

20

A Part-Task Trainer for Underwater Tether Awareness

by

Jonathan L. Zalesky

Submitted to the Department of Electrical Engineering and Computer Science

In Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 22, 1998 [June 1998]

© 1998 Massachusetts Institute of Technology. All rights reserved.

Signature of Author: _____
Department of Electrical Engineering and Computer Science
May 22, 1998

Certified by: _____
Nathaniel L. Durlach
Thesis Supervisor

Accepted by: _____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

JUL 14 1998

Eng

LIBRARY

A Part-Task Trainer for Underwater Tether Awareness

by

Jonathan L. Zalesky

Submitted to the Department of Electrical Engineering and Computer Science

May 22, 1998

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

ABSTRACT

The Training for Remote Sensing and Manipulation (TRANSoM) group has developed a virtual environment (VE) system for training ROV operators. Currently, the training system does not contain a model for tether interaction with objects in the VE. This paper describes a part-task trainer that has been developed for training tether awareness. The part-task trainer can also be used for testing tether models. A proposed algorithm for a tether model that handles collisions with the environment is also discussed.

Thesis Supervisor: Nathaniel L. Durlach

Title: Senior Research Scientist, MIT Research Laboratory for Electronics

TABLE OF CONTENTS

TABLE OF CONTENTS.....	5
LIST OF FIGURES.....	7
1 INTRODUCTION	9
2 PART-TASK TRAINER OVERVIEW	11
2.1 DESCRIPTION	11
2.2 REQUIREMENTS.....	16
3 PART-TASK TRAINER IMPLEMENTATION OVERVIEW	17
3.1 TVEC2D.....	17
3.2 TMATRIX2D	19
3.3 TOBJECT2D.....	20
3.4 TROV2D.....	21
3.5 TOBSTACLE2D	21
3.6 TBOUNDPOINT2D.....	21
3.7 TTETHER2D, TTETHERSEGMENT2D, AND TTETHERSEGMENT2DLISTITEM	22
3.8 TCURRENTDISPLAY2D.....	22
3.9 TTRACERLIST2D	23
3.10 TTETHERINFODISPLAY2D.....	23
3.11 CTRAINERMFCVIEW	23
3.12 OTHER CLASSES.....	24
4 TETHER MODELING PROBLEM	25
4.1 DEFINITIONS	25
4.2 SEGMENT MODELS	26
4.2.1 Rubber Band Model.....	26
4.2.2 'V'-Shape Model.....	27
4.2.3 Funicular Curve	28
4.3 PROPOSED INTERACTIVE MODEL DESCRIPTION	28
4.3.1 Assumptions.....	28
4.3.2 Tether-Environment Interaction Algorithms.....	29
4.3.2.1 Collision Detection and Resolution Algorithm.....	29
4.3.2.2 Collision Validation and Consolidation Algorithm.....	32
5 PROPOSED EXPERIMENTS	35
5.1 TETHER AWARENESS.....	35
5.2 TETHER LOCATION PREDICTION.....	35
6 SUGGESTED FURTHER WORK AND CONCLUSION	37
6.1 TETHER MODEL EXTENSIONS	37
6.2 AUTOMATIC GENERATION OF OBJECT BOUNDING POINTS.....	37
6.3 CONCLUSION.....	38
7 ACKNOWLEDGEMENTS.....	39
8 APPENDIX A – DEFINED TERMS GLOSSARY	41
9 BIBLIOGRAPHY	43

LIST OF FIGURES

Figure 2-1 The part-task trainer view window.....	12
Figure 2-2 Close-up views of the ROV, obstacles, tether, and tether base.....	13
Figure 2-3 Close-ups of the different information displays in the part-task trainer	14
Figure 2-4 Table of Displays and their associated toggle keys.....	14
Figure 2-5 Microsoft Sidewinder 3D Pro joystick.....	15
Figure 2-6 Diagram of the Talon ROV	15
Figure 3-1 Part-Task Trainer Module Dependency Diagram.	18
Figure 3-2 TMatrix2D mapping of a 2-by-2 matrix.....	19
Figure 3-3 Sample placement of bounding points on objects.....	22
Figure 4-1 Illustration of tether terms.....	26
Figure 4-2 Three tether-segment models.....	27
Figure 4-3 Tether collision detection and resolution algorithm.....	30
Figure 4-4 Step-by-step execution of the collision detection and resolution algorithm	31
Figure 4-5 Tether collision validation and consolidation algorithm	32
Figure 4-6 Step-by-step execution of the collision validation and consolidation algorithm.....	33

1 INTRODUCTION

Simulation-based training is becoming an increasingly attractive alternative to traditional training methods. Motivations for this shifting paradigm include decreasing budgets for training and increasing needs to train personnel for increasingly complex tasks. The Training for Remote Sensing and Manipulation (TRANSoM) program was initiated to research, design, develop and evaluate the use of virtual environment (VE) systems for training operators of remotely operated underwater vehicles (ROV). The prime contractor for this effort is Imetrix, Inc. As part of this effort, a system is currently being developed to train operators in shallow water mine countermeasures (MCM) and uses incorporates Intelligent Tutoring System (ITS) techniques.

At present, the MCM training system does not contain a method for modeling the interaction of the ROV's tether with the environment. This paper discusses the design of a part-task trainer program that can be used both as a tool for training operators in cognitive skills related to tether awareness and for interactive testing of dynamic models for underwater tethers. Tether awareness is an important cognitive skill for ROV operators that allows them to mentally track the position and shape of the tether with respect to objects in the environment. The next chapter covers an overview of the part-task trainer and its requirements. Chapter 3 discusses the design and implementation of the part-task trainer. Chapter 4 describes a set of proposed algorithms for managing tether model behavior in an interactive environment. Chapter 5 outlines some tests that that could be performed with the part-task trainer for measuring tether awareness. Finally, Chapter 6 discusses a number of possible extensions that could be added to the part-task trainer program. Appendix A defines a list of terms relevant to the paper's discussion.

2 PART-TASK TRAINER OVERVIEW

This Chapter describes the parts of the part-task trainer, gives an overview of the functional operation of the software design, and delineates the requirements for the current software implementation. For implementation overviews of each of the components, see Chapter 3.

2.1 Description

The part-task trainer is designed to satisfy two purposes. First, it serves as a tool for performing isolated tests involving tether-awareness training, a very important cognitive skill for ROV operators that allows them to mentally track the position and shape of the tether in relation to other objects in the environment. Second, it serves as a simplified test bed for verifying the new tether model designs. The part-task trainer's also design allows for extending its functionality to include other conceivable aspects of the underwater ROV training environment with very little difficulty. Possible extensions include providing auditory feedback about the ROV's motor status and modeling tether drag.

The part-task trainer is designed as a two-dimensional representation of an underwater environment. Figure 2-1 shows the part-task trainer window in a typical simulation run using the 'V'-shape tether model (discussed later in section 4.2.2). The main display depicts overhead view of the underwater environment. It appears as flat grid representing the underwater environment with horizontal, but no vertical, maneuverability. The third dimension intentionally was omitted to simplify the simulation. The generality of the trainer, however, remains – allowing the user to focus on the tests relevant to tether-awareness. Analysis thus will not be complicated by a three-dimensional environment, which greatly eases use. Models developed and tested in the part-task trainer can be converted to three-dimensional models without much difficulty.

