Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Simulation of Launch and Recovery of Small Craft Including Cable Collisions and Cable Tension from Deck Personnel

A. Roy
D. Steinke
R. Nicoll
Dynamic Systems Analysis Limited

Prepared by:
Dynamic Systems Analysis Limited
101-19 Dallas Road, Victoria, BC, Canada V8V 5A6

Project Manager: Dean Steinke, 902-407-3722
Contract Number: W7707-135587/001/HAL
Contract Scientific Authority: Kevin McTaggart, 902-426-3100 ext 253

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2013-136
August 2013

Canada

This page intentionally left blank.

Simulation of Launch and Recovery of Small Craft Including Cable Collisions and Cable Tension from Deck Personnel

A. Roy
D. Steinke
R. Nicoll

Prepared by:

Dynamic Systems Analysis Limited
101-19 Dallas Road, Victoria, BC, Canada, V8V 5A6

Project Manager: Dean Steinke 902-407-3722
Contract Number: W7707-135587/001/HAL
Contract Scientific Authority: Kevin McTaggart 902-426-3100 ext. 253

Defence Research and Development Canada – Atlantic
Contract Report
DRDC Atlantic CR 2013-136
August 2013

Abstract

This report describes the simulation of launch and recovery of a small boat from a ship using a boom crane. The simulation was developed using the existing Ship Mechanical Systems Application Programmer Interface (SMS API). The SMS API was extended to include modelling of collisions between cables and solid objects, which can include a ship or a small boat. The influence of tag lines being handled by deck personnel was modelled, with a winch model being used to approximate the cable tension being applied by a person on deck. Simulation components were verified by comparison with a number of test cases. A complete launch and recovery scenario was simulated and visualized.

Résumé

Le présent rapport décrit la simulation de mise à l'eau et de récupération d'une petite embarcation à partir d'un navire à l'aide d'une grue à flèche. La simulation a été développée à l'aide de la SMS API (Ship Mechanical Systems Application Programmer Interface) existante. La SMS API a été étendue pour inclure la modélisation des collisions entre les câbles et les objets solides, ce qui peut comprendre un navire ou une petite embarcation. L'influence des câbles stabilisateurs manipulés par le personnel du pont a été modélisée, avec un modèle de treuil utilisé pour obtenir une approximation de la tension des câbles appliquée par une personne sur le pont. Les éléments de simulation ont été vérifiés en les comparant à un nombre de cas d'essai. Un scénario de mise à l'eau et de récupération complet a été simulé et visualisé.

This page intentionally left blank.

Executive summary

Simulation of Launch and Recovery of Small Craft Including Cable Collisions and Cable Tension from Deck Personnel

A. Roy, D. Steinke, R. Nicoll; DRDC Atlantic CR 2013-136; Defence Research and Development Canada – Atlantic; August 2013.

Introduction: Launch and recovery of small boats has become an increasingly important part of naval operations. This report describes extension of the existing Ship Mechanical Systems Application Programmer Interface to improve simulation fidelity for launch and recovery of small boats.

Principal Results: Modelling of collisions between flexible cables and solid objects was implemented in the Ship Mechanical Systems Application Programmer Interface. In addition, handling tag lines by deck personnel was simulated using a winch model. DRDC Atlantic ran the delivered launch and recovery simulation and found that it ran 600 times slower than real time. Much of the computational time was likely devoted to modelling of collisions between cables and solid objects. Visualizations of the launch and recovery simulation suggested that physical phenomena were being correctly modelled.

Significance of Results: Simulation of launch and recovery simulation is promising for applications such as training and engineering design. Validation using full-scale trial data and increased computational speed are two recommended areas for development.

Future Plans: Defence R&D Canada will continue to work on development of simulation of launch and recovery.

Sommaire

Simulation of Launch and Recovery of Small Craft Including Cable Collisions and Cable Tension from Deck Personnel

A. Roy, D. Steinke, R. Nicoll ; DRDC Atlantic CR 2013-136 ; Recherche et développement pour la défense Canada – Atlantique ; août 2013.

Introduction : La mise à l'eau et la récupération de petites embarcations sont une partie de plus en plus importante des opérations navales. Ce rapport décrit l'extension de la SMS API (Ship Mechanical Systems Application Programmer Interface) existante pour améliorer la fidélité de la simulation pour la mise à l'eau et la récupération de petites embarcations.

Résultats principaux : La modélisation des collisions entre les câbles flexibles et les objets solides a été mise en œuvre dans la SMS API. De plus, la manipulation des câbles stabilisateurs par le personnel de pont a été simulée à l'aide d'un modèle de treuil. RDDC Atlantique a effectué la simulation de mise à l'eau et de récupération et a découvert qu'elle se faisait 600 fois plus lentement qu'en temps réel. Une bonne partie du temps de calcul était probablement dédiée à la modélisation des collisions entre les câbles et les objets solides. La visualisation de la simulation de mise à l'eau et de récupération suggérait que les phénomènes physiques étaient modélisés correctement.

Importance des résultats : La simulation de la mise à l'eau et de la récupération est prometteuse pour les applications comme la formation et la conception technique. La validation à l'aide de données d'essai pleine échelle et la vitesse de calcul augmentée sont deux domaines recommandés pour le développement.

Travaux ultérieurs prévus : Recherche et développement pour la défense Canada continuera à travailler au développement de la simulation de la mise à l'eau et de la récupération.

Table of contents

List of figures

List of tables

This page intentionally left blank.

1 Introduction

Dynamic Systems Analysis (DSA) was contracted by DRDC Atlantic to extend the functionality of the Ship Mechanical Systems Application Programming Interface (SMS API). This project started September 2012 and completed March 2013. This report documents the work completed by DSA during this time period.

The SMS API is a software library designed for the accurate simulation of mechanical systems such as those which might be found on a floating naval platform. In particular, the SMS API allows the simulation of launch and recovery operations of small marine vehicles from large naval platforms. The main objective of this work was to increase the fidelity of the simulation of the Launch and Recovery (L&R) operation developed in [3]. That L&R simulation contained tag lines and cables that did not interact with the ship's deck or hull. That is, the cables passed through the ship's geometry as if it were empty space because collisions were not detected or resolved. The focus of this work was to add the ability to model contact forces between cables and the ship's geometry and resolve collisions in the SMS API.

Contact resolution is a two step process: detecting collisions and determining contact forces. In order to prevent a cable from passing through the deck of a ship in a numerical simulation, the time and locations of contact must be detected. Once a collision is detected, a contact dynamics model provides the force magnitude and direction which is then applied to the bodies. Adding the contact force to the equations of motion for those models ensures separation is maintained. This work employs the Hunt-Crossley contact dynamics model to determine the contact force between cables and rigid bodies [4]. The Hunt-Crossley contact dynamics model is reasonably simple and computationally efficient.

Collision detection in multi-body simulations is usually divided into two phases: broad phase and narrow phase. Narrow phase collision detection focuses detecting collisions between two specific bodies. Broad phase collision detection prunes the total number of narrow phase collision queries. In other words, broad phase collision is charged with efficiently addressing the N-body collision detection problem [5]. The SMS API collision detection architecture was initially constructed to only resolve collisions between a `Payload` object (the small craft) and the ship's hull. To handle collisions between multiple cables and objects (the cable and the ship deck, the small craft, etc.), the ability to handle multi-body collisions was required. A new collision detection infrastructure was created to handle multi-body collisions.

This document discusses the development in updating the L&R simulation with cable collision resolution. An overview of general narrow phase collision detection methods is discussed in Section 2. Detailed descriptions of the two cable collision detection methods that were implemented are presented in Section 3. Broad phase collision detection was

addressed by using a brute force strategy as discussed in Section 4. The infrastructure changes made to the SMS API, including newly added classes, are discussed in Section 5. The Hunt-Crossley contact dynamics model is discussed in detail in Section 6. The deck hand modelling approach is presented in Section 7. Some of the challenges that were encountered in updating the L&R simulation are discussed in Section 8. With the new collision detection software infrastructure, some updates to the manual were required; these are presented in Section 9. Finally, simulation results using the new cable collision resolution method are presented in Section 10.

2 Overview of narrow phase collision detection methods

This section provides a brief overview of the narrow phase collision detection methods considered for this project. Section 2.1 reviews types of narrow phase detection algorithms. Following this, geometric considerations for choosing a collision detection method are discussed in Section 2.2. Next, important considerations for collision detection between flexible cables and generic objects are presented in Section 2.3.

The two narrow phase collision detection techniques considered were Bounding Volume Hierarchy (BVH) and Stochastic Global Optimisation (SGO) methods, which are presented in Sections 2.4 and 2.5, respectively.

2.1 Narrow phase collision detection algorithms

There are numerous methods available to resolve the narrow phase collision detection problem. They can be classified into two categories: collision queries and proximity queries. A collision query will specifically check whether two objects are interfering. The result of this algorithm is a boolean true or false. In contrast, a proximity query provides the minimum separation distance (MSD) between two objects as shown in Figure 1. The proximity information often includes the points on each object ($\mathbf{p}_1$ and $\mathbf{p}_2$) that are closest to the other. If the separation distance between the two objects is zero, the objects are interfering; thus proximity queries can also be considered a collision query [6]. The MSD is required to resolve the contact force magnitude and direction between a cable and a rigid body; a proximity query is required for this work.

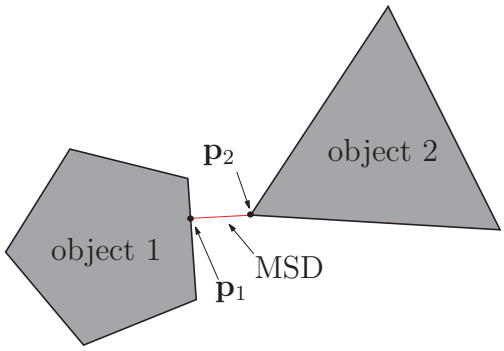


Figure 1: *Minimum separation distance (MSD) between two convex polygons. Points* $\mathbf{p}_1$ *and* $\mathbf{p}_2$ *represent the surface locations on objects 1 and 2 that correspond to the MSD.*

2.2 Geometric considerations

It is necessary to consider the geometry of the bodies when choosing a collision detection method. Most methods have been developed for polygonal geometry representation. In addition, bodies can be categorised into convex and concave. Convex objects are especially well suited to collision detection because of inherent properties [5]. The bulk of the collision detection methods described in the literature and used in practice are for convex polyhedral geometries. Some notable convex polyhedral collision detection algorithms are GJK [7], V-Clip [8], and MinDist [9].

