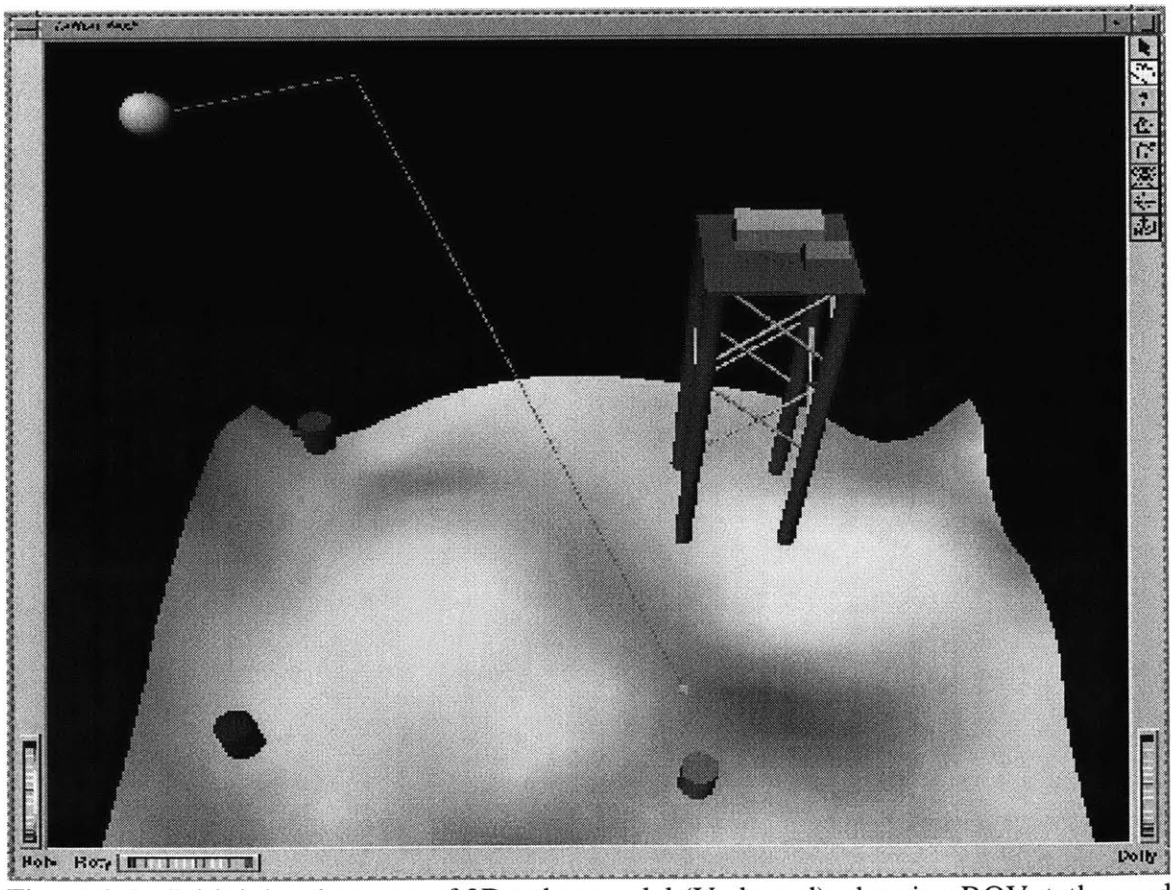

Figure 2-1: Initial development of 3D tether model (V-shaped), showing ROV, tether and environment. Tether diameter exaggerated for clarity.

2.2 V-Shaped Model

In two dimensions, the basic premise of the V-shaped model is to give an approximation to the shape the tether forms when it is effected by a current by forming a V-shape out of two links, given the length of the segment. The model begins by assuming the current is directed in the positive-Y direction using standard Cartesian axes, or the negative-Z direction using Inventor's axes. If this is not the case, all points in the segment are rotated through the angle that the current must be rotated through to align it as desired and rotated back at the end. The remaining steps in obtaining the parameters for the V-shaped model are illustrated in Figure 2-2 and described below.

First, the changes in position between the ROV and its starting point in the plane of the current, dx and dz in Inventor coordinates, are computed. Then, the angle θ that would be formed by the tether if it were fully extended to length L with change in the x-direction of dx, is determined (i.e. such that $dx = L \cdot \cos\theta$). Next, the remaining side of the right triangle formed by a leg of length dx and hypotenuse of length L is calculated;

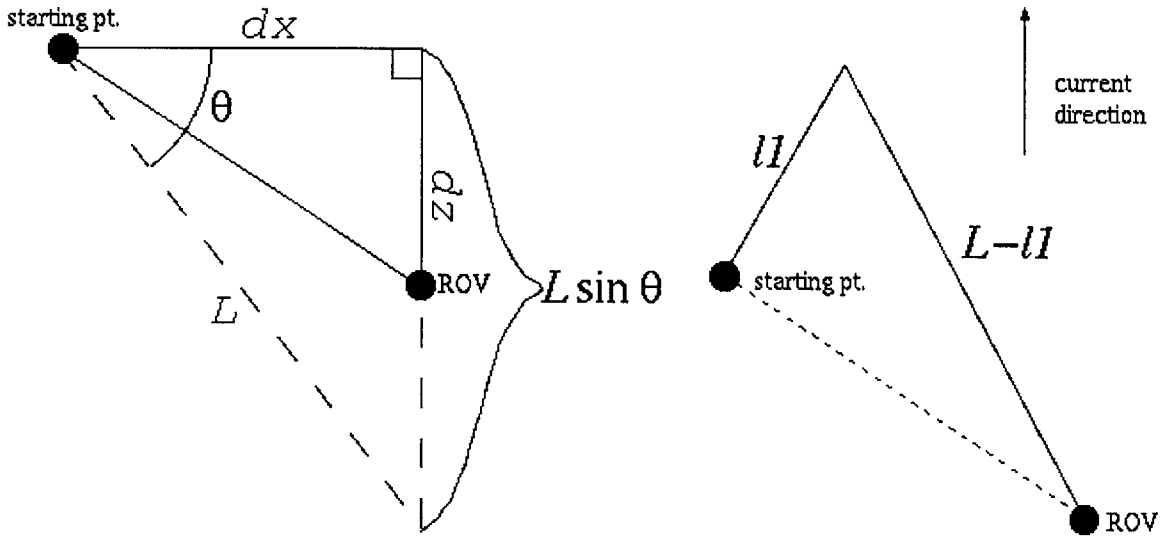

Figure 2-2: V-shaped model in two dimensions.

this leg has length $L \cdot \sin\theta$. The bend in the tether is effectively a measure of how much dz deviates from $L \cdot \sin\theta$. The length of tether from the starting point to the bend point is given by

$$ll = \frac{L}{2} - \frac{dz}{2\sin\theta} \,. \qquad \text{(Equation. 2-1)}$$

This equation can be derived by noting that when $dz = 0$, the tether should be bent exactly in the middle, so $ll = \frac{L}{2}$. Next, when the tether is fully extended against the current, $dz = L \cdot \sin\theta$, and ll should be 0. When the tether is fully extended with the current, $dz = -L \cdot \sin\theta$, and ll should be L. It is easily shown that equation 2-1 satisfies these three constraints and is a linear function of the ratio $\frac{dz}{L \cdot \sin\theta}$, as desired [12]. Thus, the tether correctly forms a V-shape given its length and the current's direction.

2.3 Catenary Model

2.3.1 Initial Conversion

In two dimensions, when a tether is subject to a uniform current and is allowed to come to equilibrium, it will naturally form a catenary: $f(x) = A\cosh\left(\frac{x - xs}{A}\right) + ys$.

Thus, it is natural to use this shape for the quasi-static model. Again, the two-dimensional model assumes that the current is flowing in the positive-Y direction using standard Cartesian axes, or the negative-Z direction using Inventor's axes. If not, the points are rotated as in the V-shaped model.

However, unlike the V-shaped model, the parameters in the catenary equation

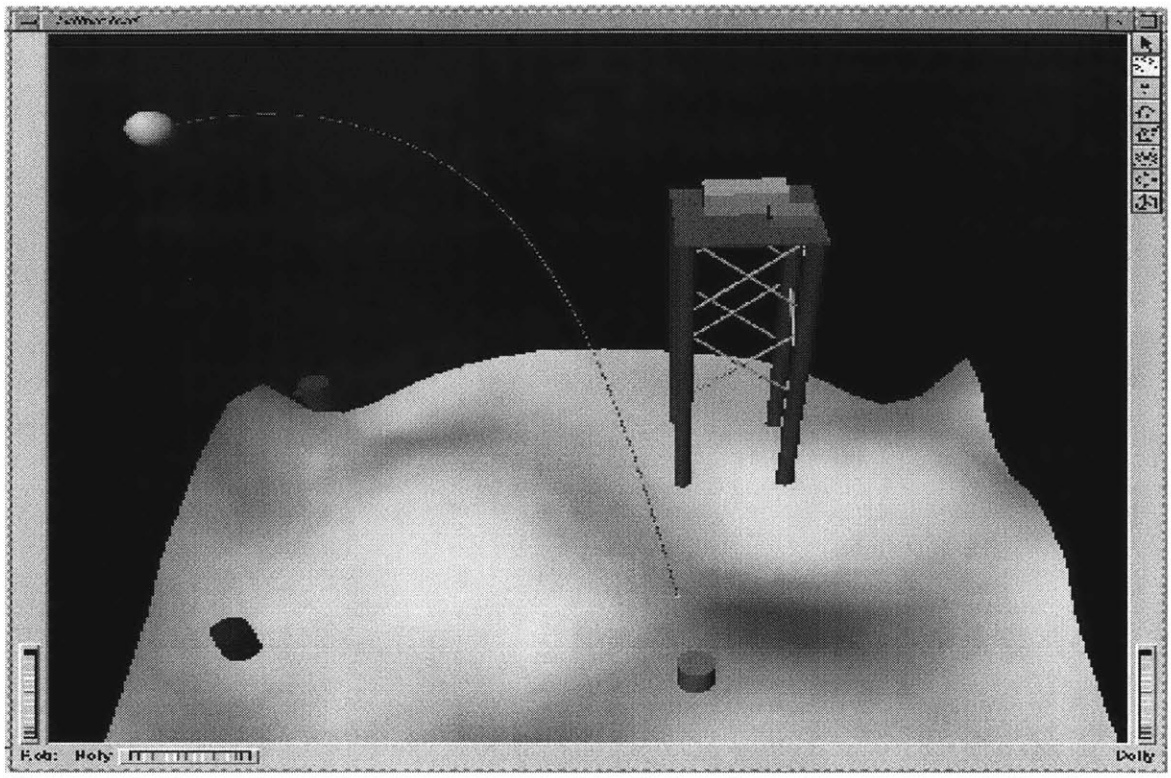
Figure 2-3: Three dimensional catenary model. Tether diameter exaggerated for clarity.