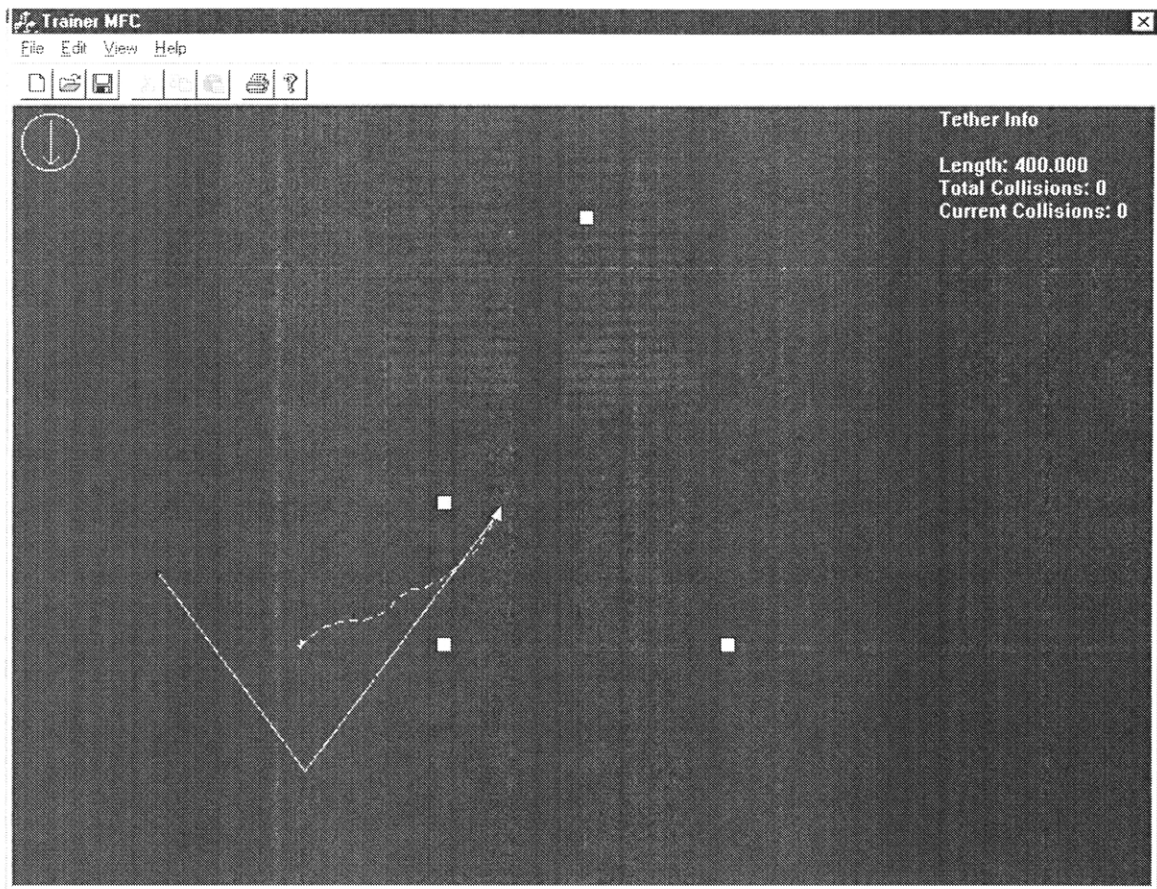


Figure 2-1 The part-task trainer view window

An ROV and a variable number of obstacles can be placed in the underwater environment. Colored polygons portray the ROV and obstacles. The default polygon for the ROV is a red isosceles triangle. The vertex connecting the two longer sides of the triangle denotes the front of the ROV and indicates the ROV's current heading. The default polygon symbol for an obstacle is a yellow square.

The tether connects the ROV to its home base. In practice, the tether would be connected to either a ship or a platform. The home base is shown in the trainer by a small black square. The tether is drawn as a series of connected white line segments running between the home base and the ROV. Figure 2-2 shows a close-up view of the ROV, obstacles, tether (rendered using the 'V'-shape model, discussed later in section 4.2.2), and tether base in the trainer.

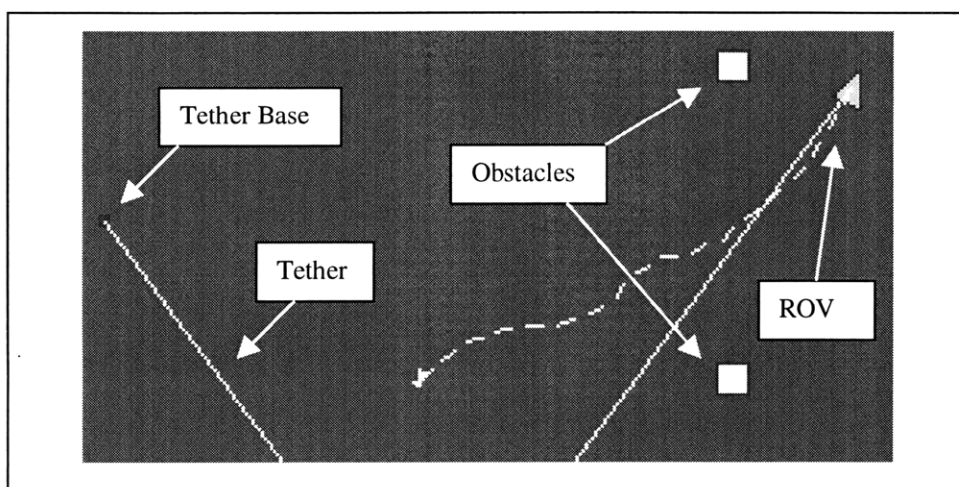


Figure 2-2 Close-up views of the ROV, obstacles, tether, and tether base

Several displays are provided to give the user ancillary information about the environment. These displays are the water current display, tracers, and the tether information display. Figure 2-3 shows details of the displays.

The water current display resides in the upper left corner of the trainer view window. It is a white circle with an arrow and resembles the look of a compass. The arrow shows the direction of the water current flow, and the length of the arrow indicates the magnitude of the current's flow.

A tracer is a line segment indicating the position and heading of the ROV at a specific point in time. A new tracer is plotted at specified uniform time intervals. The tracers give the driver information about the history of the positions and headings of the ROV by tracing a path history. The trainer is set to keep track of a default number of 100 tracers, and the default tracer recording time interval is one second.

The tether information display serves both as a debugging tool for the tether models and an information source for the driver. Examples of information the tether information display can provide are the length of the tether, the status of tether entanglement, and a running count of the number of times the tether has collided with an object. The tether information display occupies the right side of the trainer view window and the information is displayed as text.