In comparison, concave object collision detection techniques are less developed or prevalent in the literature [10, 11, 12]. One reason for this is that convex object collision techniques can be employed for concave objects through convex decomposition. The cost of this technique increases with the number of convex sub-objects required to represent the concave geometry. This problem can be minimised by pruning through the use of a bounding volume hierarchy (BVH) [5, 13, 14, 15]. Alternatively, global optimisation techniques have been used succesfully [2, 11, 16, 17].

2.3 Considerations for collision detection between cables and rigid bodies

In practice, BVHs and other spatial partitioning schemes such as Octrees are used to resolve broad phase collision detection problems [5]. However, BVHs can also be used for efficient narrow phase collision detection of concave objects that make use of convex decomposition. Because a cable's geometry is inherently concave, the use of BVH as well as SGO methods for narrow phase collision detection were considered. Both BVH and SGO methods were considered because it was unclear which method would be most effective.

A literature review of BVH and SGO methods can be found in Sections 2.4 and 2.5, respectively. These sections present techniques for detecting collisions between a cable and a rigid body. To simplify the problem, rigid bodies (such as a ship or small craft) are represented as convex polyhedra and cables are represented either as volumetric cubic splines (SGO method) or as volumetric linear line segments (BVH method).

Since there is potential for multiple simultaneous collisions between the cable and rigid body, there can be multiple MSD solutions on the concave cable geometry. There are many examples in the literature that address cable collision detection with either global optimisation methods [18, 19, 20, 21] or convex-decomposition BVH methods [1, 21, 22, 23, 24, 25].

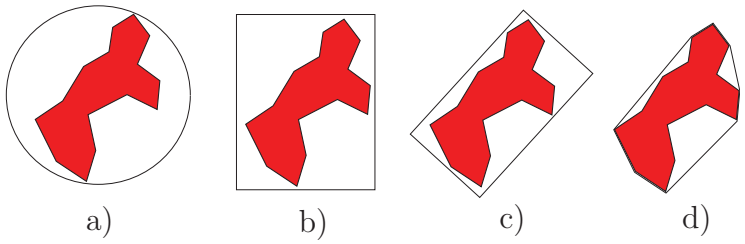


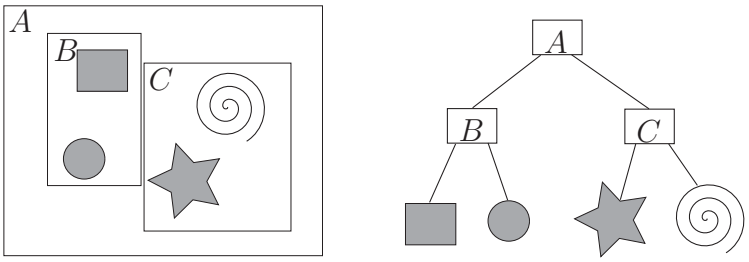
Figure 2: *Examples of BVs in order of increasing collision query computational cost: a) sphere, b) axis aligned bounding box (AABB), c) object aligned bounding box (OBB), d) convex hull.*

2.4 The BVH approach

A technique that can be used to increase computational efficiency for resolving collisions with convex decomposed concave bodies is through the use of a BVH. To create a BVH, the bounding volumes (BV) must first be assigned to individual convex sub-objects; these are the leaf-node BVs of the BVH. A BV is a geometry that completely encapsulates an object. Ideally, the encapsulation is as tight as possible. The BV is generally a much simpler geometry than the geometry it bounds. Collision queries can generally be performed more efficiently using the simplified BV geometry than the original geometry which it bounds. If the BV is not in collision, the bounded geometry is not in collision either. If the BV is in collision, an exact collision check must be performed on the true geometry bounded by the BV to determine if there is a physical collision or a false positive was indicated by the BV. Thus, BVs effectively aid in pruning out unnecessary and computationally expensive exact collision checks.

Bounding volume collision queries are designed to be less computationally expensive than queries on the geometries which they contain. Notable examples of BVs are spheres, axis aligned bounding boxes (AABB), object aligned bounding boxes (OBB), and convex hulls. These examples are illustrated in Figure 2. While the computational cost increases with complexity of the BV, a more complex BV will approximate the geometry of the physical object more accurately and will return fewer false positives and result in fewer exact collision checks. Consequently, it is not always obvious which BV should be used in a BVH to optimise accuracy or execution speed. Details on collision queries between individual bounding volume types are reviewed in [5].

A BVH is a tree structure whose nodes are BVs that bound 2 or more other bounding volumes or base geometries, as shown in Figure 3. The leaf nodes of a BVH are usually the geometry of interest in querying for collisions as shown in Figure 2. In this work, the leaf node is a BV which bounds a single convex polyhedron or a cable segment. The use of this tree structure reduces the number of expensive exact collision checks by pruning out groups of geometries from additional queries if their respective BVs are



a) b)

Figure 3: *An example bounding volume hierarchy for 4 objects. The bounding volumes are AABBs.*

not in contact.

An exhaustive collision query search for one external BV against all the leaf node BVs of a BVH has a complexity of $O(N)$, where N is the total number of leaf node BVs. With a BVH, collision checks need only be conducted on the BV geometries whose parent BVs are returning positive for collisions. Only the few leaf nodes which are likely colliding will have collision queries performed on them. The average, best, and worst case number of BV collision checks when traversing a binary BVH tree are $O(logN)$, $O(1)$, and $O(NlogN)$, respectively [26].

In most cases, particularly for deformable geometries, the BVs must be updated and potentially rebuilt at every time step. The complexity of rebuilding or updating the BVH, without considering the need to balance the tree, is an $O(NlogN)$ operation [26].

An expression that describes the expected computational cost of a BVH strategy is [5]:

$$T = N_V C_V + N_P C_P + N_U C_U + C_O \tag{1}$$

where T is the total cost, N_V is the number of bounding volumes queried for collisions, C_V is the cost of a collision query between two bounding volumes, N_P is the number of primitive pairs (leaf nodes) tested, C_P is the cost of testing a primitive pair, N_U is the number of nodes to be updated, C_U is the cost of updating each node, and C_O is a one time processing cost such as a frame transformation. Of these variables, N_V and N_P can be reduced through the use of tighter fitting bounding volumes while the cost of the collision queries C_V and C_P can be reduced through fast intersection tests of simpler, looser fitting bounding volumes. Striking a good compromise is a challenge of designing a BVH collision detection system.

The BVH approach has been used to successfully resolve self-interference with cables [22]. A typical approach which uses a BVH consists of a set of N_{seg} cable segments, which encapsulate a set of N_{seg} BVs. The topology of a cable is exploited when

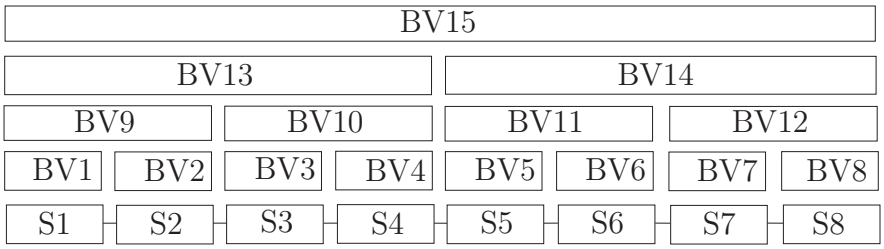


Figure 4: *The bounding volume ($BV1$-$BV7$) hierarchy for the discretised segments of a cable ($S1$-$S8$). Note: this figure is an adaption from a Figure in [1].*

constructing a BVH since adjacent cable segments will always remain adjacent. Figure 4 shows an example of the cable BVH with 8 linked cable segments ($S1$-$S8$) and the hierarchy of BVs ($BV1$-$BV15$) built around the cable's topology. The hierarchy is fixed for the duration of the simulation, though the bounding volumes may or may not be variable. Spherical bounding volumes can be used because detecting collisions and updating the bounding volume hierarchy is inexpensive [22].

2.5 The SGO approach

Optimisation methods have been used succesfully for collision detection and resolving the MSD. The optimisation formulation of the MSD problem for convex polyhedra is formulated as [9, 27]:

$$\min_{\mathbf{p}_1,\mathbf{p}_2} \quad (\mathbf{p}_1 - \mathbf{p_2})^T(\mathbf{p}_1 - \mathbf{p_2}) \tag{2}$$

subject to:

$$\mathbf{A}_1\mathbf{p}_1 - \mathbf{b}_1 \geq 0 \tag{3}$$

$$\mathbf{A}_2\mathbf{p}_2 - \mathbf{b}_2 \geq 0 \tag{4}$$

where $\mathbf{A}_i$ is a $m \times 3$ matrix of face normals for object i, m is the number of faces on object i, $\mathbf{p}_i$ is a 3×1 vector of the position of a point on or in object i, $\mathbf{b}_i$ is a $m \times 1$ vector of scalar distances which completes the set of plane equations. The constraints represent a set of bounding surfaces. If the constraint is violated, the point $\mathbf{p}_i$ lies outside of the object.

This approach, coupled with a temporal coherence strategy, can provide the absolute minimum separation distance between two convex polyhedra in constant time ($O(1)$). When considering concave geometries, multiple minima may arise. As a result, depending on the starting points, different minima may be resolved from local optimisation methods as indicated in Figure 5.