cannot be solved for directly. There are three of these parameters. First is the scaling factor, A. Next, the horizontal shift, xs is the horizontal point at which the catenary achieves its minimum value. Last, the vertical shift, ys, compensates for the vertical position of the end points, but it can be calculated from the other two. Since the scaling factor is both inside and outside the hyperbolic cosine, the parameters cannot be determined algebraically.

As a result, the parameters must be determined numerically. Let (x_0, y_0) be the starting point of the tether, and let (x_1, y_1) be the position of the ROV. The parameters are solved for by the minimization of the magnitude squared error function

$$e = e_1 + e_2, \qquad \text{(Equation 2-2)}$$

where $e_1 = \left(y_1 - A\cosh\left(\frac{x_1 - xs}{A}\right) + c \right)^2$, (Equation 2-3)

$$e_2 = \left(s - A\left(\sinh\left(\frac{x_1 - xs}{A}\right) - \sinh\left(\frac{x_0 - xs}{A}\right)\right)\right)^2, \quad \text{(Equation 2-4)}$$

s is the desired length of the tether, and $c = y_0 - A\cosh\left(\frac{x_0 - xs}{A}\right)$. Thus, the difference between the desired and actual endpoints and the desired and actual length is minimized by minimizing this function. Currently, the Nelder-Mead simplex model as implemented in Matlab version 5.2 is used to perform the minimization [2, 7].

2.3.2 Performance Enhancement

This function was initially tested only on tethers of short length (i.e. $s \approx 5$), but when tested on tethers of long length (i.e. $s \approx 100$), errors on the order of 10^{84} were being reduced to zero, taking a very long time! Thus, if y_1 is less than one, the relative error $e_1 = \left(1 - \frac{A\cosh\left(\frac{x_1 - xs}{A}\right) + c}{y_1} \right)^2$ is used instead of Equation 2-3 in Equation 2-2. Similarly, if s is less than one, $e_2 = \left(1 - \frac{A\left(\sinh\left(\frac{x_1 - xs}{A}\right) - \sinh\left(\frac{x_0 - xs}{A}\right)\right)}{s} \right)^2$ is used instead of Equation 2-4 in Equation 2-2. The same parameters are obtained, and the intermediate errors are much smaller than with the absolute calculations, resulting in a much faster convergence of the simplex method.

To further decrease computation time, the previous values of A and xs are used as initial guesses in the next minimization calculation, as in the Matlab version, but several additional steps have been taken to further speed up the computation. First, rather than use fixed values as the initial guesses for A and xs, the initial guesses are computed as a function of the length of the tether: $xs_{init} = \left(\left(3 \cdot \log_{10}(length) + \log_2(length)\right)\frac{1}{4}\right)^3$ and $A_{init} = \frac{1}{2} xs_{init}$. Although this function has been obtained by trial and error, it seems to give reasonable initial guesses, while not being very hard to calculate. An additional decrease in time is obtained by decreasing the error tolerance from 0.01% to 0.5%. These steps put together result in a speedup of two to five times over the original Matlab code, considering only the number of iterations of the minimization function—not even the speed increase in porting from Matlab to C++.

2.4 Depth Calculation

From this result, a three dimensional rendition needed to be constructed. Simply eliminating the depth coordinate before the calculation and adding it back in at the end would not produce the correct result.

2.4.1 Linear Depth Change

Originally, the depth of the tether was desired to be a linear function of the length along the tether: a point further on the tether should have a depth linearly related to points closer to the starting point. For the initial V-shaped model, the third dimension was ignored and ll was computed as in the two-dimensional version. However, the

coordinates of the bend point were altered so that the length restriction was satisfied in three dimensions. Letting the change in depth of the tether from the starting point to the ROV (or between collision points) be *dy*, *ll* and *L* were defined as above, with $ratio = \frac{ll}{L}$. If the difference between the x and z coordinates of the bend point (determined in two dimensions) and the starting point were dx_b and dz_b, respectively, and dx_b was held fixed while dz_b was redefined to $dz_b' = \sqrt{ll^2 - dx_b^2 - (ratio \cdot dy)^2}$, then the tether would form the correct shape with length *ll* between the starting and bend points and $dy_b = ratio \cdot dy$. A problem in implementing this algorithm was that roundoff errors could cause the tether's length to be reported as longer than it really was, so when computing *ratio*, the length of the tether was taken to be $0.95L$. Attempts to use a similar method for the catenary model were unsuccessful. However, it was later decided that a linear change in tether depth would not always be desirable, so a new method of computing the depth was needed.

2.4.2 Nonlinear Depth Change

To allow for nonlinear depth effects, a method originally implemented in Esch's catenary model has been implemented for both the V-shaped and catenary models [2]. Once the endpoints have been translated to move the start point to the origin and then rotated to aligned the current correctly, the tether undergoes two more rotations.

The first of these rotations compensates for the "sag factor." The sag factor is a coefficient between -1 and 1 which represents the tendency of the tether to sink (negative) or float (positive). The sag factor is currently defined as buoyancy times (1 - relative

current magnitude). At the moment, the buoyancy is another coefficient between -1 and 1, but it could be adjusted to reflect real units. The relative current magnitude is calculated by dividing the current magnitude by the maximum allowed current magnitude. The sag factor is determined in this manner because the tether has less of a tendency to sink or float in the presence of a strong current, as the current provides a lifting force, but in a weak current, the buoyancy of the tether is the primary cause of sinking or floating behavior. Once the sag factor is calculated, it is multiplied by $\frac{\pi}{2}$, and this angle is use to compute first rotation, performed around the x-axis (thus rotating in the yz-plane in Inventor or standard Cartesian coordinates).

The second of these rotations aligns the tether with the xz-plane in Inventor coordinates or the xy-plane in Cartesian coordinates. This enables the V-shape or catenary models to be run in two dimensions, where they are much easier to understand and implement. This rotation is set up by calculating the angle in the appropriate plane (xy for Inventor, xz for Cartesian) that the end point of the segment forms with the origin—just an inverse tangent calculation. Then the endpoints are rotated through the negative of this angle about the correct axis (z in Inventor, y in Cartesian) to result in the corresponding two dimensional model. Upon calculation of the model, the three inverse rotations are performed in reverse order, and the model assumes a correct three dimensional form, as illustrated in Figure 2-4. For more illustrations of the depth effects of the tether, see Appendix B.

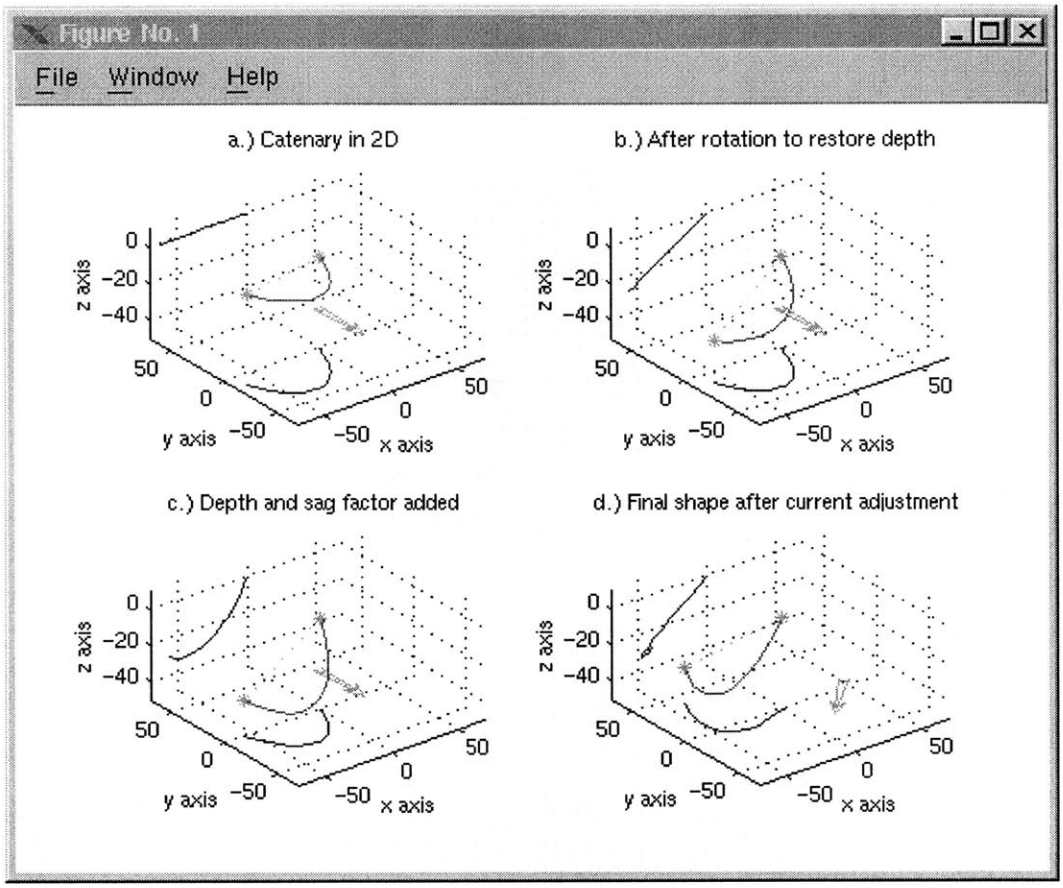

Figure 2-4: Tether after calculating 2D shape and following the three inverse rotations. Arrow shows the current direction. Shapes along edges are projections onto xy and xz planes.