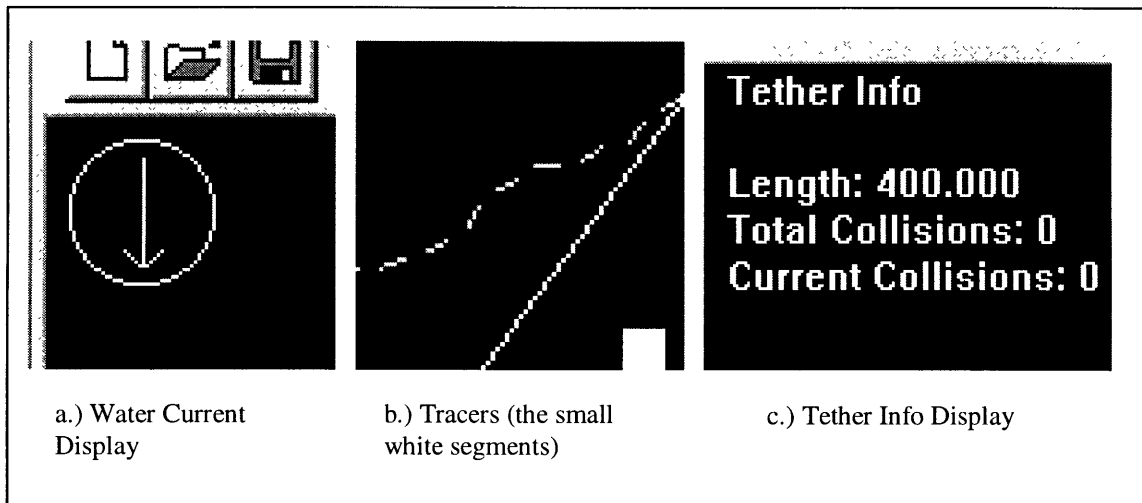


Figure 2-3 Close-ups of the different information displays in the part-task trainer

The display of each component can be turned on or off. Each component is mapped to a keyboard key that toggles the display status. A table of the displays and their associated toggle keys is shown in Figure 2-4 below.

Component	Toggle Key
Obstacles	'O'
ROV	'R'
Tether	'T'
Tether Base	'B'
Tether Info Display	'I'
Tracers	'X'
Water Current Display	'C'

Figure 2-4 Table of Displays and their associated toggle keys

The ROV can be controlled either with a joystick or the keyboard. The intended interface is the joystick because it more closely simulates the operating environment for remote control of an underwater vehicle. The joystick used for this project is the Microsoft Sidewinder 3D Pro. A picture of the joystick is shown in Figure 2-5. It was chosen because, in addition to the two degrees of freedom that a traditional joystick

possesses, the joystick also has a third degree of freedom called rudder control. Rotating the joystick handle operates the rudder control, and controls the rotation of the ROV.



Figure 2-5 Microsoft Sidewinder 3D Pro joystick

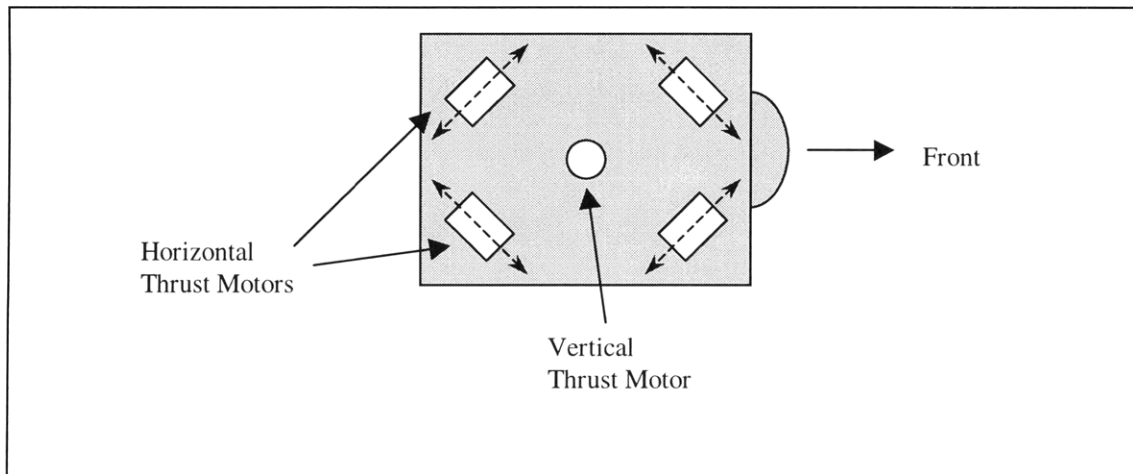


Figure 2-6 Diagram of the Talon ROV

The trainer's ROV dynamics were loosely modeled after the Imetrix's ROV, the Talon. A simple block model of the Talon is shown in Figure 2-6. Talon has four bi-directional thrusters mounted at approximately 45-degree angles at each of the four corners of the ROV. The dashed arrows indicate the direction of thrust delivered by each of the motors. These four motors control the horizontal movement and rotation of the

ROV. A fifth motor is positioned vertically in the center of the ROV and controls vertical movement.

The trainer also has the ability to record a simulation scenario to a log file or to playback log files of previously recorded simulation scenarios.

2.2 Requirements

The part-task trainer requires a minimum configuration of a 60-MHz Pentium computer or equivalent running Microsoft Windows 95 or Windows NT 4.0.

3 PART-TASK TRAINER IMPLEMENTATION OVERVIEW

This Chapter outlines the implementation of the part-task trainer program. It discusses the various object classes and provides reasons for the choices in division of the classes.

The program was developed in C++ using the Microsoft Developers Studio and uses version 4.2 of the Microsoft Foundation Classes (MFC) libraries. Figure 3-1 depicts the module dependency diagram of the program. The module dependency diagram illustrates the implementation design breakdown as well as the dependencies of each of the objects. Solid arrows indicate the dependency of one module upon another. Dashed arrows indicate the sub-classing of modules from their respective super-class modules.

3.1 TVec2D

The TVec2D object was developed as the basic two-dimensional vector representation. It can be used to represent either points or vectors. TVec2D is used primarily in the part-task trainer implementation to represent object vertices, water current direction and magnitude, and force vectors. It has the following mathematical properties and methods for two-dimensional vectors:

Properties:

- X- and Y-coordinates
- Magnitude

Methods:

- Scaling
- Vector addition
- Vector dot product
- Vector cross product
- Angle between two vectors