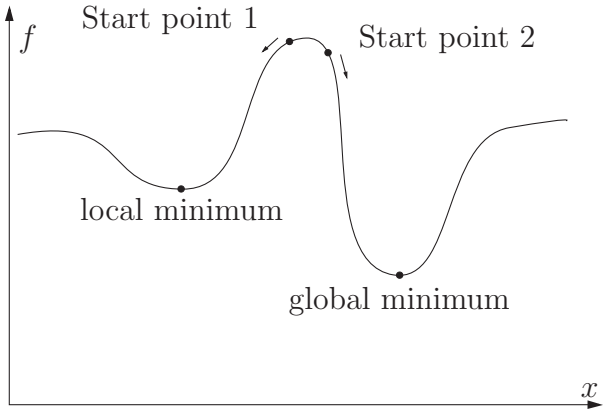


Figure 5: *Local versus global minima. This figure is adapted from [2].*

Collisions with concave geometries using convex optimisation methods can be resolved in several steps. First, convex decomposition must be applied to concave geometry. Next, local optimisation routines are used to find the MSD for the convex sub-objects. The absolute minimum of this process is the MSD [28]. A BVH of the convex sub-objects reduces the computational cost of this process.

Stochastic global optimisation methods have been used successfully to resolve the minimum separation distance problem between concave geometries [2, 10]. The concave geometry's surface is discretised using a mesh, where the points $\mathbf{p}_1$ and $\mathbf{p}_2$ on each object are limited to the vertices of the mesh. This reformulates the minimisation problem as an unconstrained combinatorial one. The MSD solution is approximate since it is limited to discrete points on the geometries. An unconstrained continuous local optimisation routine is usually used to determine the exact MSD solution using the combinatorial solution as an initial guess.

Global optimisation routines cannot guarantee the return of the absolute minimum. However, exploiting temporal coherence considerably reduces the chances of missing a collision. Time domain simulation of cables requires a small time step, which results in small discrete changes in displacement. This means that between consecutive time steps, the MSD solution will change very little. By assuming temporal coherence, the MSD solution from the previous time step can be used as an initial guess for the subsequent time step. This has two benefits for concave distance minimisation. First, since the trial solution begins at the minimum, very few optimisation iterations are needed, leading to $O(1)$ complexity. Second, since the minima from previous time steps are remembered for future time steps, they will not be missed in future evaluations.

Depending on the complexity of some arbitrary concave geometry, the probability of missing a collision on a single timestep may be high using the stochastic approach. However, by exploiting temporal coherence, the probability of missing a collision after

a number of time steps have elasped is greatly reduced because new minima are always sought and established minima are effectively kept in memory as initial guesses for future time steps. The result is that as the number of elapsed time steps increases, the probability of missing a collision decreases significantly. Even if a collision is missed on a single time step, there is a significant probability that it will be detected in ensuing time steps. The literature exhibits several sucessful uses of global optimisation to find the MSD [18, 19, 20].

A global optimisation approach was considered for use in this work in Section 3 [20]. The method showcased a combinatorial optimisation stage which minimised the distance between a discrete set of points on the cable and the polyhedron. The minimised combinatorial pairs were then sorted by distance, and identical pairs were pruned. If the separation distance between a minimised combinatorial pair is small enough to warrant a more accurate MSD solution, an unconstrained local optimisation method was used to establish the solution within some tolerance.

3 Detailed description of selected narrow phase cable collision detection methods

To resolve cable collisions, two collision detection techniques have been selected for implementation:

- convex-decomposed bounding volume hierarchy (BVH) and,
- stochastic global optimisation (SGO) methods.

The selected approaches are similar to those described in the literature [20, 22]. Due to the complexity of the problem and techniques, it was not obvious which method was preferable. Sections 3.1 and 3.2 discuss the methods in detail and performs a comparison between their computational costs. With both methods implemented, the computational efficiency of both methods was quantified; these results are presented in Sections 10.3 and 10.4. These results made it clear that the BVH method implementation performed the best and thus the SGO implementation has since been deprecated.

After the implementation of both methods was completed, some time was spent investigating efficient broad phase collision detection strategies. During the investigation, it became clear that if a BVH for the cable existed, the information required to perform spatial partition was already available. In contrast, if an SGO method was used, the same required information would need to be calculated, thereby adding to computational expense. This extra expense is not considered in the computational cost analysis found in Section 10.4, though it is an additional benefit to using a BVH approach. Some discussion on the broad phase collision detection strategy investigation is presented in Section 4.

3.1 Bounding volume hierarchy

The complexities of various required operations of a BVH approach are summarised in Table 1. To create a BVH for a cable, a discretisation of N_{seg} linear cylindrical segments is utilised as illustrated in Figure 6. Increasing the number of segments increases the accuracy of representation of the cable and the resulting collision detection solution. This accuracy comes at the cost of increased computational expense and simulation execution speed. A BVH of N_{seg} cable segments will require $N_{seg}logN_{seg}$ bounding volumes for a binary tree structure as shown in Figure 4. The first operation is to build the BVH, which only needs to be completed once and is therefore a pre-processing step.

At every time step, each BV of the BVH must be updated to ensure they continue to bound their cylindrical line segment. This operation has a complexity of $O(N_{seg}logN_{seg})$ since there are $N_{seg}logN_{seg}$ bounding volumes that need to be updated. If simple bounding volumes are used, the cost of updating the bounding volumes is relatively low. When

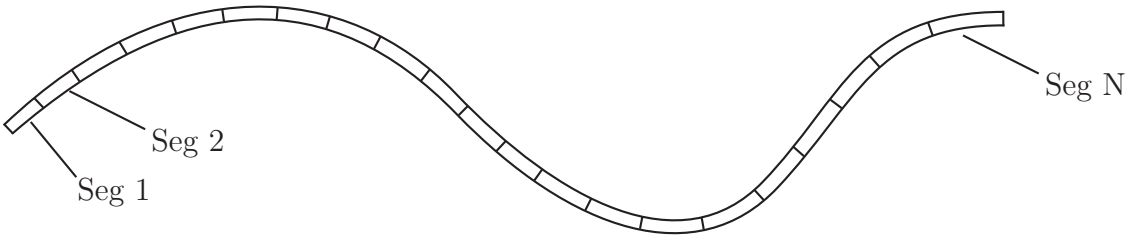


Figure 6: *A continuous cable discretised into N_{seg} segments which is encapsulated by a spherical bounding volume.*

Table 1: *The complexity of the tasks required by the BVH approach. The desired tolerance in arc length location of the minima is represented as s_{tol}.*

Task	**Complexity**
building BVH (pre-processing)	$O(N_{seg}logN_{seg})$
updating BVH (every time step)	$O(N_{seg}logN_{seg})$
collision queries	$O(1)$ best case $O(logN_{seg})$ average case $O(N_{seg}logN_{seg})$ worst case
line-segment vs. polyhedron MSD query	$O(1)$ (temporal coherence) $O(logN_p)$ average case $O(N_p)$ worst case

more complex bounding boxes are used, such as OBBs or convex hulls, the cost of a single bounding volume update will be higher. However, the computational costs of specific BV types are not being considered. The BV type used is AABB since fits are tighter than spheres and are reasonably simple to update and check for collisions.

There are a number of methods to choose from for building the BVH as well as for descending the hierarchy during a collision check. This work uses a bottom-up BVH construction approach as well as a hierarchy descent rule based on largest volume. Descriptions of these two methods including benefits are available in the literature [5].

When testing the cable for collisions against a convex polyhedron, there is an average case complexity of $O(logN_{seg})$ BV collision checks. This is because if a BV returns negative for collisions, the BVs further down that branch need not be processed. In the worst case, every single BV will be tested leading to $O(N_{seg}logN_{seg})$ collision checks including $O(N_{seg})$ leaf-node collision checks. The average-case complexity is an important consideration for simulation execution speed, particularly for simulations with real-time requirements.

3.1.1 Leaf node collision checks

Checking leaf node collisions is the most expensive operation of the BVH approach. At the end of the collision detection process, N_{coll} leaf-node BVs returned positive for collision. The bounding volume collision query does not return enough information to determine a reasonable contact force, so a more detailed collision query must be effectuated. To accomplish this, the leaf-node's cylindrical line segments are tested for collisions directly against the polygons of the convex polyhedron of the other object. The method is an MSD query between a line segment and a polyhedron. Temporal coherence can be used to reduce computational cost ($O(1)$ complexity).

3.1.1.1 Line segment and polyhedron MSD query

When a cable's leaf node BV and polyhedron's leaf node BV are in collision, the separation distance between the line segment and the polyhedron must be determined. Computing the exact distance is required to determine if the cable is truly in collision with the polyhedron. The exact collision information is also used as input for the Hunt-Crossley contact dynamics model.

The MSD is the distance between the line segment and the closest polygon of the polyhedron. If it is the first distance query between the line segment and the polyhedron, a polygon from the polyhedron is selected at random to begin the process. Otherwise, the closest polygon returned in the last MSD query for that segment is used.

Because edges are shared by two polygons, a resulting MSD that lies on an edge of a polygon results in an additional distance query on the adjacent polygon. If the distance query between the polygon and the line segment returns a MSD that lies on one of the polygon's edges, the line segment is then tested against the polygon that lies adjacent to that edge. This process is continued until the closest polygon and MSD are found. Crawling the mesh in this fashion to find the closest polygon is at worst an $O(logN_p)$ operation and at best an $O(1)$ operation, where N_p is the number of polygons on the polyhedral mesh.

Though this operation is reasonably efficient, particularly when temporal coherence is exploited, it can still be a significant computational burden when many leaf node BVs are colliding. Thus, it is important that the collision detection strategy minimises the number of false positive leaf node collisions.