2.5 Summary

With these models implemented in three dimensions, models have been developed that predict the shape of the tether in open water. According to ROV pilots, the models are realistic, given that a current has been acting on the tether for some time. However, this has only been part of the work in developing the full three-dimensional tether model. Improved collision detection and resolution is also needed.

3 Collision Resolution

3.1 Introduction

An important factor in piloting an ROV is being aware of tether entanglements. As a result, a trainer for ROV pilots should be able to simulate these entanglements, so it must have some way of determining when the tether has come into contact with another object. In the two-dimensional model, collision identification is a relatively simple task. All objects in the virtual world are preassigned bounding points that form a convex hull around them, and then the tether can only collide with these points. Since this constraint is too limiting in three dimensions, an improved method for detecting and reacting to collisions is needed. This new method is based on the V-Collide system from the University of North Carolina in Chapel Hill with some custom modifications. Using V-Collide, a tether collision with any point of another model can be determined correctly.

Additionally, once a tether collision has been identified, the tether model must react in a reasonable manner to the collision. The initial idea for the tether's response to a collision was to have the tether split into two segments of the same model type at the point of collision, and then the segment attached to the ROV would be allowed to move, but the other segment would be fixed in place (as though the contact had infinitely high friction or stickiness). However, after trying this method for a little while, it has become clear that this idea is too restrictive and under some conditions even leads to gross absurdities, such as the tether model becoming stuck far inside an object. Thus, some new concepts have been added to the collision resolution to fix the errors and to attempt to add some dynamic behavior to the collision.

Lastly, the tether model must provide some mechanism for allowing the tether to free itself from a collision. A collision is regarded as invalid, and the two segments on either side of the collision are rejoined, when the angle between the two segments is greater than or equal to 185 degrees. Originally, this number had been 180 degrees, but there were some problems with collisions becoming undone too quickly. With all these factors taken into account, the tether model should exhibit reasonable behavior upon collisions with objects in the environment.

3.2 Basis for Collision Detection: RAPID and V-Collide

3.2.1 RAPID

V-Collide is based on another system, also developed by the University of North Carolina, called RAPID, with modifications to improve its speed. RAPID uses a hierarchy of object-oriented bounding boxes to accurately and quickly determine if two objects overlap. Object-aligned bounding boxes are rectangular shapes that enclose all the vertices in an object. In contrast to axis-aligned bounding boxes, in which the axes of the boxes are aligned with the global coordinate axes, object-aligned boxes have their axes oriented along the dimensions of the objects they enclose. As a result of this property, an object-aligned box around an object will be no larger than an axis-aligned box around the same object. RAPID takes the coordinates of the vertices that comprise an object and computes a tree of bounding boxes, as shown in Figure 3-1. This tree extends from boxes that contain two triangles up to one that surrounds the entire object, and it is this tree that is used in the intersection computation [5].

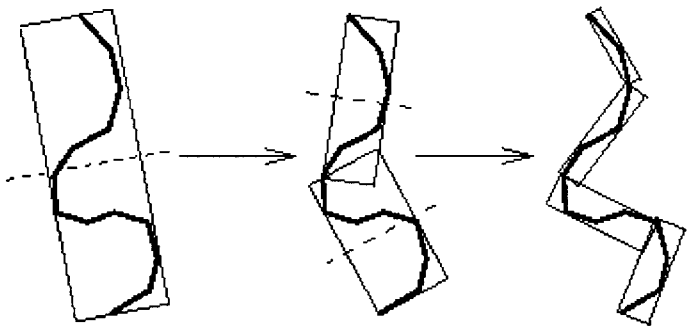

Figure 3-1: Creating object-oriented bounding box hierarchy by dividing and bounding groups of polygons in the object [5]

RAPID uses the hierarchical setup of the boxes to accelerate the collision detection. It checks to see if two objects intersect by checking if the object-level bounding boxes overlap. If not, the objects do not intersect. If so, then the tree of boxes is traversed until either it is determined that no lower-level boxes intersect, in which case the objects are separated, or until triangles in the object models are determined to be overlapping. In this manner, RAPID can give a list of intersecting pairs of triangles (each object contributing one triangle to the pair) [10].

3.2.2 V-Collide

V-Collide uses RAPID as its underlying collision-resolution engine. V-Collide has better performance than RAPID, though, because it only checks for collisions among those objects that are close together. Thus, pairs of objects that are far apart in the world do not contribute time to the collision detection, as they are never tested. As distributed by UNC, V-Collide only reports whether a pair of objects is colliding or not, rather than returning a list of all triangles in each model that are in contact. However, with a few modifications, V-Collide can be made to return lists of intersecting triangles for all pairs of colliding objects, just as RAPID does [6].

3.3 Extending the Collision Detection

3.3.1 Modifications to V-Collide

As mentioned above, several modifications to V-Collide have been made to enable the reporting of pairs of intersecting triangles. Additionally, several other methods are added for convenience. For a complete description of V-Collide functions, see the V-Collide user's manual, [6]. The main change to V-Collide is the addition of intersecting-triangle-pair reporting. As triangles are added to an object model, V-Collide numbers each triangle consecutively so that they each have unique ids. This can easily be extended to allow the object creator to specify the id of each triangle as it is added, but it was felt that sequential numbering would be most often used, so it has been implemented by default. Also, modifications have been made to the report structure generated by V-Collide when performing a collision query. In addition to relating the ids of impacting objects, two arrays of triangle pointers are returned, one for each object. The corresponding pairs in the lists are the triangles in each object that are in contact. To support this change, further modifications had to be made to a few of the other classes (such as PairData and NBody), but they are only minor.

Along with the report structure change, a few new methods have been appended to the VCollide (and underlying VInternal) class. Since the ROV simulation only tests for collisions between the ROV or tether and other objects, most pairs of objects should not be tested during the collision query. Thus, a method to deactivate all pairs of objects has been created. Furthermore, since the tether object is not treated as a rigid body—it changes shape every time-step of the simulation, it seemed unwieldy to create a new object for the tether every time-step. Additionally, since V-Collide gives every new

object a unique id, this could potentially exhaust the number of ids available, while most numbers would be wasted for just one time-step. Thus, a method has been implemented that takes an existing object and resets it by destroying its RAPID model, but keeping its id and other properties, such as current transformation. Finally, a method to check whether two objects collided in the most recent collision check has been added. These modifications to the V-Collide software itself enable more powerful collision resolution.

3.3.2 ivCollide

Besides the modifications to the V-Collide code, a wrapper class has been developed for easy use of Open Inventor models with V-Collide. This class is called ivCollide. This class has members to automatically add an Inventor scene graph to V-Collide as a single object and to update an object's transformation using an Inventor SoTransform. The conversion of an Inventor scene graph to a V-Collide object is based on a sample program from the Inventor distribution that decomposes an object into its primitive triangles and manipulates the triangles [9]. The transform alteration simply converts the transformation matrices from one format to another. As a result, this class makes interfacing between Open Inventor and V-Collide simple.

3.4 Collision Point Measurement

Unlike the relatively simple case of an ROV collision, where the only information needed is which object the ROV hit, a tether collision is more complicated. As the tether splits into two segments upon a collision, the point at which the collision occurs must be calculated. The basic idea of this algorithm is to use the pairs of colliding triangles to

determine the point of collision.

Once V-Collide determines that the tether has collided with another object, the pairs of intersecting triangles are returned. The triangles from the tether in the impact will be called the tether triangles, and the triangles from the other object will be called the other triangles. Lastly, it is assumed that each tether link (the part between two adjacent points) will be part of at most one collision, so only one link is examined at a time if more than one tether collision is found.