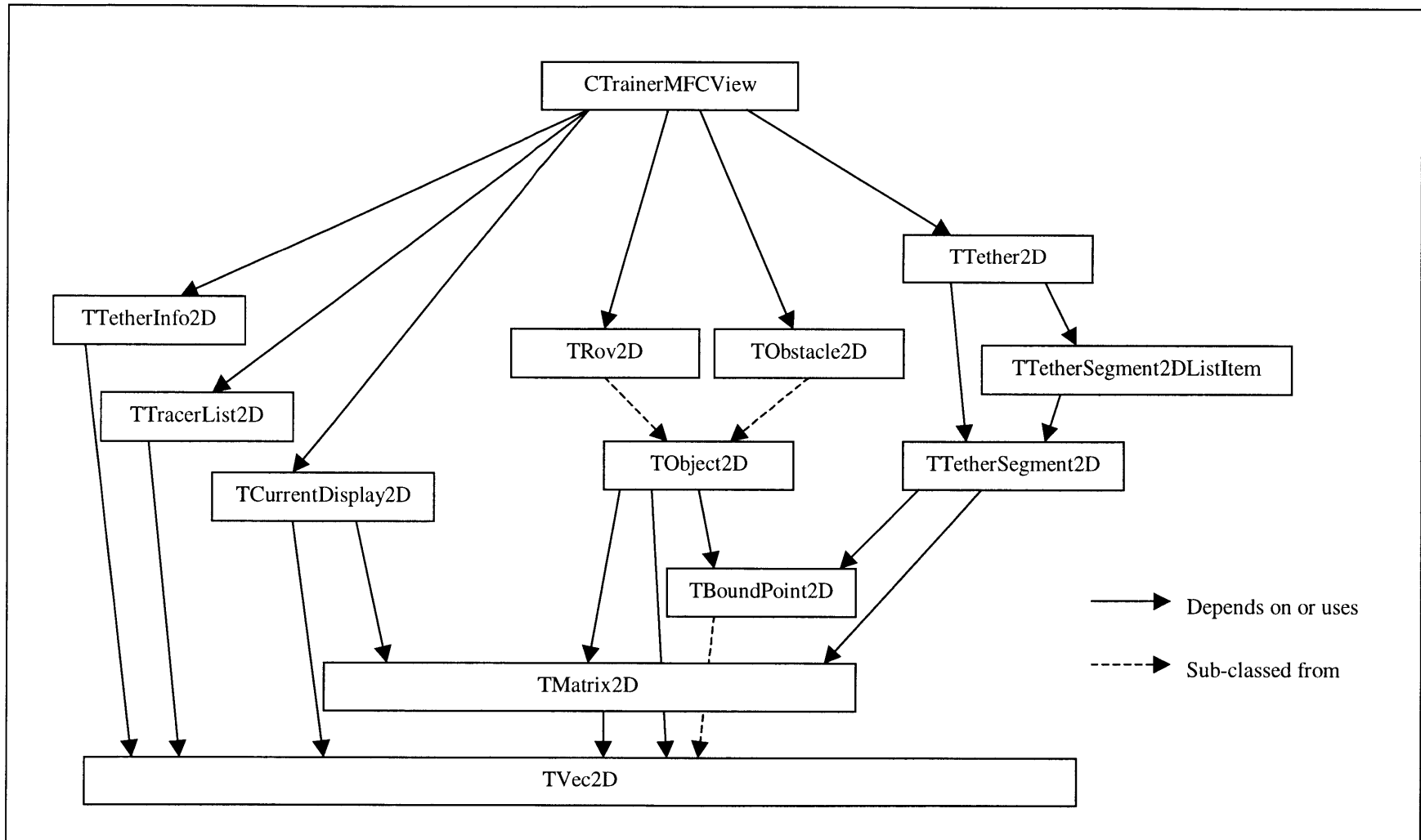


Figure 3-1 Part-Task Trainer Module Dependency Diagram.

3.2 TMatrix2D

The TMatrix2D object is a typical 2-by-2 matrix. The object is mainly used to store rotation matrices for transforming object points and vectors between coordinate spaces, such as transforming objects from object-space to world-space. The matrix is represented internally as a pair of TVec2D vectors. Figure 3-2 shows the mapping of the internal vectors to the elements of the 2-by-2 matrix.

$$\begin{bmatrix} x_x & x_y \\ y_x & y_y \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} x_x \\ y_x \end{bmatrix} & \begin{bmatrix} x_y \\ y_y \end{bmatrix} \end{bmatrix}$$

In TMatrix2D, the Right vector represents the vector pair (x_x, y_x) and the Up vector represents the vector pair (x_y, y_y) .

Figure 3-2 TMatrix2D mapping of a 2-by-2 matrix

The TMatrix2D class has the following properties and methods:

Properties:

- The up and right vectors
- The individual elements of the matrix
- The determinant of the matrix

Methods:

- Matrix addition
- Matrix multiplication
- Multiplication of matrix by a vector
- Scaling
- Taking the inverse of the matrix

3.3 TObject2D

The TObject2D class is the generic object super-class. All two-dimensional objects in the trainer environment (with the exception of the tether, which is described in Chapter 3.7) are sub-classed from the class TObject2D. It was developed so that most of the objects in the environment would have a common interface for performing common operations, such as collision detection between objects and drawing objects to the screen.

It has the following properties and methods common to all generic two-dimensional objects:

Properties:

- Name
- Type (i.e., ROV, obstacle, etc.)
- Color
- Position
- Vertex list
- Bounding radius (for collision detection)
- Bound point list (for collision detection with tether)
- Maneuverability (i.e., mobile or static)

Methods:

- Draw the object
- Update the object's status, given the next time step of the simulation
- Resolve collisions with other objects

3.4 TRov2D

The TRov2D class is sub-classed from the TObject2D super-class and handles the ROV dynamics in the part-task trainer simulation environment. It has the following properties and methods in addition to the TObject2D super-class:

Properties:

- Heading
- Present velocity
- Maximum velocity
- Tether hook point

Methods:

- Apply thrust impulses forward, backward, left, or right
- Apply rotational thrust impulses

3.5 TObstacle2D

The TObstacle2D class is sub-classed from the TObject2D super-class. It is used to represent obstacles in the part-task trainer simulation environment.

3.6 TBoundPoint2D

The TBoundPoint2D class is sub-classed from the TVec2D super-class. It is used to the bounding points of objects. Bounding points are used in detecting collisions between the tether and objects. Currently, obstacles are the only objects that have bounding points. Figure 3-3 shows a sample of where bounding points could be placed on an object. The bounding points create an imaginary hull around the object. Typically, the bounding points line up with the vertices of the object, as shown by the left object in the figure. For irregularly shaped objects, such as the object on the right in the figure, bounding points are placed only on the protruding points.

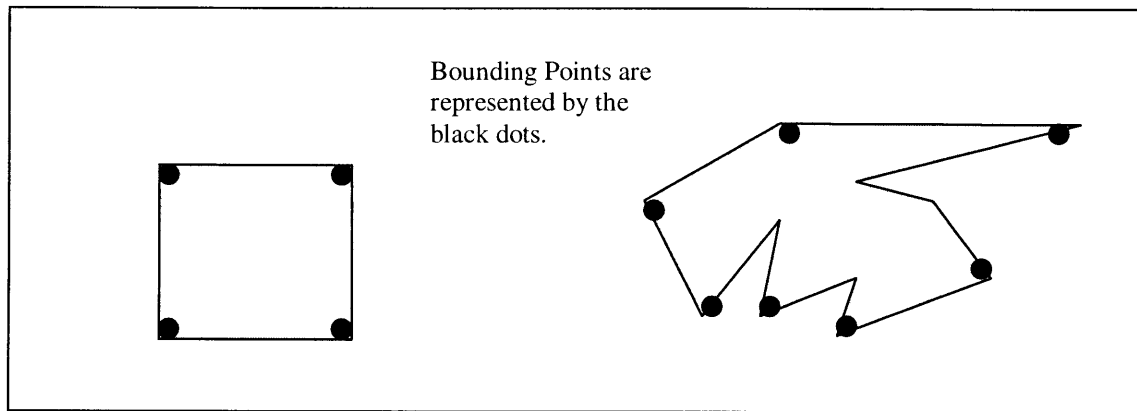


Figure 3-3 Sample placement of bounding points on objects

3.7 TTether2D, TTetherSegment2D, and TTetherSegment2DListItem

TTether2D and TTetherSegment2D describe the tether object shape and behavior in the part-task trainer. The TTetherSegment2D class handles the static shape of the tether model between two defined points. The TTether2D class manages the tether's interaction with the environment. It checks for collisions with obstacles in the environment and manages the internal segments of the tether appropriately. The TTether2D class uses the TTetherSegment2DListItem class to manage the TTetherSegment2D objects as a linked list. A more detailed description of the tether algorithm is described in Chapter 4.

3.8 TCurrentDisplay2D

The TCurrentDisplay2D class is responsible for keeping track of the water current direction. The TCurrentDisplay2D class takes a vector representing the water current direction. The TCurrentDisplay2D class draws itself as a circle in the upper left corner of the view window. It also draws an arrow inside the circle indicating the direction of the water current flow. The display looks similar to a magnetic compass.

3.9 TTracerList2D

The TTracerList2D keeps track of the list of tracers that facilitate tracking the ROV's movement history. When a specified number of simulation time steps elapse, the TTracerList2D generates a line segment (tracer) aligned with the ROV's heading at the ROV's position. The TTracerList2D manages the tracers in a ring buffer array. The TracerList2D stores the newest tracer in the oldest tracer's position and updates the ring buffer pointer appropriately.