3.1.1.2 Line segment and polygon MSD query

The line segment and polygon distance query discussed here is limited to convex polygons with co-planar vertices. Determining the MSD between a line segment and such a polygon requires several checks and measurements. The procedure is as follows:

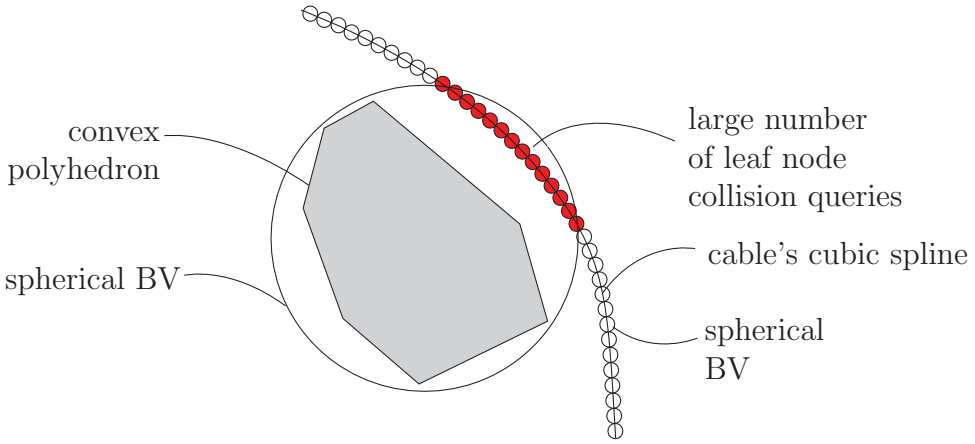


Figure 7: *Example of a loose fitting bounding volume around polyhedron leading to many false positives and leaf node collision checks.*

1. Check if either of the line segment's end nodes lies within the polygon boundaries when projected onto the polygon's plane.
2. If so the distance between the node and the polygon plane is determined.
3. If both end nodes project inside of the polygon's boundaries, the end node with the shortest distance to the polygon's plane represents the MSD and the search is terminated here.
4. Otherwise, the line segment's distance against the edges of the polygon must also be tested; these consist of line segment to line segment distance checks.
5. The shortest distance between the end nodes and the polygon plane as well as the line segment and the edges of the polygon represents the MSD

It is important to remember which edge, if any, the MSD lies on. This information is used to search for the closest polygon of a polyhedron to the line segment as described in Section 3.1.1.1. Descriptions of efficient implementations of these distance queries, including line segment to line segment queries, are given in the literature [5].

3.1.2 BV for convex polyhedra

It is important that a rigid body's convex polyhedron's bounding volume fits tightly. The size of the convex polyhedra will be relatively large compared to the size of the cable line segments and diameter. Such a situation can lead to many leaf node collision checks as demonstrated in Figure 7. Tighter fitting BVs or a BVH of polygons may be required.

A BVH could be created for the convex polyhedron that consists of its leaf node BVs

that bound its individual polygons. This would lead to a larger number of BV collision queries, but it would considerably reduce the number of line-segment to polygon distance queries. This approach would have the advantage that after the BV queries, the exact polygons to test against the line segments are known. Thus no mesh crawling is ever needed. This work has not considered the use of a BVH of polygons for a polyhedron, though one could be implemented in the future. Because the BVH would need updating every iteration, it is unclear if the performance gains would outweigh the added overhead of this approach. For now, this work relies on the assumption that the AABB is reasonably tight fitting around the polyhedron.

3.2 Stochastic global optimisation

The approach discussed in this section is closely related to a method called multi-local search [29]. The method begins with an unconstrained combinatorial optimisation stage which provides an approximate solution. This is used as input for the following local optimisation search stage which refines the solution to find the exact MSD.

3.2.1 Unconstrained combinatorial stage

The algorithm begins by discretising the cable into a set of N_{res} equally spaced points along the cable as shown in Figure 8. The number of points N_{res} should be high enough such that no more than one minimum can lie within two points, but low enough to reduce the number of combinatorial optimisation steps, which is on average $O(logN_{res})$. After discretising the cable into N_{res} points, N_{rand} combinatorial MSD pairs are randomly selected as the number of pairs to optimise: $N_{pairs} = N_{rand}$. For this case, a combinatorial MSD pair consists of one of the N_{res} points on the cable and one of the polygons on the mesh of the convex polyhedron. The distance between each pair is minimised by combinatorial optimisation using the following steps:

1. start with the point in the pair belonging to the cable
2. compare the distance from the point on the cable to the closest polygon on the convex polyhedron against the distance between the two neighboring discrete points on the cable and their closest points on the closest polygons of the convex polyhedron.
3. if one of the distances of the neighboring points is shorter, the point on the cable is changed to the point with the shortest distance.
4. if the pair has changed since step 2, return to step 2, else pair is minimised.

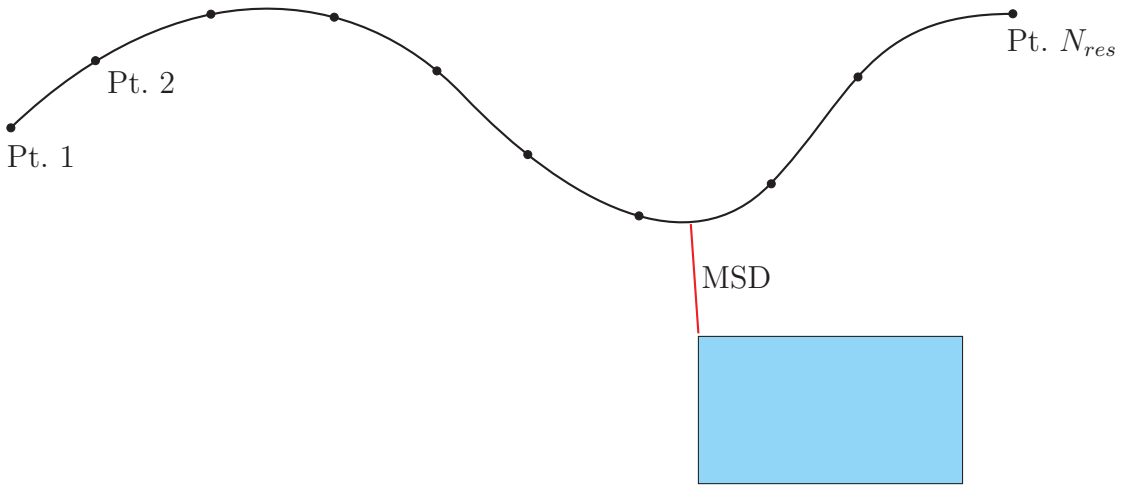


Figure 8: *The discretisation of a cable into N_{res} combinatorial points, where N_{res} is chosen high enough that only a single minimum can potentially exist between two adjacent cable combinatorial points and a convex polyhedron.*

3.2.1.1 Finding the closest polygon for random pairs

In this combinatorial optimisation step, the possible worst case scenario would be needing to check all N_{res} points on the cable. Every distance check between a cable point and the polygon of a mesh requires the traversal across of the polygon mesh surface to find a corresponding MSD. Finding the closest polygon to some point in space takes in the average and worst case $O(logN_p)$ point-polygon distance queries by making use of mesh connectivity information. By employing temporal coherence, this can be reduced to $O(1)$. However, temporal coherence can not be employed for the first point-polygon distance check of a random pair. Because of this, on average, minimising the distance between a random pair will require $O(logN_{res} + logN_p)$ distance checks.

3.2.2 Exploiting temporal coherence

At the end of the combinatorial optimisation stage, there are N_{pairs} minimised pairs. To reduce future computational costs, the minimised pairs should be sorted by distance and duplicate pairs should be eliminated. The best sorting algorithms have a complexity of $O(N_{pairs}logN_{pairs})$ operations, which for this case are considered low cost. Once pruned, one is left with N_{pruned} unique, minimised distance pairs. These pairs are saved for the next time step for use as the initial solution to the combinatorial optimisation stage. Due to temporal coherence, minimising the N_{pruned} pairs in the next time step will require $O(1)$ distance checks for each pair, which is a negligible computational cost.

To continue searching for new minima, N_{rand} new random pairs are selected at every time step. The total number of pairs for the next iteration is $N_{pairs} = N_{pruned} + N_{rand}$.

In general, the total number of new random pairs used is problem-dependent. However, due to small time steps inherent in dynamic time domain simulations and temporal coherence, the number of new random pairs can be kept low thereby relying on numerous time step iterations to catch and remember new minima. In every time step except for the first, there will be N_{pruned} and N_{rand} pairs to minimise. The cost of minimising N_{pruned} pairs is negligible, while the cost of minimising N_{rand} pairs is kept low by using few pairs and spreading the burden of finding new minima over time.

3.2.3 Local search

Once a set of N_{pruned} ordered minimised pairs is obtained, a continuous unconstrained local optimisation routine is used to more accurately resolve the MSD for pair distances that are within a minimum tolerance. The local optimisation method implemented for this work is the method of golden sections. The bounds of the golden sections search method is chosen to be between the discrete points adjacent to the combinatorial minimum. This method has a computational complexity of $O(\frac{N_{pruned}}{s_{tol}})$, where s_{tol} is the convergence tolerance for the method.

This local search method is relatively inefficient due to the iterative nature of the method and the number of point to polygon distance checks which must be effectuated. There is potential to improve performance by implementing a local search method similar to the one described in the BVH based method where cable line segments are tested against polygons.

3.2.4 Complexity of SGO method

Table 2 shows the complexities of various required operations of an SGO method to resolving collisions. Minimising random pairs can be moderately expensive. However, the number of random pairs can be kept low by exploiting temporal coherence. For temporal coherence, large numbers of minimised pairs can be remembered and optimised at every iteration with negligible costs compared to random pairs. This SGO method returns exact separation distance minima, though the global minima cannot be guaranteed. The probability of missing a collision is driven to insignificant levels by exploiting temporal coherence.

It was unclear whether BVHs or global optimisation methods would be the most efficient solution for the Launch and Recovery scenario. Global optimisation routines were less complex and can be implemented quickly and so it was selected for implementation first. The SGO method demonstrated the feasibility of cable collision resolution for the cable model used by the SMS API. Extensive testing showed Launch and Recovery simulations using BVH executed significantly faster than SGO and so further development on SGO ceased.

Table 2: *The complexity of the tasks required by the use of a global optimisation based approach to collision detection of cables. The desired tolerance in arc length location of the minima is represented as* s_{tol}.