For each pair of triangles, each edge of the tether triangle in the pair is tested for intersection with the other triangle. This is computed via a test for intersection between a line segment (the edge of the tether triangle) and a plane (the plane of the other triangle). Let v_1, v_2, and v_3 be the vertices of the other triangle, and let R_1 and R_2 be the endpoints of the edge of the colliding tether link. Furthermore, let $\vec{u} = \vec{R}_2 - \vec{R}_1$, $\vec{L} = t\,\vec{u} + \vec{R}_1$ (where t

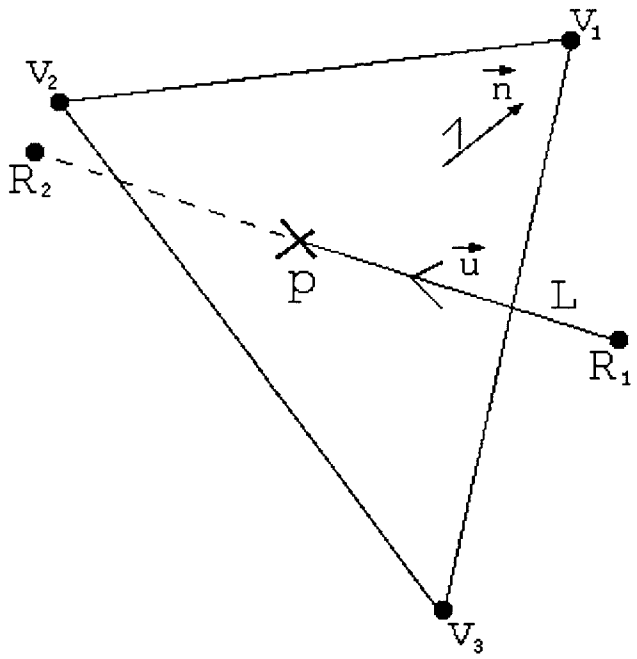

Figure 3-2: Line segment-triangle intersection test. The line segment is determined by points R_1 and R_2. The triangle has vertices v_1, v_2, and v_3.

ranges from 0 to 1), $\vec{s}_{21} = \overrightarrow{v_1 v_2} = \vec{v}_2 - \vec{v}_1$, $\vec{s}_{31} = \overrightarrow{v_1 v_3}$, $\vec{s}_{32} = \overrightarrow{v_2 v_3}$, and $\vec{n} = \vec{s}_{21} \times \vec{s}_{31}$, as shown in Figure 3-2, and let $\vec{x} = (x, y, z)$ be any point in the plane of the triangle. Arrows over variables indicate vectors or explicit treatment of points as vectors. By the equation of a plane, $\vec{n} \cdot (\vec{x} - \vec{v}_1) = 0$, but $\vec{x}$ must also lie on $\vec{L}$, so it must be true that $\vec{n} \cdot (t\,\vec{u} + \vec{R}_1 - \vec{v}_1) = 0$. After simple algebra to solve for *t*, it is found that $t = \dfrac{\vec{n} \cdot (\vec{v}_1 - \vec{R}_1)}{\vec{u} \cdot \vec{n}}$. This determines the intersection of the line with the plane of the triangle.

However, it is necessary to determine the intersection of the line segment with the triangle, which is not quite the same. First, this calculated *t* value is checked to see if it is within the range 0 to 1. Next, the point *p* is checked to see if it is actually within the bounds of the triangle by testing to see if it can be written as $\vec{p} = \alpha\,\vec{s}_{21} + \beta\,\vec{s}_{32}$, where α and β are both in the range 0 to 1. The actual code to perform this check does so by first projecting the triangle onto a plane and then performing the check [1]. Thus, if the calculated point passes these tests, it is considered to be the intersection point between that link of the tether and that other triangle. As long as the diameter of the tether is small, this is a reasonable calculation.

Once the program determines that a tether triangle edge has in fact intersected the other triangle, the intersection point is added together with the other intersection points for the link. Additionally, the normal vector to the other triangle is added together with the other collision normals for the link, and the plane of the other triangle is saved. Then the collision point and normal sums are divided by the number of points in the single link collision to get an average collision point and normal. Finally, the distance from the

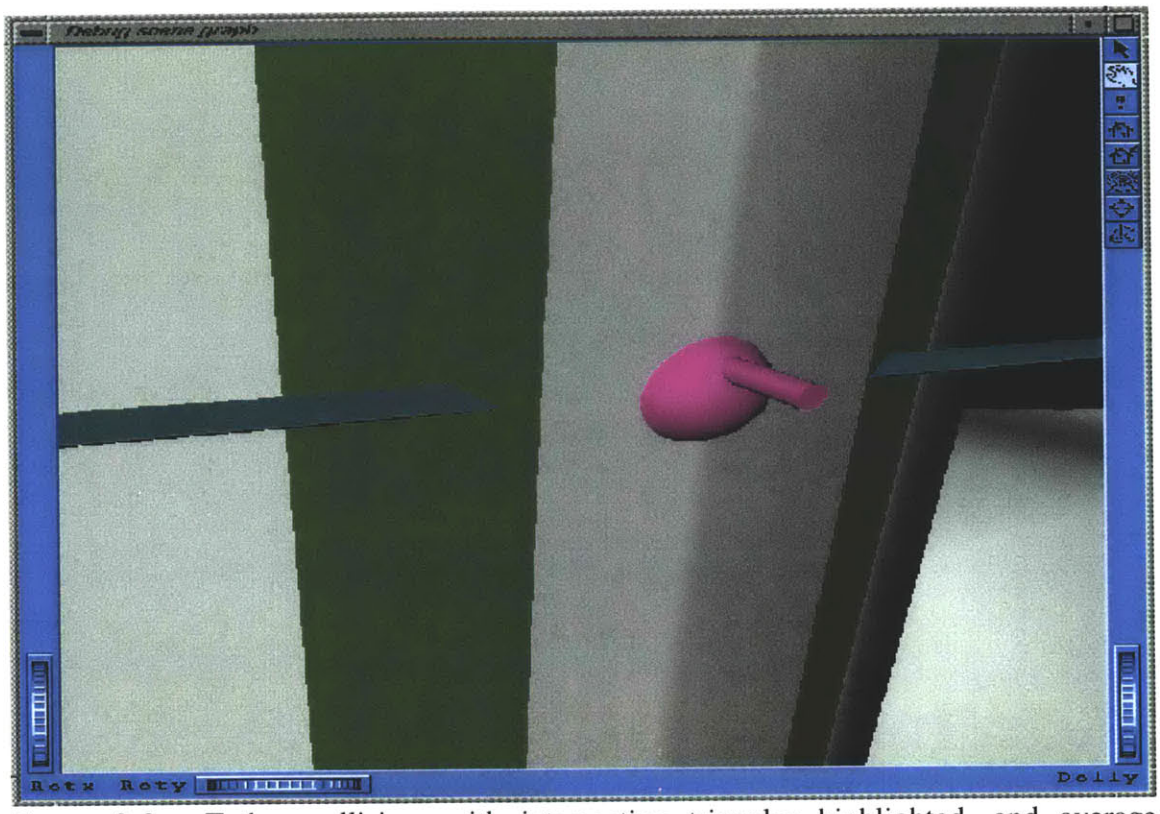

Figure 3-3: Tether collision, with intersecting triangles highlighted, and average collision point (sphere) and average collision normal (cylinder) shown.

Figure 3-4: Collision point showing where tether will be broken so as to avoid further collisions at this point.

average collision point $\bar{p}$ to each of the saved planes is computed using the distance formula $d = |\vec{n} \cdot \vec{\bar{p}} + D|$, where the equation of the plane is $\vec{n} \cdot (x, y, z) + D = 0$, and $\|\vec{n}\| = 1$. The maximum of these distances is calculated and the point of collision is moved this distance plus $\sqrt{2}$ times the radius of the tether along the average normal. This last step is taken so that when the tether is split at the collision point, the tether segments will not start out in a collision. At this point the tether is broken at the computed point, as described earlier.

3.5 Modifications to Collision Resolution

After the three-dimensional collision point computation had been worked out, the tether model still needed modifications. The collision resolution would cause errors sometimes, and ROV pilots felt that the collision response was too unrealistic at times.

3.5.1 Addressing the Errors

As mentioned previously, when the tether collided with an object, it split into two sub-tethers. The initial idea was to have the sub-tethers each be of the same type of model (i.e. both catenary or both V-shape). However, it was quickly learned that in two cases, this procedure could cause serious errors in the model. In one case, when the tether collided with the back side of an object (relative to the current direction), it might come unstuck right away. Also, when a catenary or V-shaped model was created at the point of collision, one or more of the segments adjacent to that point might immediately collide with the object. Since the tether would be taut for the first collision, such incidents

generally did not occur then (although due to roundoff error and similar factors, they sometimes would). They tended to occur on later collisions. The errors were fixable, although the fixes created problems of their own.

The first problem occurs when the tether collides with the back side of an object, relative to the direction of the current. Splitting the tether into two V-shaped or catenary segments will cause an error due to the attempt to unwrap the tether after the split occurs. After splitting, the current will cause the tether to form two segments as illustrated in Figure 3-5. Since the angle between the two segments is much greater than 180 degrees, on the next time step the tether will rejoin at the collision point, and then the next collision will be detected incorrectly. In such a scenario, as the ROV moves, successive tether collisions will be detected and then immediately rejoin further and further into the object until finally the tether gets stuck in the object, and the simulation gives unpredictable results.