3.10 TTetherInfoDisplay2D

The TTetherInfoDisplay2D object displays information about the tether in the view window. The TTetherInfoDisplay2D object queries information from the TTether2D object and draws the information on the right side of the view window in the form of text. The displayed information currently consists of the tether length, tether collision status, and total number of times the tether has collided with an object.

3.11 CTrainerMFCView

The CTrainerMFCView class manages all events related to the view window. The CTrainerMFCView class is also responsible for the running the simulation and handling input from the keyboard and joystick.

During initialization of the view window, the CTrainerMFCView class sets up the configuration of the simulation and sets a timer to the interval specified for the simulation time steps. When the timer elapses, it sends an event to the CTrainerMFCView class and resets itself. Upon receiving the event, the CTrainerMFCView class performs the following tasks:

1. Gets the user input state (either from the joystick, keyboard, or log file)
2. Updates the ROV based on the user input
3. Checks for and resolves collisions between the ROV and the walls of the window and between the ROV and the obstacles

4. Updates the tether model
5. Checks for and resolves collisions between the tether and the obstacles
6. Draws the displays that are currently turned on

The CTrainerMFCView also receives keyboard input for toggling the displays or controlling the ROV.

3.12 Other Classes

The other classes included in the project are perform peripheral functions. A discussion of these classes is not part of the main thrust of the project, but they are mentioned here for completeness.

1. CMainFrame – the MFC wrapper for the view window (CTrainerMFCView), toolbar, and menu bar
2. CTrainerMFCApp – the MFC wrapper for the whole windows application
3. CTrainerMFCDoc – the MFC wrapper for documents (not used)
4. CAboutDlg – the typical “about...” dialog box, which has been edited to provide version and copyright information, joystick and keyboard control information, and display toggle key mappings

4 TETHER MODELING PROBLEM

Many factors influence the behavior of the tether such as gravity, water current, and ROV dynamics. Other significant influences include the physical properties of the tether itself. Tethers come in a variety of lengths, thicknesses, and materials. The material of which a tether is made affects its flexibility and buoyancy. The behavior of a thick, inflexible, dense tether rarely matches the behavior of a thin, highly flexible, neutrally buoyant tether. This Chapter proposes a model that handles interaction between a tether and objects in the environment. Although only flexible, neutrally buoyant tethers were observed (namely, the tether for the Talon ROV), the model can be adjusted to take into account the properties of various tethers.

The model developed in this thesis separates the tether modeling problem into two distinct parts in order to make the problem more manageable. One part deals solely with the tether's shape between two endpoints. The other part detects and resolves collisions between the tether and other objects in the environment. Chapter 4.2 discusses the part of the model that determines the tether's static shape and covers the tether shapes that have been implemented. Chapter 4.3 discusses the algorithm developed for handling tether-object collisions and collision resolution.

4.1 Definitions

For purposes of discussion the following terms must be defined. Refer to Figure 4-1 for an illustration of the terms.

1. Tether segment – a section of tether with a defined start point, end point, and length. The shape models discussed in section 4.2 are applied to each tether segment of a tether.
2. Segment link – a section of a tether segment represented as a line segment.
3. Tether – the entire tether entity with a defined base point (where the tether enters the water) and end point attached at the ROV. It is represented as an ordered list of one or more tether segments. The list is in order starting from the base point. The end point of the first tether segment is the start point for the next segment, and so on.

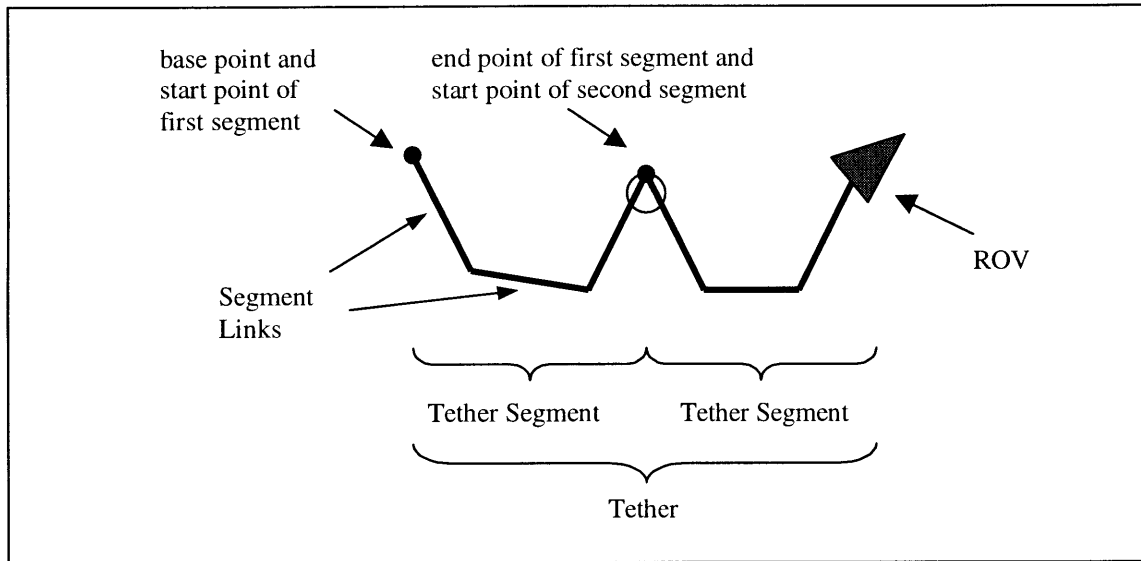


Figure 4-1 Illustration of tether terms

4.2 Segment Models

When a tether-segment model is rendered, it is represented as a series of line segments that are referred to as tether segment links, like the links of a chain. There are many different shapes a tether segment can take given the length the tether segment, water current and gravity. The following models from Matt Esch’s thesis, “Determining the Position of Underwater Tethers in Real-Time”, have been implemented in the part-task trainer. These models are meant to be approximations that capture some characteristics of the free-form shape of the tether.

4.2.1 Rubber Band Model

The first model is dubbed the rubber band model. In this model the tether is depicted as a straight line between the two end points regardless of the distance between the two end points. Figure 4-2a shows a sample rendering of a tether segment using the rubber band model. The rubber band model serves as a first-order approximation of the tether’s position. It traces a direct path between two end points and shows the tether’s whereabouts relative to other objects in the environment. The rubber band model also has the benefit that it is not very calculation-intensive. The model does, however, have

the severe limitation that it yields no visual information about the actual length or shape of the tether segment unless the tether is stretched to the limit.

4.2.2 'V'-Shape Model

The 'V'-shape model is modeled as two segments. The 'V'-shape model has an advantage over the rubber band model in that it gives visual information about the length and shape of the tether segment between the two end points. Figure 4-2b shows a sample rendering of the tether segment between the same two end points as Figure 4-2a using the 'V'-shape segment model.

The motivation behind the 'V'-shape model comes from the observation that water current tends to drag the tether into a natural funicular curve shape. The 'V'-shape is used as a simple approximation to this shape. The kink between the two tether segment links in the 'V'-shape is found with very simple geometry calculations. The 'V'-shape gives an impression of the bending direction and the general shape of the tether.