Task	**Complexity**
Minimising 1 combinatorial pair	$O(1)$ (exploiting temporal coherence - best case) $O(logN_{res} + logN_{poly})$ (random pair - avg case) $O(N_{res})$ (random pair - worst case)
Sorting minima by distance	$O(N_{pairs}logN_{pairs})$
Local optimisation N_{pruned} pairs	$O(\frac{N_{pruned}}{s_{tol}})$ (method of golden sections)

4 Broad phase collision detection

Research was conducted to determine an efficient broad phase collision detection strategy alternative to the brute force method. The software infrastructure required to perform broad phase collision detection was created to facilitate its future implementation when the new collision detection system was created. The new collision detection code infrastructure and other modifications made to the SMS API over the course of this project are discussed in Section 5.

Code profiling of the L&R simulation indicated that a more efficient broad phase collision detection strategy would not significantly reduce computations or increase simulation execution speed. This is because there are only a few objects that could possibly collide with each other during the course of this simulation. These objects are clustered fairly tightly together and some are long, thin, and can cross much of the simulation space. This makes it difficult to perform spatial partition which could outperform the brute force approach. For this specific problem, the added overhead of an efficient broad phase collision detection scheme may outweigh the potential benefits.

The brute force broad phase collision detection method was kept and combined with a simple ignore list to maximize performance. The ignore list tells the collision handler to ignore collisions between individual pairs of objects. This way, it can be ensured that cables would never be tested against other cables. For the case of the L&R simulation, the collision handler was also told to ignore collisions between the rescue boat and any cables. The ignore list provided significant performance gains.

It should be noted that for general use case simulation scenarios, particularly with large numbers of simulation objects, an efficient broad phase collision detection strategy could be warranted.

5 Modifications to collision detection code infrastructure

5.1 The CollisionHandler class

The original SMS API collision detection architecture was designed to support collision detection between a `Payload` object and the ship's hull. This project introduces the detection of collisions between cables and the ships deck as well as potentially with payloads. Because the collision detection system now requires the ability to handle collisions between more than just two objects, the existing collision detection system had to be redesigned to handle multi-body collisions.

To address this, a new class was created called the `CollisionHandler`. The `CollisionHandler` is a class designed to manage the detection of collisions between `SimObject`s. The `CollisionHandler` class inherits from the `SimObject` class in order to be able to manage the integration rate of the `SimObject`s it is responsible for. In short, it ensures the `SimObject`s it is handling are numerically integrating together in lock step. Adding `SimObject`s to it for collision detection creates master-slave relationships between the `CollisionHandler` and the `SimObject`s. The `CollisionHandler` must always be the master.

Before adding `SimObject`s to the `CollisionHandler`, first one must add a `CollisionObject` to the `SimObject`. The `CollisionObject` object provides a description of a `SimObject`'s collision geometry. Afterward, all the `SimObject`s with `CollisionObject`s are added to the CollisionHandler which will manage any and all collisions.

The `CollisionHandler`'s main job is ensuring managed `SimObject`s are integrating in lock step. It must however rely on the use of another class to manage broad phase collision detection, the `BroadPhaseCollisionDetection` class. The `CollisionHandler` provides its `BroadPhaseCollisionDetection` object with the `CollisionObject`s of the `SimObject`s it is managing. The `BroadPhaseCollisionDetection` class handles updating the `CollisionObject`s as necessary. Before the dynamics of the `SimObject`s are evaluated for a particular time step, the `CollisionHandler` object will get the `BroadPhaseCollisionDetection` object to detect all collisions. The `CollisionObject` for each SimObject retains information about any collision it is in. The `SimObject`s have access to the results of any detected collisions and will use it to compute contact forces when its dynamics are evaluated. Narrow phase collision queries are computed by the `CollisionObject`s themselves by passing a `CollisionObject` to another as described in [3].

5.2 The CollObjCubicSpline class

There were a few existing `CollisionObject`s from previous work, particularly the `CollObjConvex` class and the `CollObjConvexDecomp` class [3]. A new `CollisionObject` was needed to handle collisions with cubic spline cables. To handle this, the `CollObj-CubicSpline` class was created. It is responsible for handling narrow phase collision queries between itself and `CollObjConvex` or `CollObjConvexDecomp`.

The SGO method described in Section 3.2 as well as the BVH method described in 3.1 were implemented inside this class. Collisions between cables and any number of rigid bodies can be resolved using this class. Demonstrations of the use of this class for cable collision resolution can be found in Section 10.2.

6 Contact dynamics model

This section describes how the contact forces are determined when collisions occur. There are a plethora of ways to model contact forces [30]. The model selected for this work is the Hunt-Crossley model. It offers a reasonable balance between accuracy and execution speed.

The model requires relatively few computations since the metric which the model depends on is the penetration distance rather than a more complex metric such as the interference volume. One limitation of the current implementation of the model is the assumption of constant contact interface geometry. This can significantly affect contact dynamics fidelity. However, because the simulated cables are thin and compliant, errors in the dynamics of the collision are tolerable as the primary objective is to realistically maintain separation between cables and polyhedrons and not to accurately compute local stresses due to contact loads.

The Hunt-Crossley normal force model is based on the Hertzian theory of general contact with a superimposed penetration depth dependent damping component [30]. The normal contact force $\mathbf{f}_n$ for the Hunt-Crossley model is described as:

$$\mathbf{f}_n = (k\chi^n + \lambda\chi^p\dot{\chi}^q)\hat{\mathbf{n}} \tag{5}$$

where k is the equivalent stiffness coefficient, χ is the penetration depth, $\hat{\mathbf{n}}$ is the normal contact force direction aligned parallel with the vector representing the penetration depth, and n, p, and q are geometry and material constants. The parameters p and q are generally set to n and 1, respectively [30]. For the case of impacting spheres, n is $\frac{3}{2}$. These commonly used values for n, p, and q are used as an approximation for this work, even though the geometries involved are cylindrical cubic splines and flat polygons with sharp corners.

The damping factor λ can be difficult to determine without experimental data. Conveniently, when p and q are set to n and 1, respectively, a closed-form solution for the damping factor as a function of the coefficient of restitution can be resolved based on the principle of conservation of energy [31]. By making this assumption, equation 5 becomes:

$$\mathbf{f}_n = k\chi^n(1 + a\dot{\chi})\hat{\mathbf{n}} \tag{6}$$

where a is the ratio of the damping factor to the stiffness coefficient, $a = \frac{\lambda}{k}$. There are a number of works that discuss the relationship between λ, or a and the coefficient of

restitution [31, 32, 33]. For the L&R simulation discussed in Section 10.6, a damping ratio of $a = 0.7$ was used with good results.

The penetration depth between a volumetric cubic spline cable and a polyhedron can be determined if the distance d between the cubic spline geometry (the center line of the volumetric cubic spline) and the polyhedron is less than the cable's radius R. The penetration depth can then be defined as:

$$\chi = R - d. \tag{7}$$

The rate of change of the penetration depth $\dot{\chi}$ can be determined as the normal direction component of the relative velocity between the two contact points, one on each geometry.

7 Human deck hand using the winch model

Numerical modelling of human performance capabilities is a complex problem. A literature review was completed to identify the human performance capabilities for a deck hand pulling on a rope. Information was found that indicated a reasonable safe maximum human horizontal pulling force [34]. However, only general information was available with no further guidance on slipping or tipping behavior that may result from dynamic tensions in the rope. In addition, no information was found on the influence of deck acceleration on the deck hand behavior.

Due to the potential complexity and lack of information to help characterise deck hand behavior, only a simple model was produced. The previously developed `Winch` model was employed to represent the deck hand using tension mode exclusively. To model the deck hand, the `Winch`'s PID payout controller was configured using a high P-gain with a payin and payout acceleration limit. The high P-gain ensures that the payin/payout rate will always be able to at least match the tag line and attached rigid body velocity. The acceleration limit acts like a filter that prevents the controller from inducing rapid payout velocity oscillations that can destabilize the system.

The acceleration limits prevent the controller from setting severe changes in payout or payin speed. During dynamic simulation, this can result in tension temporarily increasing or decreasing around the tension setpoint. The tension setpoint was set to a value of maximum human horizontal pull force.

There are many complex behaviors that are not captured by this simple model. The deck hand may move around on the ship, the line may slip, or the deck hand may let go under certain circumstances. These and other effects may have an important influence on the behavior of the tag lines.

8 Launch and recovery simulation challenges

With the new additions and significant modifications made to the SMS API, some time was required to update the launch and recovery simulation. One particular design challenge that arose was with the original master-slave relationship management code of the `SimObject` class. The master-slave relationship management code became obsolete with the introduction of the new `CollisionHandler` class. This is described below in Section 8.1. A potential complication that could arise through the use of the BVH based approach is discussed in Section 8.2. Minor adjustments made to the simulation configuration itself are discussed in Section 8.3.

8.1 Addressing master-slave SimObject relationships

The `CollisionHandler` must be the ultimate master of all other `SimObject`s in the simulation in spite of any dynamic connections such as those between cables and rescue boats. In previous simulations, managing the master-slave relationships was simple; most of the simulation objects were their own masters, except for cables attached to a rigid body or a rigid body attached to a cable that is already a slave.

With the introduction of the `CollisionHandler`, all individual `SimObject`s being considered for collisions, along with any slaves they may have, become slaves of the `CollisionHandler` object. The master-slave relationship hierarchy at the beginning and at the end of the launch and recovery simulation is shown in Figure 9 a). This configuration is necessary because the `CollisionHandler` must control the integration rate of the `SimObject`s whose collisions it is managing. The `SimObject`s must evolve through time together for the dynamics of the collisions to be properly resolved. This added a new level of complexity for managing master-slave relationships, particularly when `SimObject`s are connected/disconnected on the fly and attached to other objects further up or down the master-slave hierarchy. A new more general method of handling the master-slave relationships was developed to allow live cable connections and disconnections.

Figure 9 b) shows the hierarchy when both the stern and bow tag lines, as well as the crane's cable, are attached to the rescue boat. It also shows how the cables now keep track of other objects under which they have an indirect master-slave relationship. That is, the cable is connected to both the `MarineVehicle` as well as the `Payload` and must be a slave to both in the hierarchy. A direct master is the `SimObject` responsible for ensuring that the dynamics of the slave are being evaluated. The indirect master is also attached to the slave; however, it is not responsible for ensuring that the slaves dynamics are evaluated. If a direct master is disconnected from its slave, and if the slave has an indirect master, then the indirect master takes over the role of the direct master.