In fact, such an action should never happen, as the current could not effect the tether on the back side of the object in this manner because the object would shield the

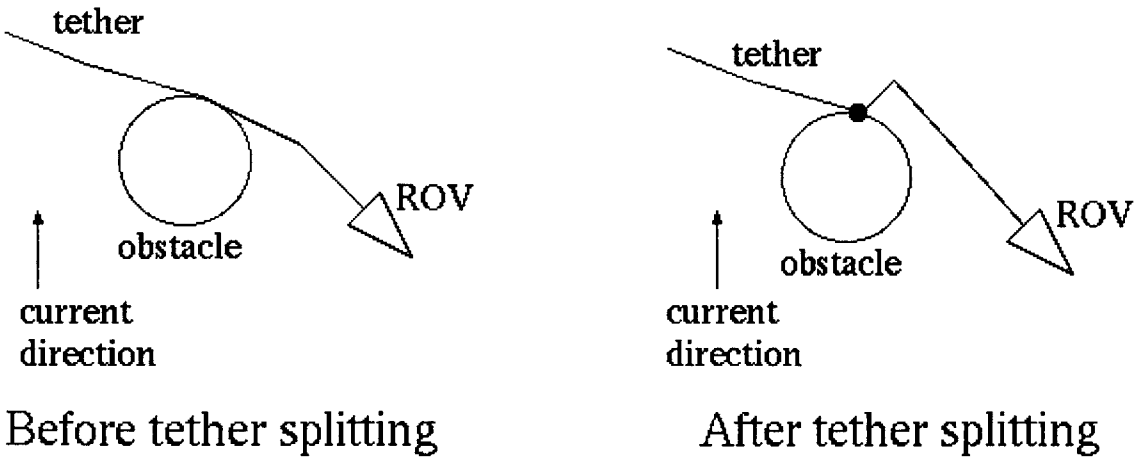

Figure 3-5: Incorrect tether splitting due to "back side" collision. This collision would be undone in the next time-step.

tether from the current, neglecting turbulence. Consequently, the second segment of the tether should be a straight-line segment. Thus, after a collision occurs, the model is tested to see if it would rejoin right away, and if so, the second model is changed to a rubber band one and recalculated.

The other case in which symmetrical tether splitting causes a problem is when one or both of the segments resulting from a collision would now cause another collision, as shown in Figure 3-6. When the model predicts this type of situation, the next tether collision will be detected far inside the object. The collision analysis in the tether model assumes that collisions will be detected within close proximity to the surface of the object that the tether has run into. Also, it presumes that the tether is nearly perpendicular to the surface of the object when the collision occurs. As long as the ROV and tether move slowly, this conjecture is a reasonable one to make. However, when the tether model jumps and violates this supposition, then the collisions response is unrealistic.

To correct for this discrepancy, after a collision is registered and the adjoining segment models have been run, another collision check is run. If a collision is detected

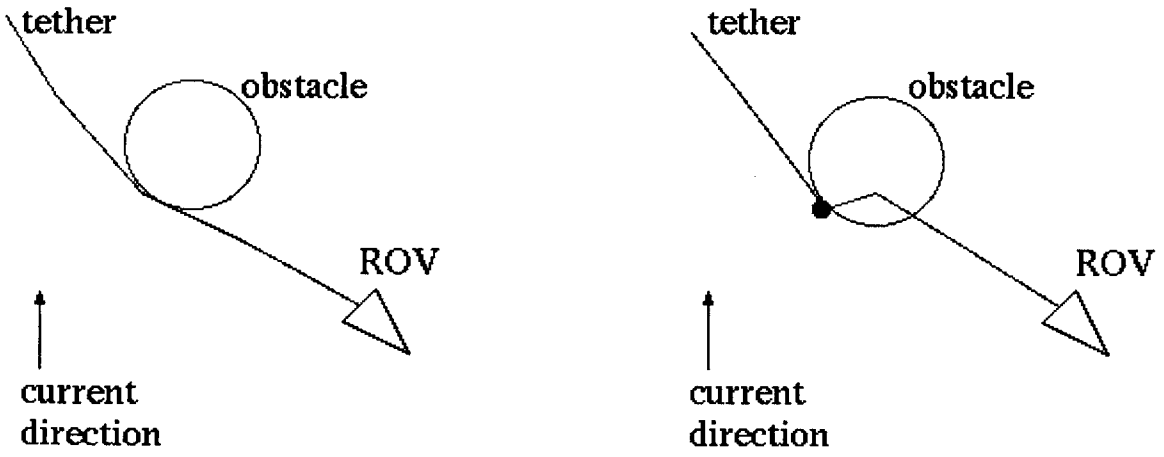

Figure 3-6: Tether splitting that would immediately cause another collision

this time, one of the two segments is recomputed as a rubber band model. If no collision is detected this time, the segment just changed must have been responsible for the collision. If another collision is detected, the other segment or both are responsible for the collision, and another collision check is made. Once the segment or segments responsible for the new collision have been determined, the model type of the responsible segment or segments is changed to rubber band, and the original collision resolution ends.

This procedure assumes that changing a model type to rubber band will fix the problem. That assumption may not be realistic, but so far it has always worked. This is likely due to the fact that when such a situation arises, the average collision point may be too far inside the object for an accurate response, but, presumably, there is a direct line between the collision point and the ROV.

Both of these problems are fixed by changing one or more tether segments from catenary or V-shaped models to rubber band models. However, since the rubber band model can change length, it is not a very realistic model. Furthermore, it can cause other problems. For example, consider a collision that resolves into a segment which causes another collision, so that the last segment is made into a rubber band model (as would happen to the collision in Figure 3-6). If the ROV flies toward the obstacle and then back to the left, so that the collision is invalidated, then sometimes the rejoined segment will be created through the obstacle. This problem is partially solved by only allowing the segments adjoining the collision point to rejoin when the angle between them is significantly greater than 180 degrees; currently 185 degrees is used. However, the fundamental problem is that using the rubber band model is very unrealistic, yet so are other alternatives.

3.5.2 Addressing the Realism Problems

Since the rubber band model is unrealistic, one may ask why it is used at all. However, in some cases, such as acting as a temporary model while the tether is wrapping around an object, the rubber band model functions well. Additionally, always using catenary or V-shaped models is unrealistic as well. One of the fundamental assumptions of the tether model is that between collision points the tether acts as if a current has been acting on it for a long time, so that the forces acting on the tether will come to equilibrium. On the other hand, when a collision occurs, it is not feasible for the tether to immediately form a catenary shape, but as the model currently exists, it does not take into account much dynamic behavior. For example, in a real ROV system, the end of the tether closer to the vehicle will move directly with the vehicle, but tether further back will take some time before it begins to move. As a result, a model that adds some dynamic adjustment to the collision response, while maintaining the quasi-static overall model, has been developed to address these concerns.

The approach that has been taken to resolve these issues is to allow the tether to slide along the point of collision. When sliding is allowed and the tether collides with an object, it is almost as if there is no friction at the point of collision in that the length of the tether is kept constant by allowing slack to move freely between the two segments at the collision point, as illustrated in Figure 3-7. Since there is negligible friction at the point of collision in this plan, the tether forms only one bend: the longest segment forms a catenary or V-shape, while the other segments are treated as rubber band models. However, the total length of the tether is kept constant.

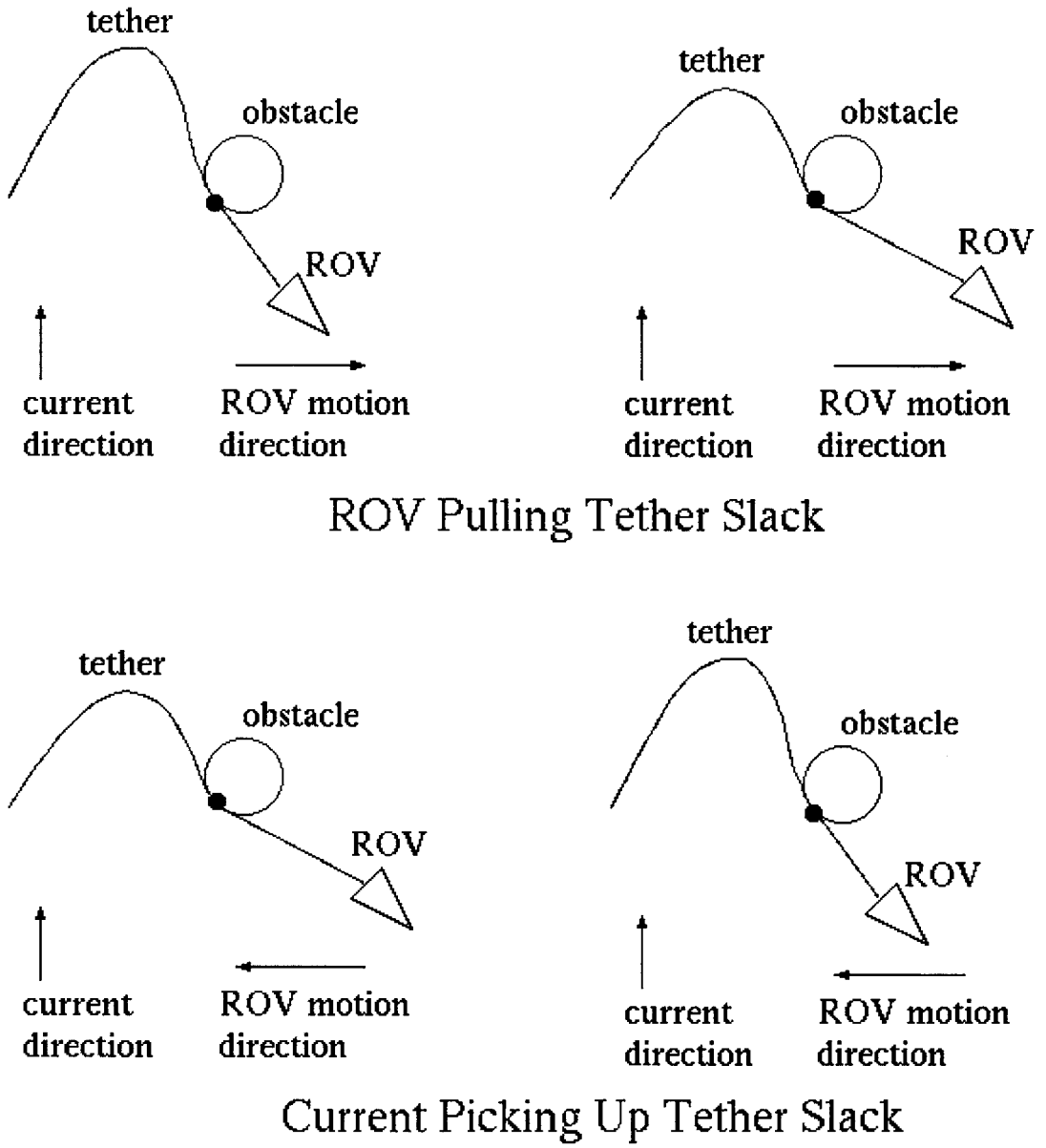

Figure 3-7: Tether sliding along point of collision

Assuming that the catenary or V-shaped segment is farthest from the ROV, this constraint is maintained by ensuring that the last rubber band segment—which is the only one that can move—is only as long as the actual distance between the ROV and the collision point. As the ROV moves away from the point of collision, the rubber band segment is extended, while the length of the bent segment shrinks. This transform only

takes place if the bent segment is not actually taut already; if so, the ROV is not allowed to move. In real life, often the ROV is able to pull the tether along the object the tether has collided with, so this extension to the existing model is reasonable.