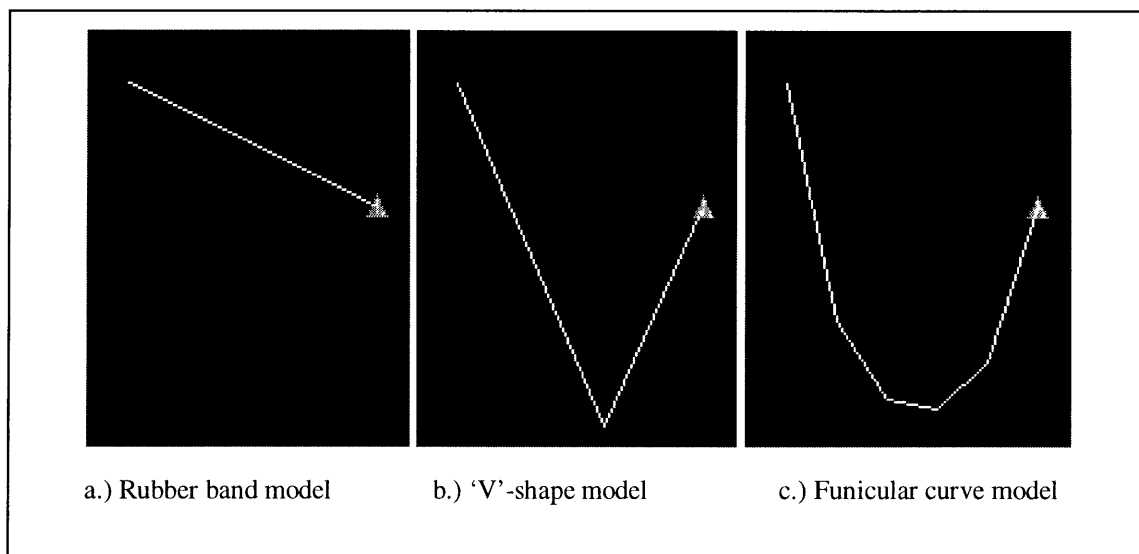


Figure 4-2 Three tether-segment models

4.2.3 *Funicular Curve*

The funicular curve model is the natural shape that a cable takes when acted upon by a force. Figure 4-2c shows a sample rendering of the tether segment using the funicular curve model. It is rendered with the same two end points as the models in Figure 4-2a and Figure 4-2b. The funicular curve model is the most physically accurate of the three models. It closely resembles the shape of a real tether in water with a strong current.

4.3 Proposed Interactive Model Description

This Chapter describes the algorithm developed for managing the tether's interaction with the environment. First, the assumptions upon which the model is based are described. Next, the algorithm is described. Finally, the current implementation of the algorithm in the part-task trainer is described.

4.3.1 *Assumptions*

The following assumptions were made in the development of the algorithm:

1. The time step of the simulation is small enough that the tether model's shape and position will not vary significantly between each time step.
2. The model for each tether segment generates a list of connected line segment links representing the tether.
3. Each object in the environment that can collide with the tether has one or more bounding points associated with it.
4. Only immovable objects can collide with the tether.

Assumption 1 allows for continuity in the simulation of the tether model. For instance, if the time step were too large, the quantum of movement accounted for during in the time steps would also be large, causing significant movement in the ROV's position. If the ROV moves too far in a time step, the recalculation of the tether model

could cause the tether to appear to go through an obstacle. Assumptions 2 and 3 provide the method to detect collisions with objects. The tether-segment links are the geometric model used in testing for collisions with object bounding points. Assumption 4 simplifies the algorithm further by allowing only static objects to impact the tether. Collisions with movable objects create a host of additional problems, such as having to track additional dynamic forces acting on the tether and the moving points of contact between the mobile objects and the tether.

4.3.2 Tether-Environment Interaction Algorithms

The tether is composed of a chain of tether segments that must be managed. When dealing with the tether segments in the environment, the tether management system must detect collisions between any of the tether segments and objects in the environment. Once collisions are detected, the system must do the bookkeeping necessary to keep track of the collision. The system must also recalculate the tether segment models using the tether's end points and the tether's contact with the object. Second, the tether management system must constantly monitor the places where the tether is in contact with an object and determine whether the collision is affecting the shape of the tether. The following sections describe algorithms that have been developed to handle both of the operations in this system.

4.3.2.1 Collision Detection and Resolution Algorithm

Figure 4-3 shows the pseudo-code for the tether collision detection and resolution algorithm. The algorithm checks for collisions between the tether segments and the bounding points of objects in the environment and adjusts the tether appropriately. It iterates through each object in the environment and checks for a collision between the object's bounding points and any of the links of each tether segment.

```

CheckCollisions(tether_segment_list, object_list)
1  For each object in object_list do
2      For each bound_point of object do
3          For each tether_segment of tether_segment_list do
4              If Collision(tether_segment, bound_point) then
                    Break tether_segment into tether_segment_a and
                    tether_segment_b at bound_point;
6              Insert tether_segment_a and tether_segment_b into tether_segment_list
                    in place of tether_segment;
7              Record direction vector of collision junction;
8              RunModel(tether_segment_a);
9              RunModel(tether_segment_b);
10             End % if collision
11         End % for each tether_segment
12     End % for each bound_point
13 End % for each object

```

Figure 4-3 Tether collision detection and resolution algorithm

If a collision between the tether segment and a bounding point is found, lines 5-10 handle the collision management. Line 5 breaks the tether segment into two tether segments at the location of bounding point. The bounding point becomes the end point and junction of the two new tether segments. One of the new tether segments is assigned length of the tether between the old segment's start point and the bounding point. The other new segment is assigned the rest of the old segment length. Line 6 removes the old tether segment from the tether segment list and inserts the two new segments in its place. The direction of the collision is recorded in line 7 for validating the collision. The collision validation is done in the tether collision validation and consolidation algorithm discussed in the next section. Finally, lines 8 and 9 calculate the shapes of the new tether segments. Figure 4-4 provides an illustration of a step-by-step execution of the algorithm.