A cable is now always the slave of any `SimObject`s to which it is connected to and cannot be the master of any other `SimObject`s besides itself. Figure 9 b) also shows that the `Payload` keeps the `CollisionHandler` as an indirect master. This is because if the Payload is disconnected from the cables, all master-slave relationships will be severed and the `CollisionHandler` must be reassigned as its direct master, as shown in Figure 9 a).

If a cable is a slave of two `SimObject`s, then all three `SimObject`s must be under the same hierarchy branch. The lowest `SimObject` is the direct master of the cable, while the highest `SimObject` is an indirect master. If the direct master is disconnected from the cable, then the cable's indirect master becomes its direct master again. They must be under the same branch to ensure that when an indirect master queries its slave for boundary condition information that the dynamics of the slave have been evaluated.

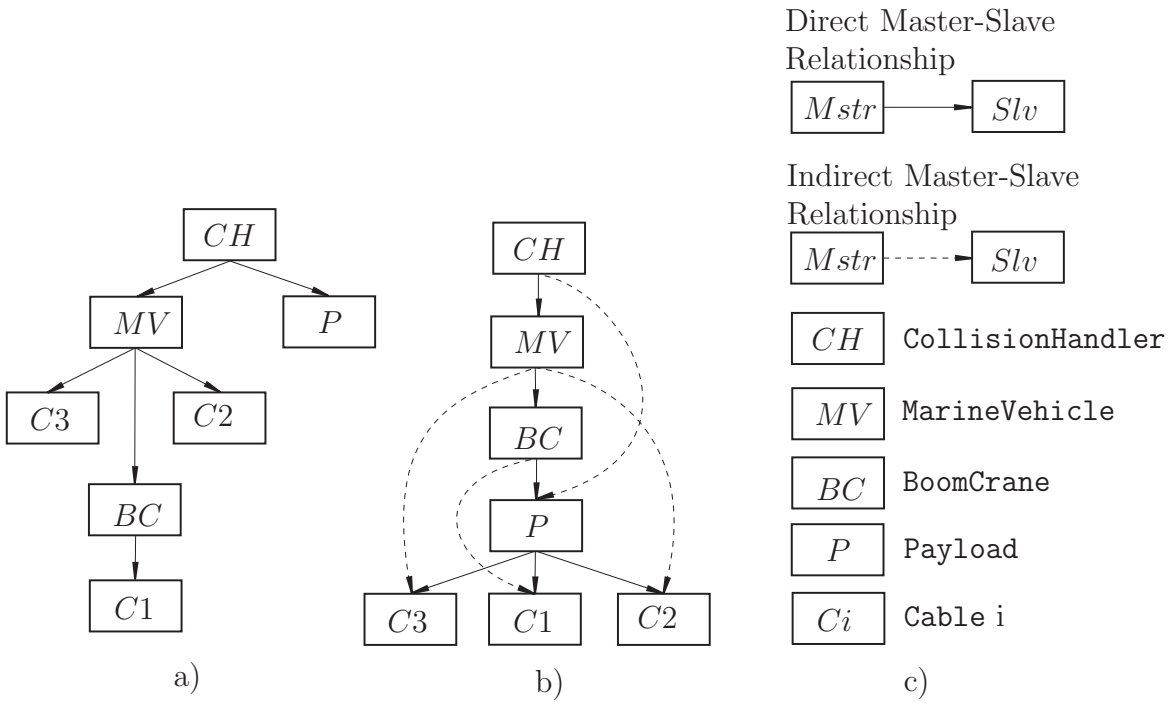


Figure 9: *a) The master-slave relationship hierarchy at the start and finish of the L&R simulation. b) The master-slave relationship hierarchy of the L&R simulation when all cables are attached to the* `Payload`*. c) A legend for this Figure.*

8.2 Complications due to coarse cable contact mesh

Regardless of the length of the cable, which can change over the course of a simulation, or the number of nodes used in the FEA model, the BVH based collision detection method will discretise the cable into a constant number of line segments as defined

at simulation initialisation. These line segments are used to test for collisions against other geometries. If the discretisation is too coarse, the cubic cable profile could be poorly represented as shown in Figure 10. The cable is always evenly discretised by a static number of line segments. Enough line segments should be specified to account for changing cable length through the course of a simulation.

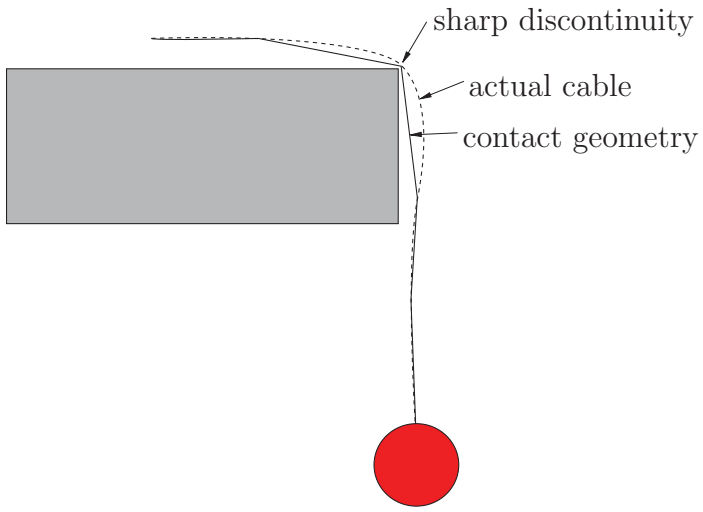


Figure 10: *An example of a coarse line segment discretization with a poor fit to the cable profile.*

Coarse discretisation of line segments can cause integration problems at sharp corners of polyhedrons, as illustrated in Figure 11. When two line segments form an "acute" angle and are in contact with a sharp corner, the minima and applied force direction can jump quickly from force direction labeled 1 to 2 as shown in Figure 11 between two consecutive time steps. The rapid change in direction can lead to strong perturbations in the dynamics of the cable and lead to time step reductions in the adaptive Runge-Kutta integrator. If the problem persists, the integration time step could underflow, causing a premature termination of a simulation. This problem can be avoided by adding more line segments to the discretisation.

8.3 Minor changes to the L&R simulation script and setup

Some minor changes were made to the scenario. The more notable changes are:

- ShipMo3D time series data has a resolution of 0.001 units. This produces discontinuous ship motions which was problematic for cable contact simulations. Cable contacts are sensitive to small changes in position due to stiffness of the contact

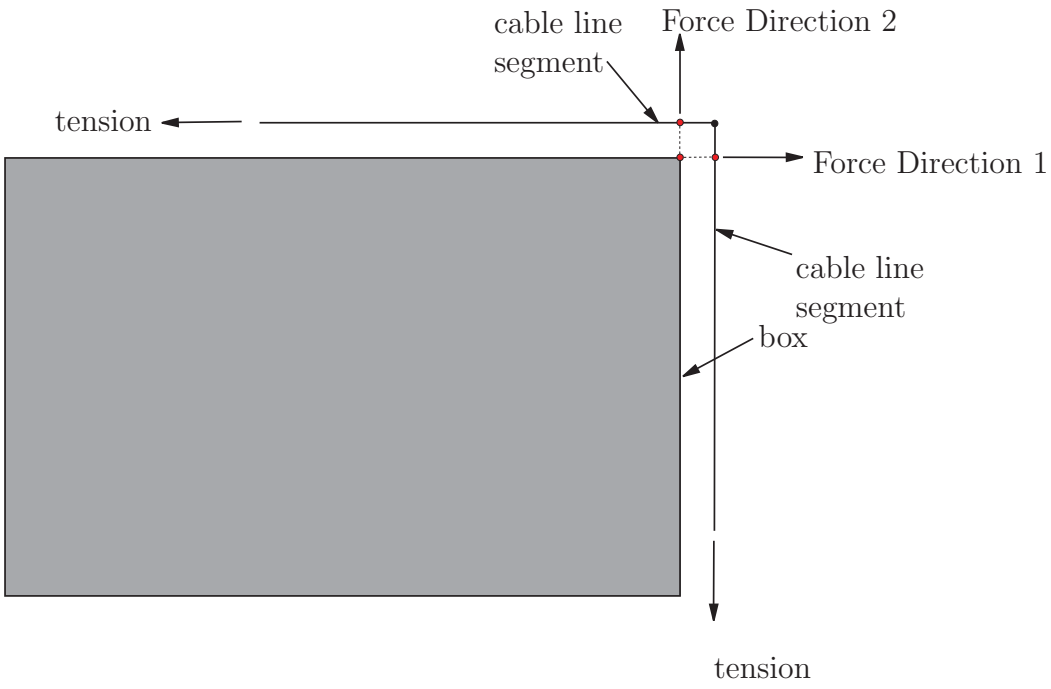


Figure 11: *Contact force oscillations induced by coarse cable collision mesh on sharp polyhedron corners.*

problem and the small cable diameters. To resolve this, ShipMo3D was modified to output time series data with a precision of 10^{-6} units.

- Tag line cables are now attached to the frigate instead of their end nodes being kinematically driven.
- The rescue boat is now facing forwards inside the cradle.
- The boom crane has been shifted closer towards the central axis such that it does not interfere with the tag lines.
- Tag lines were spread further apart to make them more effective at controlling the rescue boat.
- Tag line winch models were modified such that the tensions they can sustain reasonably models a human's capabilities (on the order of hundreds of Newtons).
- The collision handler was added to manage the collisions between the rescue boat, frigate and the tag lines.
- The ShipMo3D DeepSeaway dynamically linked library has been removed from the simulation and replace with a custom wave class which is identical to DeepSeaway. This resolved a few issues in the development environment caused by the use of the Common Language Runtime (required for using the C# based DeepSeaway dll).

9 Manual Updates

The project has both added some new functionality and altered some pre-existing functionality. This section serves to update the SMS API manual with the new and altered functionality. The interface changes made to provide collision geometry descriptions to the `SimObject`s are presented in Section 9.1 while the interface to the new `CollisionHandler` class is discussed in Section 9.2.