In addition to this change, a similar modification has been made so that when the ROV moves closer to the point of collision, the length of the rubber band segment is shortened, while the bent segment is lengthened. This scenario is not quite as realistic as the ROV pulling the tether. However, if there is little friction at the point of collision and the current is strong, then it may be able to pick up this slack.

Although it is unrealistic to assume that the tether gets stuck to an object as soon as they collide, it is also implausible to believe that friction between the tether and other object is insignificant. Therefore, the model has been modified yet again. The tether is assumed to be in one of two states. In one state, there is no friction between the tether an the object, so the tether will slide. In the other state, there is infinite friction between the tether and the other object, so the tether is stuck. Additionally, when the tether is stuck, attempting to pay out tether (put more tether in the water) will not have any results. In real life, it is common for extra tether to have no effect when the ROV is truly stuck, so not letting out any more tether simulates this effect. To switch between the states, two criteria have been investigated. One criterion is that the tether becomes stuck after a certain number of collisions have occurred. The other criterion is that the tether becomes stuck after the angle between the normal vectors of the first and last collision exceed a given angle. For the most part, the angle criterion appears to be a better measure of when the tether should be stuck, but it is not perfect, so in some cases it may be augmented with

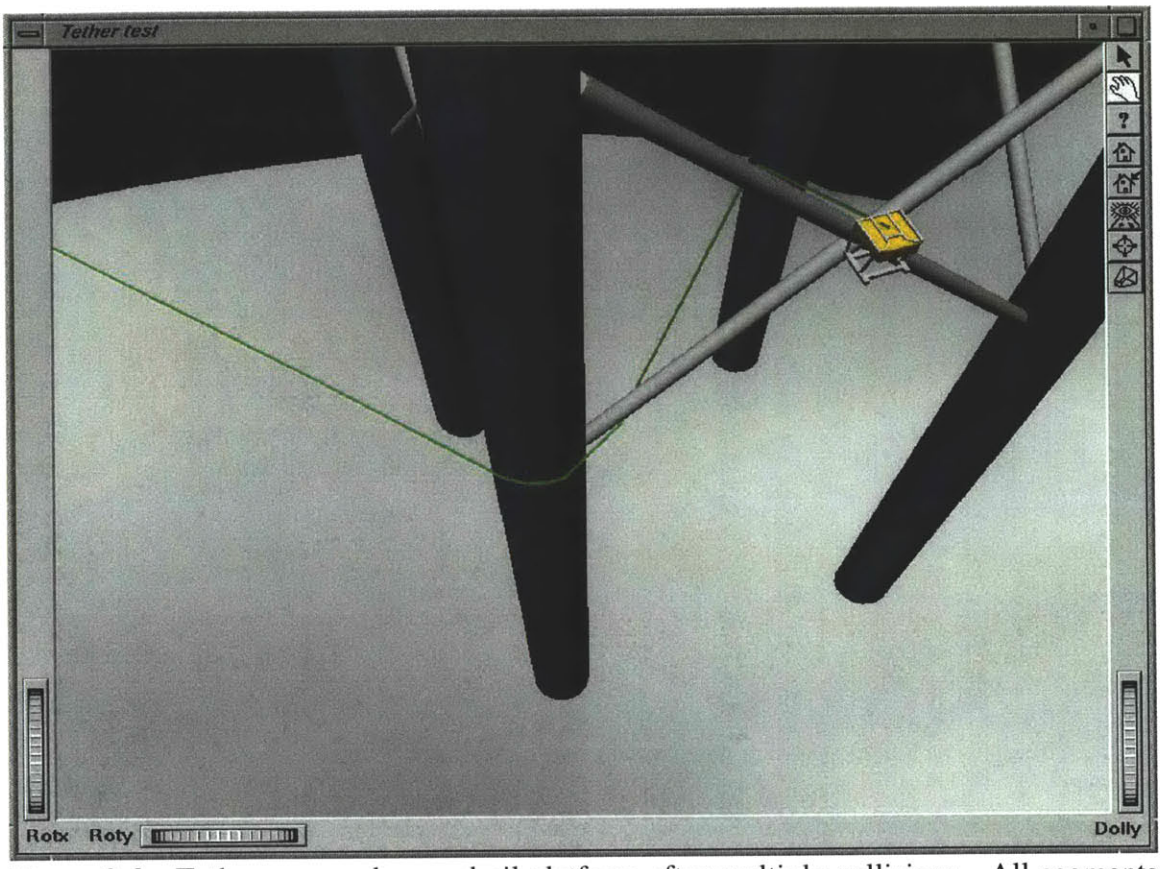

Figure 3-8: Tether wrapped around oil platform after multiple collisions. All segments after initial (on left side) are rubber band.

a check for the number of collision points. With these last modifications, the model approaches the behavior of a dynamic model, but it still does not achieve that.

3.5 Summary

With this improved collision detection, tether collisions are now accurately reported with any point of any object in the world. The initial method of breaking the tether upon a collision had been to make two identical-type segments. However, this method results in tether models that give errors or are unrealistic. Attempts to address this problem by allowing some segments to use the rubber band model have been relatively successful. Further efforts to increase the realism of the collision model by

allowing the tether to slide along the point of collision have also been rather successful, but a full dynamic model is really needed for realism.

4 Conclusions

4.1 Synopsis

Modeling the tether of an underwater ROV has been a complicated task. Despite this difficulty, it is important that ROV pilots be aware of the location of their tethers, so some kind of a reasonable tether model must be developed for use in a training simulator. Initial work on such a model began last year, with the creation of a three-dimensional model for use in the absence of collisions with other objects and a two-dimensional collision response. Nonetheless, a design that could react to collisions in three dimensions was needed.

As detailed in chapter 2, the conversion of the open-water V-shaped and catenary models was largely a matter of transformations. The three-dimensional problem was transformed to an equivalent two-dimensional problem. This two-dimensional problem was solved analytically for the V-shaped model or numerically for the catenary model. Upon completion of these calculations, the model was transformed back to three dimensions. Additionally, several parts of the catenary model were changed to make the computation more efficient. With these modifications, the three-dimensional conversion was completed.

Chapter 3 described the changes necessary in updating the collision detection and resolution. First, the existing collision detection scheme, in which predefined bounding points marked where the tether could collide with other objects, had to be replaced with a much less restrictive system that allowed the tether to collide with any other object at any point. Next, the response of the model to the collisions needed modifications so that

tether segments were allowed to have different model types. In some cases, this change was necessary to correct errors. Unfortunately, using the variable-length rubber band model exacerbated some problems with unrealistic behavior in the tether collision response. As a result, an attempt was made to keep the tether length constant, regardless of the type of model used. With these alterations in place, a less restrictive and more realistic collision response has been created, but some shortcomings still exist.

4.2 Future Work

There are two main directions in which the work presented in this thesis can be continued. First, the existing model could use further validation and verification. Second, the model can be improved by creating a full dynamic model, rather than the quasi-static model that is used currently.

4.2.1 Validation and Verification

So far, the performance of the tether model has been based upon the judgment of expert ROV pilots and some mechanical theory. Although they can provide some guidelines for analyzing the behavior of the tether model, more detailed metrics are needed in two regards. Initially, one may consider trying to measure the deviation of the tether model from the physical behavior of a real tether. For example, a real ROV can move through a predetermined course in a given environment, and the position of the tether can be recorded. Then, the same ROV movements can be fed to the simulation, and the difference in tether positions can be determined. However, a number of challenges face such a measure. First, measuring the position of the tether, even at a

small number of points along its length can be a daunting task. Perhaps the tether could be instrumented with a set of acoustic transponders, and receivers on the surface of the water could pick up the signals. Unfortunately, to get measurements of enough points on the tether to make an accurate comparison, the signals may cause too much interference for them to be picked up by the receivers. Secondly, the simulator is a much more controlled environment than the real ocean, so it may be hard to determine how exactly the tether was being acted upon by the current. Thus, it is not likely that a detailed measurement of real tether behavior will occur (at least not soon).

Since the tether model is ultimately intended to be used to help pilots increase their awareness of a real tether's position, a better metric may be to measure the performance benefit from using the tether model in a simulator. Such a benefit could be measured by having a group of people perform a situational awareness test using a real ROV after training with the real ROV, while another group takes the same test after training with the simulated ROV and tether, and a third group could take the test without training. A similar test has been conducted to test the effectiveness of the intelligent tutoring system of the TRANSoM system [3]. If the simulated tether model shows enough training benefit, then despite its sometimes unrealistic behavior, it can be considered useful for its intended purpose.