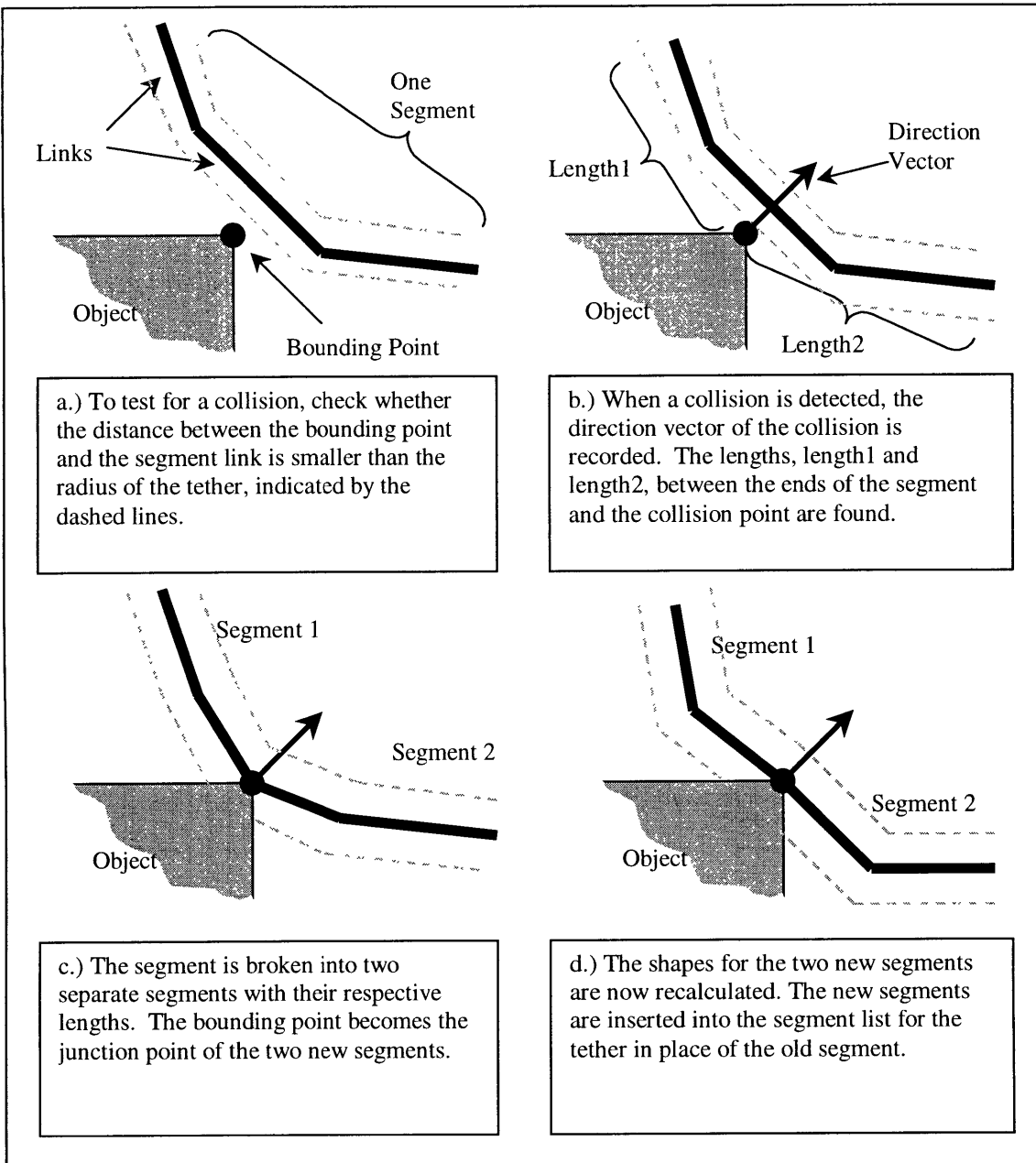


Figure 4-4 Step-by-step execution of the collision detection and resolution algorithm

4.3.2.2 Collision Validation and Consolidation Algorithm

```
ValidateCollisions(tether_segment_list)
1  tether_segment  $\leftarrow$  first segment in tether_segment_list;
2  While tether_segment  $\neq$  last segment in tether_segment_list do
3      tether_segment_next  $\leftarrow$  next segment after tether_segment in tether_segment_list;
4      direction_vector  $\leftarrow$  DirectionVector(tether_segment, tether_segment_next);
5      link_a  $\leftarrow$  LastLink(tether_segment);
6      link_b  $\leftarrow$  FirstLink(tether_segment_next);
7      If Angle(link_a, direction_vector) + Angle(link_b, direction_vector) < 180° then
8          EndPoint(tether_segment)  $\leftarrow$  EndPoint(tether_segment_next);
9          Length(tether_segment)  $\leftarrow$  Length(tether_segment) + Length(tether_segment_next);
10         Remove tether_segment_next from list;
11         tether_segment  $\leftarrow$  next segment after tether_segment in tether_segment_list;
12     Else % collision still valid
13         tether_segment  $\leftarrow$  tether_segment_next;
14     End % if angle between the two links < 180°
15 End % while tether segment not equal to last segment in list
```

Figure 4-5 Tether collision validation and consolidation algorithm

Figure 4-5 shows the pseudo-code for the tether collision validation and consolidation algorithm. The algorithm inspects each of the tether segment junctions. Lines 4 through 6 extract the two segment links connecting the two tether segments at the junction and the direction vector. Line 7 measures the angle between the two links at the junction. The direction vector is used as a reference to determine which angle to measure. If the angle between the two links is less than 180 degrees, lines 8 through 11 consolidate the two tether segments. The end point of first tether segment is set to the end point of the second tether segment. The length of the first segment is set to the combined length of the two segments. Finally, the second segment is removed from the list. Figure 4-6 provides an illustration of a step-by-step execution of the algorithm.

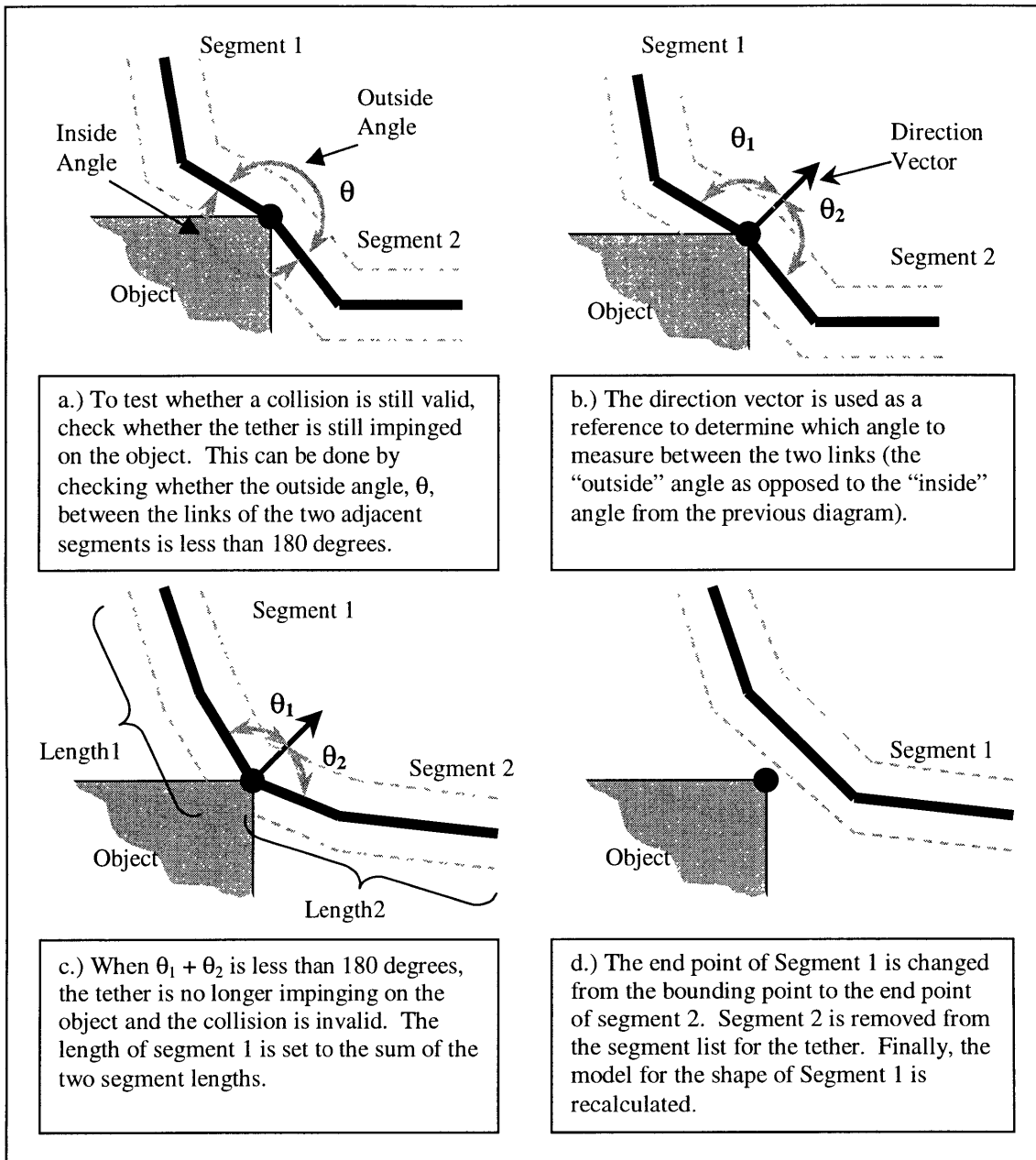


Figure 4-6 Step-by-step execution of the collision validation and consolidation algorithm

5 PROPOSED EXPERIMENTS

This Chapter describes two proposed experiments that can be performed using the part-task trainer.