9.1 Adding `CollisionObject`s to `SimObject`s

There where a few modifications made to the interface of the `Payload` and `Cable` classes to improve how `CollisionObject`s are added. The functions to provide collision geometries to `Payload`'s and `Cable`s are provided in Section 9.1.1 and 9.1.2 respectively.

9.1.1 Adding collision objects to `Payload`s

Before adding a `SimObject` to the `CollisionHandler` for collision detection management, the collision geometries must be defined. The collision geometry for `Payload`s can be define using:

```
void Payload::AddOBJContactPolyhedron(string file_name,
double scaleX,
double scaleY,
double scaleZ,
ub::vector<double> &posOri,
double &stiffness,
double &damping,
string matType)
```

where `file_name` is the file name with path of the .obj file which contains the convex polyhedral geometry that describes the collision geometry, `scaleX`, `scaleY`, and `scaleZ` are geometry scaling factors about the geometry's local frame axes, `stiffness` and `damping` are the stiffness and damping coefficients of the collision material with units of Pa and $Pa \cdot s$ respectively. The vector `posOri` is a size 6 vector containing a length 3 vector describing the position of the collision geometry relative to the `SimObject`'s local frame, and 3 Euler angles describing the orientation of the collision geometry relative to the `SimObject`'s local frame. The string `matType` is the name of a material used to obtain friction coefficients via look up table. The friction coefficient look up table is rather sparse at this stage, it is recommended to simply use "steel" for `matType`.

9.1.2 Adding a collision object to a `Cable`

Before adding a `SimObject` to the `CollisionHandler` for collision detection management, the collision geometries must be defined. The collision geometry for `Cable`s can be define using:

```
void AddContactObject(double &stiffness,
double &damping,
std::string matType,
unsigned int discreteSubDivs)
```

where `stiffness` and `damping` are the stiffness and damping coefficients of the collision material with units of Pa and $Pa \cdot s$ respectively. The string `matType` is the name of a material used to obtain friction coefficients via look up table. The friction coefficient look up table is rather sparse at this stage, and it is recommended to simply use "steel" for `matType`. The parameter `discreteSubDivs` is the number of discrete line segments to use to describe the `Cable`'s collision geometry.

Currently, `stiffness`, `damping` and `matType` are being ignored for simplicity. Instead cable contact model properties have been hard coded to ensure separation is maintained until more time is devoted to fleshing out the interface.

9.2 The `CollisionHandler` class

Once the `SimObject`s have had their collision geometries defined, they are ready to be supplied to the `CollisionHandler`. The `CollisionHandler` will become that master `SimObject` over any other `SimObject` supplied to it and will handle their integration over time.

9.2.1 Creating a `CollisionHandler` object

The `CollisionHandler` is a `SimObject` and is instantiated like any other `SimObject`:

```
CollisionHandler::CollisionHandler(std::string init_file,
double t0,
bool file_output)
```

where `init_file` is the path and name of the `CollisionHandler`'s initialisation file, `t0` is the SimObject's initial start simulation time, and `file_output` is a boolean which turns on and off file output.

The `CollisionHandler` requires an initialisation file to be defined. Currently, the `CollisionHandler` has two parameters that must be defined in the initialisation file.

Each parameter currently only supports a single option. The initialisation file must contain exactly the following:

```
$BroadPhaseCollisionDetectionStrategy BruteForce
$BoundingVolumeType CollObjAABB
```

9.2.2 Adding SimObjects to the CollisionHandler

Adding a SimObject with defined collision geometry to a CollisionHandler can be accomplished as follows:

```
void CollisionHandler::AddObject(SimObject* obj)
```

where obj is a pointer to the SimObject being added.

9.2.3 Telling the CollisionHandler to ignore SimObject pairs

It is possible to instruct the CollisionHandler to ignore collisions between individual pairs of SimObjects that have been added to the CollisionHandler. This can be accomplished by calling the following CollisionHandler method:

```
void CollisionHandler::AddBruteForceIgnorePair(
SimObject* obj1, SimObject* obj2)
```

where obj1 and obj2 are pointers to a pair of SimObjects to ignore that have been previously added to the CollisionHandler.

9.2.4 Advancing Time

When a CollisionHandler is managing a set of SimObjects, the CollisionHandler is charged with managing their integration. The simulation of the SimObjects managed by the CollisionHandler can be advanced through time by calling the following CollisionHandler method, like any other SimObject:

```
int SimObject::AdvanceTime(double DeltaT)
```

where DeltaT is the amount of time by which to advance the simulation.

10 Simulation Results

This section presents the setup and simulation results for 6 simulations which were conducted after various milestones in the project were completed. Section 10.1 presents the results of a simulation demonstrating the new N-body collision detection capability of the `CollisionHandler` while Section 10.2 demonstrates the SGO cable detection capabilities showing a cable colliding with 2 boxes.

The overhead required by both the BVH and SGO method are quantified in Sections 10.3 and 10.4. Section 10.5 shows a demonstration simulation showing the collisions between a cable and a rigid body against the side of a ship. Finally, Section 10.6 discusses the current state of the L&R simulation, the main deliverable for this project.

10.1 N-body collisions

10.1.1 Setup

The purpose of this simulation is to demonstrate the ability of the `CollisionHandler` class to handle multiple `SimObject`s with `CollisionObject`s in one simulation. The simulation consists of 7 Payload objects each fitted with cuboid shaped `CollObjConvex` `CollisionObject`s. One `Payload` object acts as a ground plane. It has a large flat cuboid `CollisionObject` and is kinematically fixed in space. The other 6 `Payload`s are assigned a 1.0 cubic meter cube `CollisionObject` and are arranged in a pyramid configuration as shown in Figure 12. They are initially not in contact with each other and begin the simulation with gaps between them. The simulation consists of letting them fall onto the ground plane and settle by resting in collision with one another, held up in the pyramid configuration.

The mass of each Payload was set to 7000 kg and appropriate mass moments of inertia for a solid cube were assigned. The material properties for the volume based contact dynamics model employed [3, 31] were set to a Young's Modulus of 1.0E7 Pa and a damping coefficient of 1.0E6 Pa/s.

10.1.2 Results

A picture of the simulation in its final resting state can be seen in Figure 13. Note that the top `Payload` is applying loads on the two Payloads below it giving them a small angular deflection. Using a stiffer material property would alleviate this issue.