Clearly, much work remains to be done to determine the adequacy of the tether model. Since the goal is not to be the most accurate tether model, but to be a reasonable model for training use, measurements of its adherence to a physical tether and its usefulness for training are both relevant.

4.2.2 Dynamic Modeling

As mentioned in the previous section, the tether model as it stands may be realistic enough to function as a training aid. However, if it is determined that this is not so, or if a more realistic model is desired for some other tether modeling application, a fully dynamic model is needed. This type of model begins by treating the tether as a large number of small, connected segments. It then determines the position of the tether by considering the effect of the forces, such as the current and ROV motion, at the points on the tether where they are applied. Once each segment has been affected by some force, the effects are propagated throughout the tether until an equilibrium is reached. Thus, one part of the tether may move differently than others during a given time-step.

During the initial investigations into the tether model, a finite element model of the tether in a steady current had been examined, and it ran too slowly to be used in the simulation [2]. Although it was not the same as a dynamic model of the tether, the computation requirements were similar. Both treated the tether as numerous small segments and operated on all the pieces iteratively until an equilibrium was found. Thus, it seemed as if a dynamic model of the tether would also be very slow, as well as harder to formulate.

On the other hand, the limits of the quasi-static model have been reached. It has provided a decent idea of the tether shape in open water, but once the tether starts to wrap around objects, important assumptions are being violated. In the quasi-static model, it is assumed that the current has been acting on the tether for a while at each time step. Even in the absence of tether collisions, as the ROV moves around, this assumption is being violated. However, the shape of the tether is not changing by much. On the other hand,

when the tether is wrapping around an object, the shape changes drastically. This is why many of the unrealistic behaviors are seen.

As a result, if further extensions to the tether model are to be made, a dynamic model must be used to achieve realistic behavior. With the increasing speed of computers, the computation time may not even be as much of a bottleneck as it had previously been.

Appendix A: Class Structure

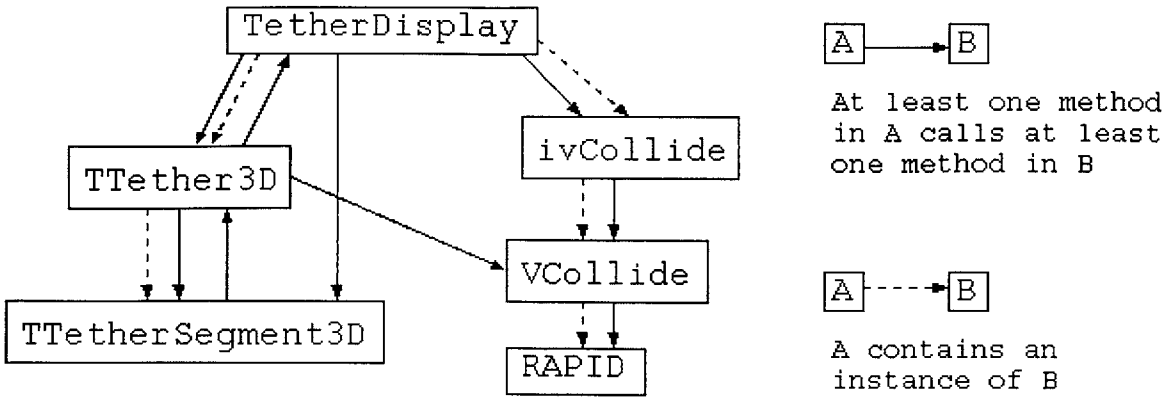

Figure A-1: Major classes that comprise the tether model and collision detection, and their relationships

- TetherDisplay: This class maintains the graphical tether model. It is the only class that directly modifies the Inventor scene graph of the tether model, via the update method. It is also the only class that changes the VCollide representation of the tether. After checking for collisions using VCollide, the resolveCollisions method is used to check for and respond to tether collisions.
- TTether3D: This class represents the analytical tether model as a whole. It has methods to set and get various properties of the tether model. It also has the RunModel method that is called by the application to update the tether model after changing some feature such as the end point. During collision response checking, it will update the graphical tether model and check for immediate collisions using VCollide.
- TTetherSegment3D: This class models a segment of tether, which is a piece between two collision points or an end point and a collision point. It is responsible for computing the position of the points within the segment, based on the type of model used. This information is extracted by the TetherDisplay class to

compute the locations of the vertices of the graphical model. When the RunModel method in the TTether3D class is invoked, the local RunModel method is called for every tether segment that has changed since the last time-step.

- ivCollide: This class is the interface for easily adding Open Inventor scene graphs to VCollide. It includes the addInventorObj method to add an Inventor object to VCollide or to modify an existing one. Also, there is the updateTransform method, which updates the VCollide transformation from an Inventor SoTransform.
- VCollide: The main collision detection library. Modified version includes methods to add, modify, or remove objects in the database. Also, has method to perform collision test and to get reports of results. For more information, see [6].
- RAPID: The basis for VCollide. No methods are used by tether model; only data structures are. See [10] for more information.

Appendix B: More Tether Illustrations

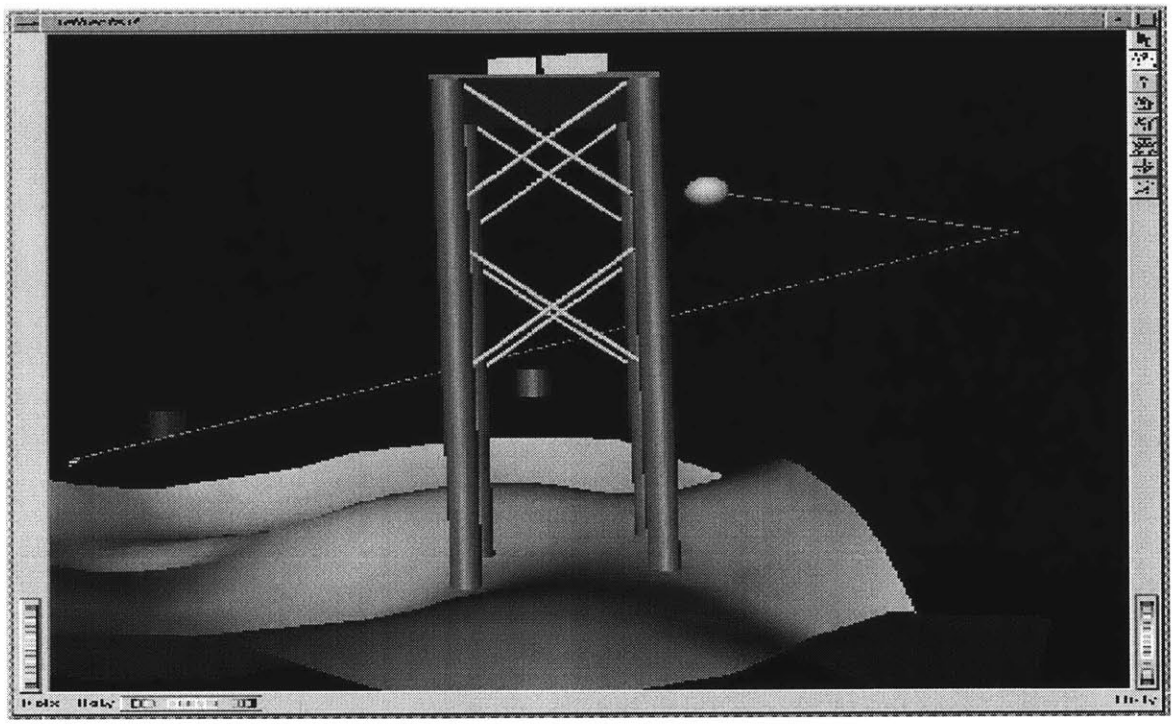

Figure B-1: V-shaped model showing linear depth change.

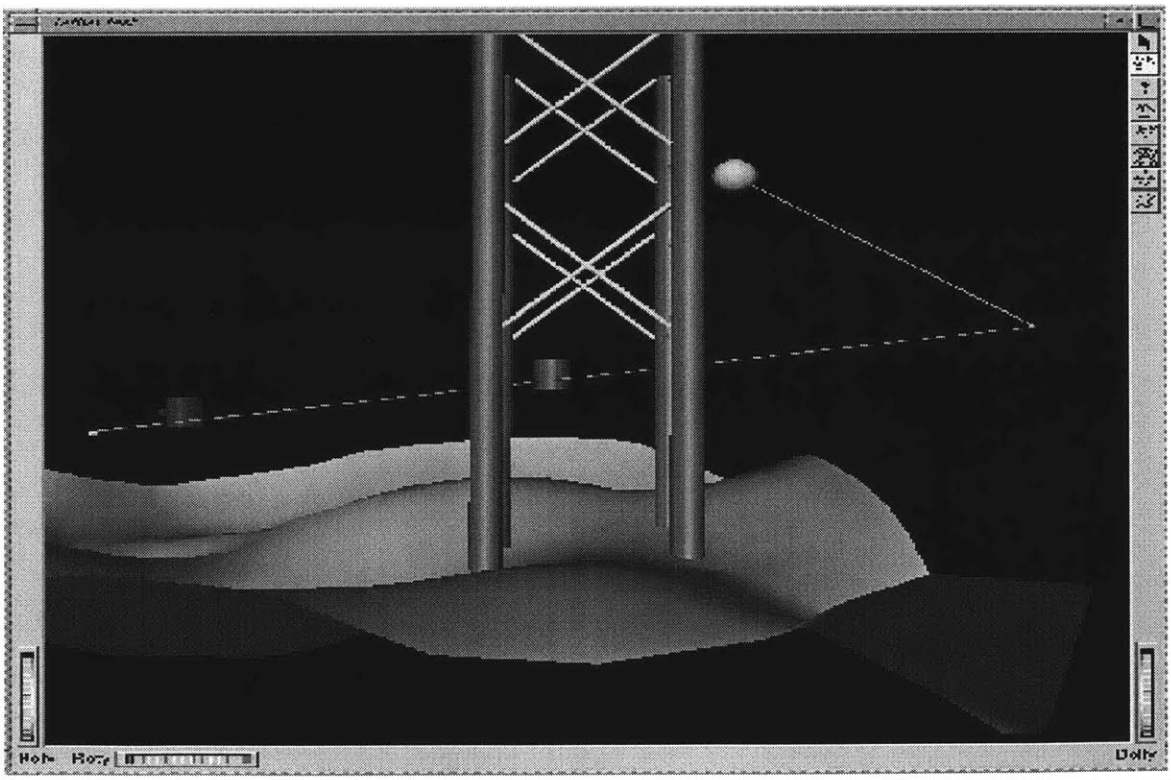

Figure B-2: V-shaped model showing nonlinear depth change: the first link has a steeper slope than the second.