5.1 Tether Awareness

Tether awareness is an important cognitive skill for ROV operators. The operators must have an idea of where the tether is relative to objects in the environment while navigating the ROV in order to prevent the tether from getting snagged on any of the objects.

In this test, the operator's goal is to navigate the ROV through a series of obstacles or to a defined destination and then to return to base within a specified time limit. The control group will perform the test while the tether is not visible but will see a visible base point and the tether length displayed in the tether information display. The second group will perform the test first with the tether visible and again with the tether not visible.

The scoring for the test can be based on the amount of time the operator takes to complete the task and the total number of tether collisions during the course of the test. The test will provide insight on whether training with the visible tether model teaches operators tether awareness.

5.2 Tether Location Prediction

Tether location prediction is a valuable cognitive skill for ROV operators to possess while navigating. An operator can perform navigation tasks more effectively if he has a mental image of the shape of the tether at any specific point in time.

In this test the operator is asked to navigate the ROV towards a destination. The test can be performed with or without obstacles in the environment. The simulation is paused after a fixed time limit and the operator is asked to plot the shape of the tether. The control group will perform the test with the tether not visible and without prior

knowledge of the tether model. The second group will drive the ROV around while the tether is visible in order to become familiar with the tether model and then perform the test with the tether not visible.

The scoring can be based on the area between the plot of the operator's estimation of the tether shape and the plot of the actual tether shape. A comparison of the scorings in the groups will determine whether the second group is able to learn the tether dynamics more effectively than the first group. The comparison will also give insight into the differences in the ways that each group is able to learn aspects of the shape and behavior of the tether.

6 SUGGESTED FURTHER WORK AND CONCLUSION

The part-task trainer is a simple self-contained tool providing a mechanism for performing experiments related to tether awareness training, such as those described previously in Chapter 5. However, there are many things can be done to improve or extend aspects of the trainer. Some of these improvements are described below.

6.1 Tether Model Extensions

Many extensions can be made to the tether model in the part-task trainer. The interactive tether model algorithm proposed in this paper specifies that the tether is locked to an object after it collides with that object. The model could be extended to include the tether forces at collision points and at the connection to the ROV. The model can use forces on the tether at the collision points to model tether slippage around the object at the point of collision. The simulation can use tether forces at the ROV connection point to more accurately model the effects of tether drag on the ROV's maneuverability. Each of these extensions will help in the creation of a more physically accurate model of tether behavior and its effect on the ROV.

The tether collision model can also be extended to a three-dimensional model. A set of bounding line segments could be generated to create a bounding cage for the three-dimensional objects instead of bounding points. The bounding line segments of the objects would then be used to detect for collisions between the tether segment links and the objects. Collision resolution between the tether and object at the point of collision would then proceed as previously described in section 4.3.

6.2 Automatic Generation of Object Bounding Points

Object bounding points must be defined manually for all objects in the simulation. A method can be developed to automatically generate bounding points for the objects. One such method to explore is running a convex hull algorithm on the object points to find bounding point candidates.

6.3 Conclusion

This paper has described the design and implementation of a part-task trainer computer program. The part-task trainer can serve as a tool for teaching ROV operators the cognitive skills related to tether awareness. Using the part-task trainer operators can learn to develop a mental picture of the tether's shape and learn to remember the tether path back to the home base so that the tether will not get snagged on an object.

The part-task trainer can also be used as a tool for developing and testing interactive tether model algorithms. The design of the part-task trainer allows for flexibility in implementing tether models other than the ones described previously in Chapter 4. The design of the trainer may also serve as the basis for the development of other part-task trainers, such as one that incorporates auditory feedback on the status ROV's motors.

It is hoped that the TRANSoM group and others researching in similar areas will find the part-task trainer a useful addition to their efforts.

7 ACKNOWLEDGEMENTS

The author would like to thank all of those who provided support, both technical and emotional, throughout this research process. David Manowitz, Matt Esch, and Christopher Gadda helped in the brainstorming of possible interactive tether models and contributed to the development of the model presented here. Abigail Vargus helped with the structuring and editing of this paper. Nathaniel Durlach and Thomas Wiegand provided guidance and support throughout the research and development process.

8 APPENDIX A – DEFINED TERMS GLOSSARY

The following terms are defined as they relate to the interpretation of this thesis. Most of the terms are generally accepted terms. Some terms, however, have been created to have a language for communicating concepts within the tether algorithm design paradigm.

ITS – Intelligent Tutoring System

MCM – Mine Countermeasures Mission

MFC – Microsoft Foundation Classes

Part-task trainer – a program that simplified version of a full-featured training system whose design is focused on a specific aspect of the training simulation or performing a specific task

ROV – Remote Operated Vehicle

Super-class – (in computer programming) a generic class with properties and methods many similar classes have in common

Sub-class – (in computer programming) a class that inherited the properties and methods of another class

Tether-awareness – a cognitive skill valuable to ROV operators, which involves having a mental picture of the tether's shape and position relative to objects in the environment during ROV operations

Tether – A tether is the cable that connects an ROV to the control base. Tethers contain control lines for the ROV and sensor feedback lines. Sometimes tethers also provide the power lines for the ROV.

Tether Link – A tether link is a section of a tether segment represented as a line segment. Tether links are used in the tether model in collision detection and visual representation of the tether.

Tether Segment – a section of the tether in free space between two objects in the environment

Tracer – a line segment plotted at the current position and orientation of the ROV in the part-task trainer

TRANSoM – (Traning for Remote Sensing and Manipulation) group whose goal is to research, develop, and evaluate VE systems for training in the operation of ROVs

VE – Virtual Environment

9 BIBLIOGRAPHY

- [1] Dennis, N., "On the Formation of Funicular Curves", International Journal of Mechanical Science, Vol. 36, No. 3, pp. 183-188, 1994.
- [2] Esch, Matt, "Determining the Position of Underwater Tethers in Real-Time", to be published, 1998.
- [3] Fletcher, Barbara and Stewart Harris, "Development of a Virtual Environment Based Training System for ROV Pilots", Oceans '96 MTS/IEEE Conference Proceedings, September 1996.
- [4] Fletcher, Barbara, "MCM Applications of a Virtual Environment Based Training System for ROV Pilots", Symposium on Technology and the Mine Problem, November 1996.
- [5] Foley, James D., et. al. (1985) Computer Graphics: Principles and Practice, 2nd ed. New York: Addison-Wesley.
- [6] Plastock, Roy A. & Gordon Kalley (1986) Schaum's Outline Series: Theory and Problems of Computer Graphics. New York: McGraw-Hill.
- [7] "Training for Remote Sensing and Manipulation (TRANSoM)", BBN web site reference (May 1998), <http://copernicus.bbn.com/Transom/>.