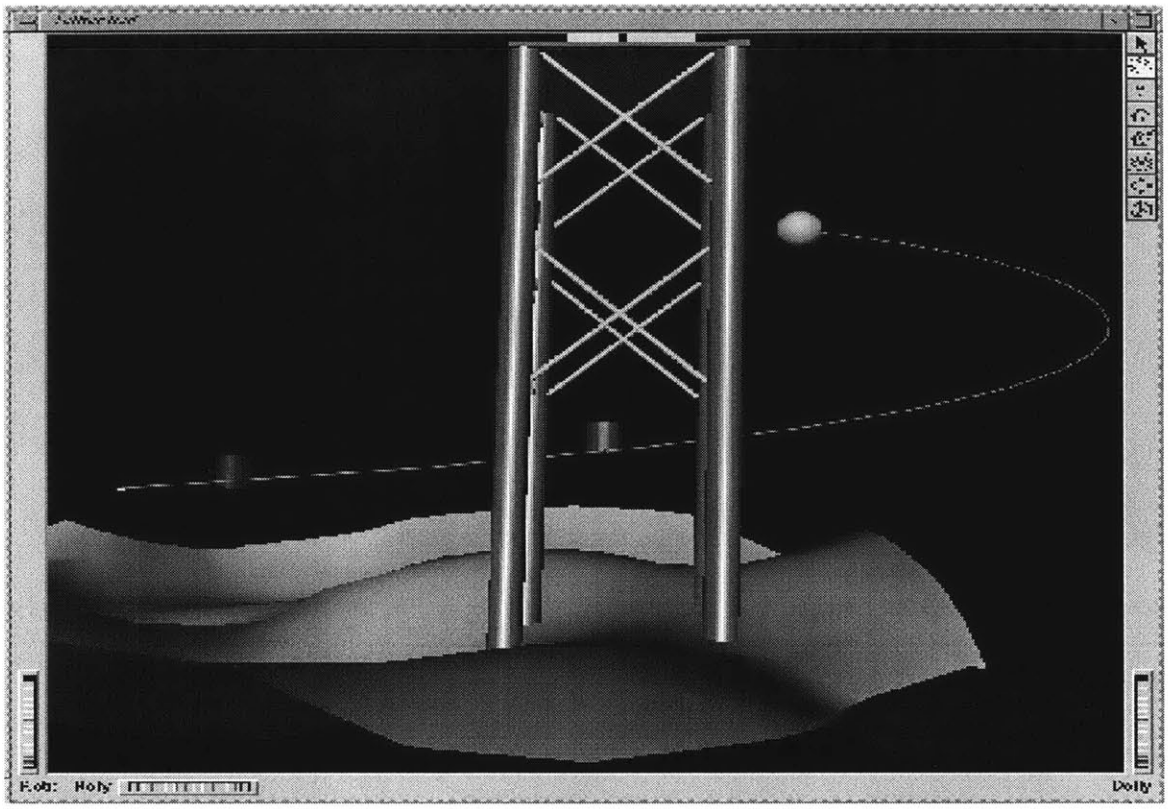

Figure B-3: Catenary model showing linear decrease in depth.

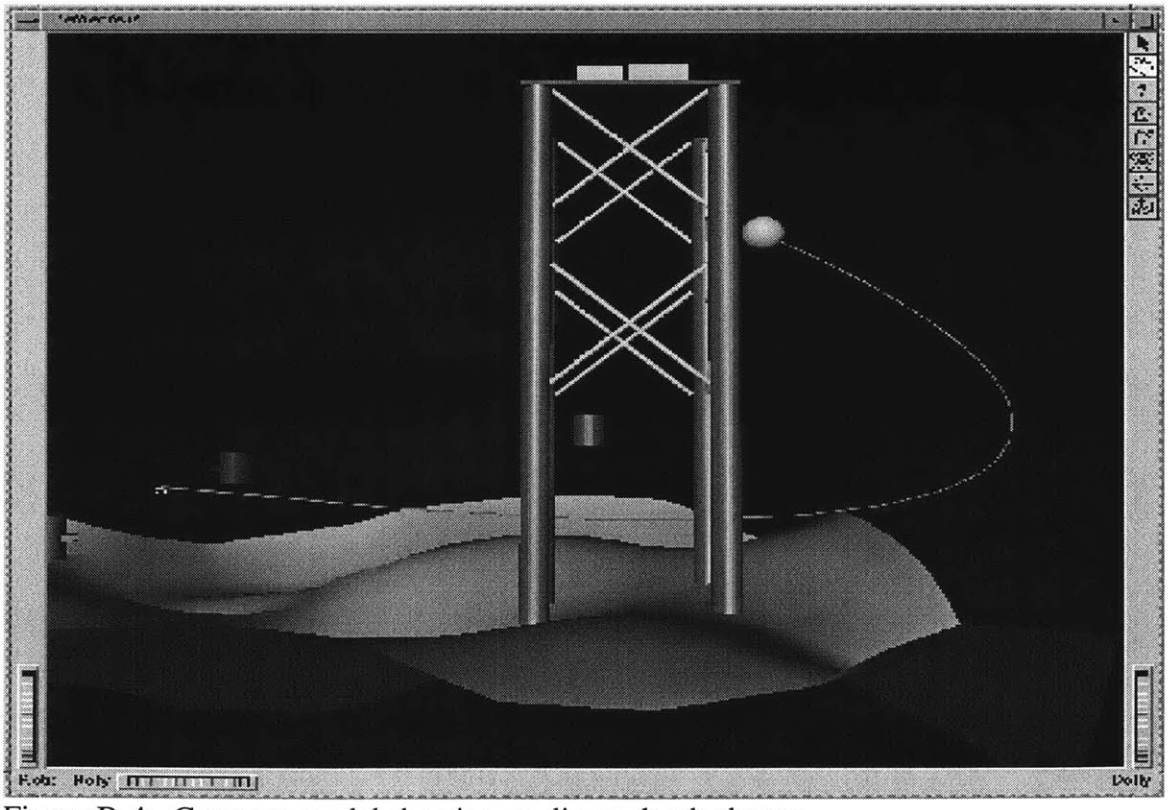

Figure B-4: Catenary model showing nonlinear depth change.

References

[1] Didier Badouel. "An Efficient Ray-Polygon Intersection." *Graphics Gems.* Ed. Andrew S. Glassner. Boston: Academic Press, 1990. 390–393.

[2] Matthew E. Esch. "Determining the Position of Underwater Tethers in Real Time," M.Eng. thesis. MIT, Cambridge, MA. 1998.

[3] Barbara Fletcher. "TRANSoM ROV Pilot Training: Test Results." *Underwater Intervention 1998 Conference Proceedings.* <http://copernicus.bbn.com/Transom/>

[4] Barbara Fletcher and Stewart Harris. "Development of a Virtual Environment Based Training System for ROV Pilots." *Oceans 1996 MTS/IEEE Conference Proceedings.* Sept. 23–26, 1996. Fort Lauderdale, FL.

[5] S. Gottschalk, M. Lin and D. Manocha. "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection." *Proceedings of ACM Siggraph 1996.* Aug. 4–9, 1996. New Orleans, LA. <http://www.cs.unc.edu/~dm/collision.html>

[6] Thomas C. Hudson, et. al. "V-COLLIDE: Accelerated Collision Detection for VRML". *VRML 1997 Conference Proceedings.* Feb. 24–26, 1997. Monterey, CA. <http://www.cs.unc.edu/~geom/collide.html>

[7] Mathworks, Inc. Function fmins.m implemented in *Matlab v5.2.* <http://www.mathworks.com>

[8] Kenneth B. Russell. "The Header2Scheme Home Page." <http://www-white.media.mit.edu/~kbrussel/Header2Scheme/>

[9] Paul S. Strauss. "Vortex." Distributed as sample program vortex.c++ with Open Inventor version 2.1. <http://www.sgi.com/Technology/Inventor/>

[10] UNC Research Group on Modeling, Physically-Based Simulation and Applications. "RAPID User's Manual." Included as part of RAPID library from <http://www.cs.unc.edu/~geom/OBB/version201.html>

[11] Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2.* Reading, Mass.: Addison-Wesley, 1994.

[12] Jonathan L. Zalesky. "A Part-Task Trainer for Underwater Tether Modeling," M.Eng. thesis. MIT, Cambridge, MA. 1998.