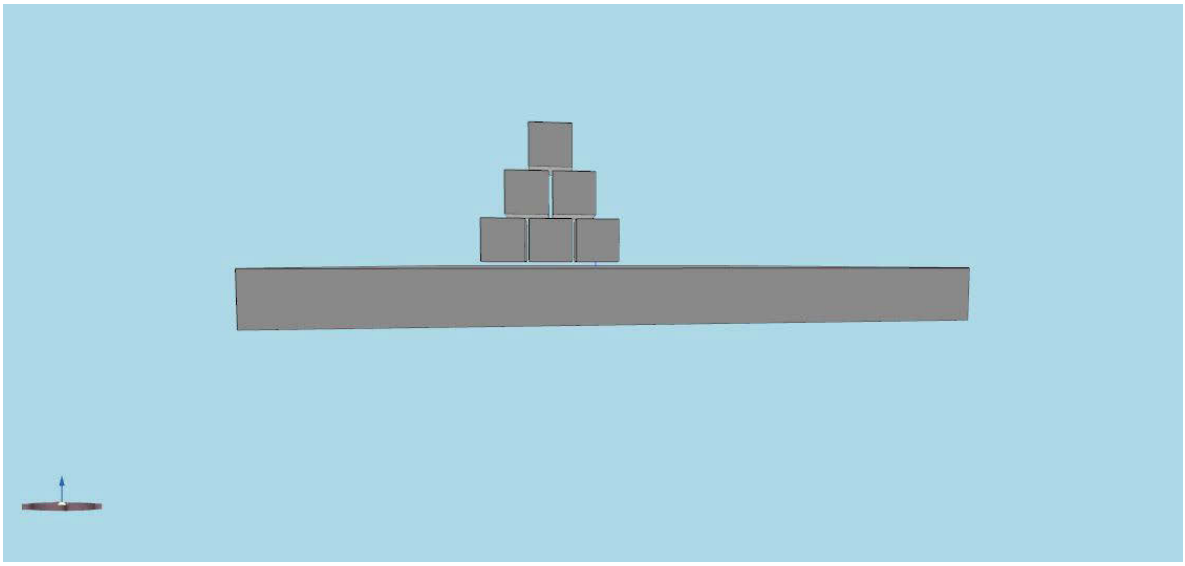


Figure 12: *The initial conditions for the N-Body collision test.*

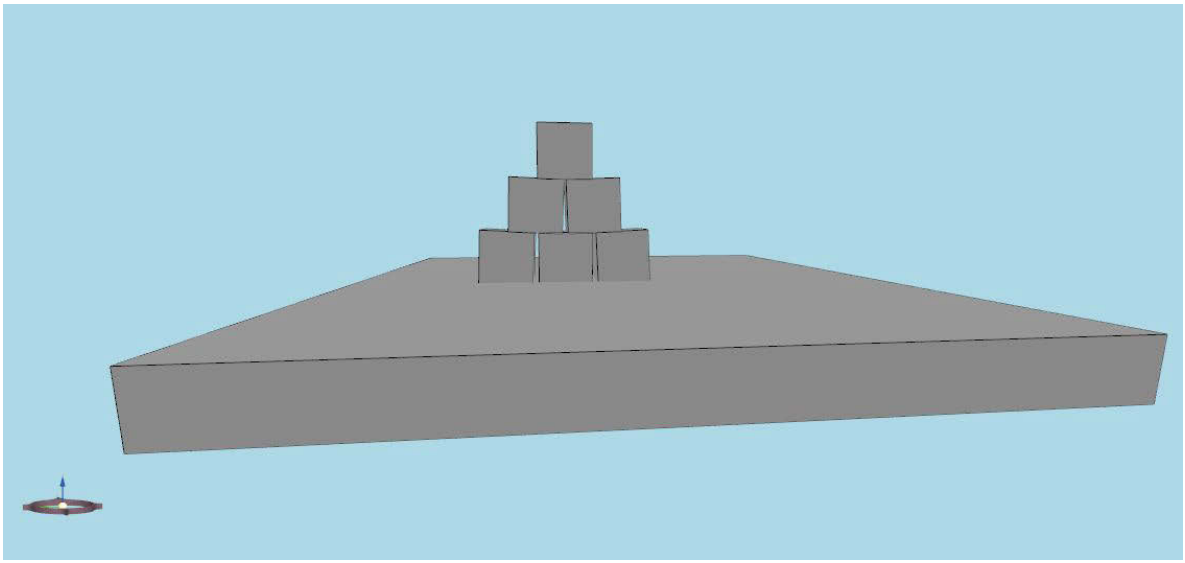


Figure 13: *The final resting state for the N-Body collision test.*

10.2 Cable collision detection with convex polyhedra

10.2.1 Setup

The purpose of this simulation is to demonstrate the new cable and convex-polyhedra collision detection capabilities of the method described in Section 3.2. This simulation consists of two `Payload`s and one `Cable`. One of the `Payload`s is kinematically fixed in space and is given a large flat cuboid `CollObjConvex` to act as a ground plane. Placed on top of the ground plane is another `Payload` with a 1 cubic meter shaped `CollObjConvex`. Finally a long `Cable` with a `CollObjCubicSpline` is held above both boxes with one end fixed in space. When the simulation begins, the cable will fall onto both Payload objects and finally come to rest on both. A figure illustrating the initial setup of the simulation can be found in Figure 14.

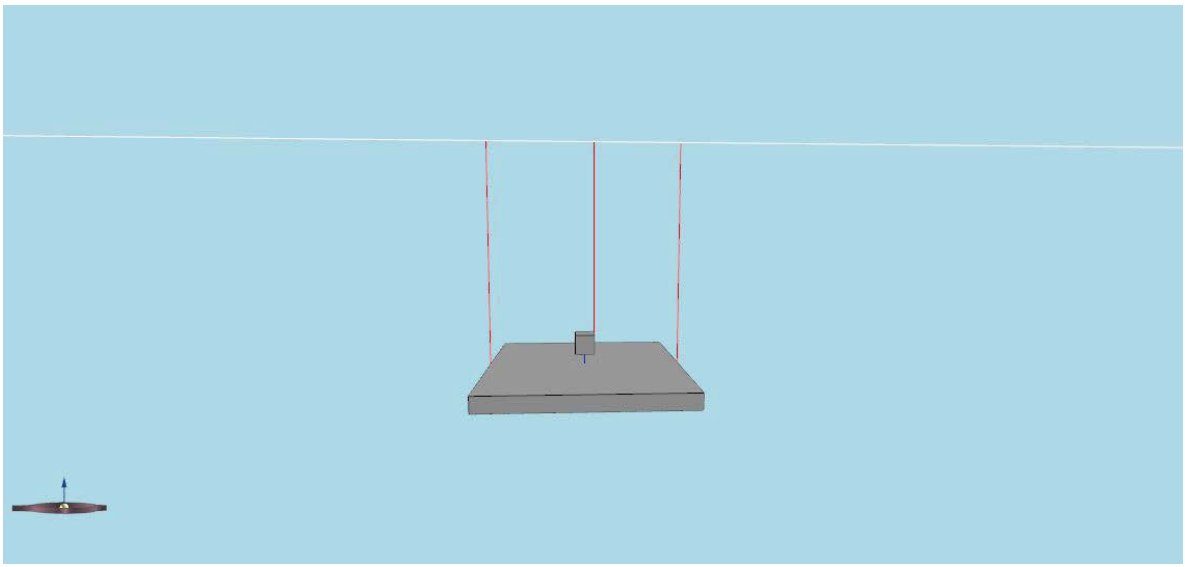


Figure 14: *The initial conditions for the cable collision detection test.*

10.2.2 Results

A picture of the simulation in its final state can be found in Figure 15. Note that the red line segments drawn between the cable and the boxes in Figures 14 and 15 represent the exact separation distance minima found by proximity query. There were at least 3 minima found during the course of the simulation and there were no missed collisions. Separation between the cable and other objects was successfully maintained.

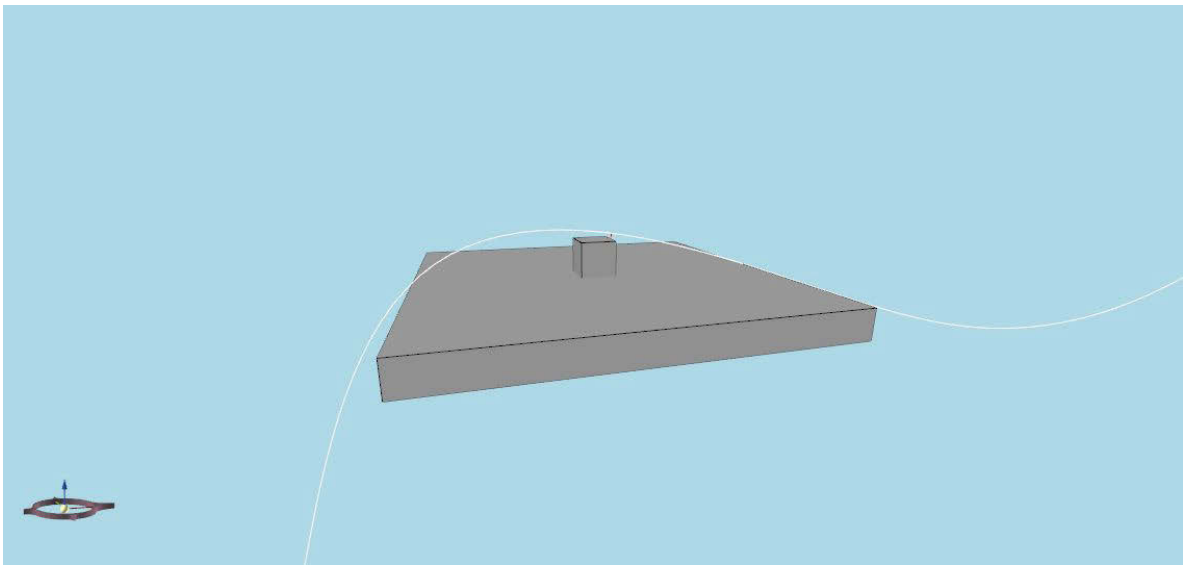


Figure 15: *The final static state for the cable collision detection test.*

10.3 Cable collision detection system overhead

10.3.1 Setup

The purpose of this test is to demonstrate the computational overhead of detecting collisions using the BVH based approach between a cable and a convex polyhedron. This demonstration consists of 3 permutations of the same case. These cases are illustrated in Figure 16. In the first permutation, a cable and a box shaped `Payload` object are arranged such that their BVs are not interfering. The cable and box are managed by a `CollisionHandler` in order to ensure they integrate in lock step. The BVH for this permutation consists of a single BV; thus, there is practically no collision detection overhead for their simulation.

The second permutation is identical to the first; however, the objects are managed by a `CollisionHandler`. The cable is discretized into 10 linear segments. Around each line segment an AABB BV is wrapped to form the leaf nodes for the BVH. This second permutation will demonstrate the overhead required to maintain the BVH for the cable at every iteration. Because the BVs of the two objects are not interfering, only one AABB BV collision query is performed which reports that there is no collision.

The third permutation is identical to the second except the cable is close enough to the box shaped `Payload` without actually colliding with it. The collision queries between the AABB BV of the box and the BVH of the cable return positives right down to most of the leaf-nodes. This triggers a set of minimum separation queries between the interfering leaf-node's line segments and the polyhedron. This permutation demonstrates the overhead associated with leaf-node distance queries.

The 3 permutations for this test are summarised below:

1. Permutation 1: Cable BVH consists of 1 BV, cable and box BV not interfering,
2. Permutation 2: Cable has 10 leaf node BVs, BVs not interfering,
3. Permutation 3: Cable has 10 leaf node BVs, BVs are interfering causing exact MSD queries.

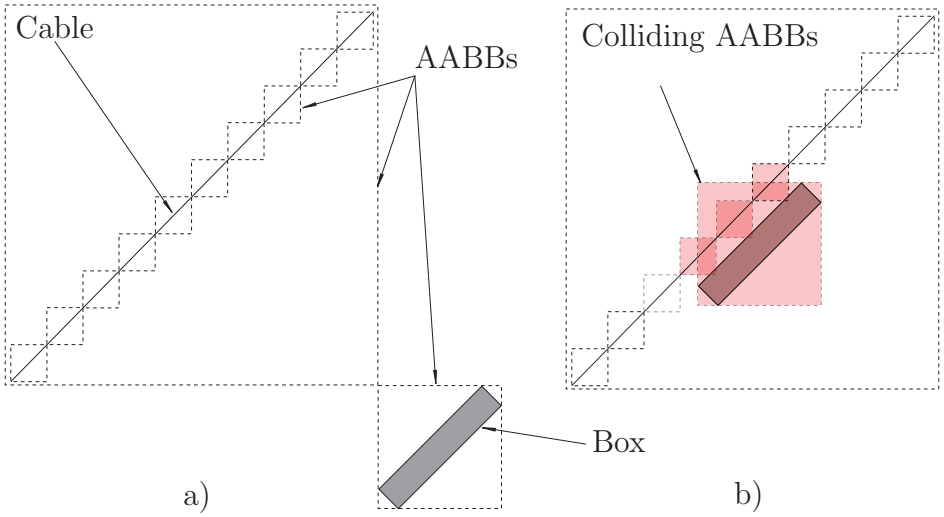


Figure 16: *The simulation setup for the first two permutations in a) and the third permutation in c). Note that only the first and last levels of bounding volumes for the cable's bounding volume hierarchy are shown.*

The simulations for all three cases are 1 second in length, their execution times are recorded and compared.

10.3.2 Results

Listed in Table 3 are the execution times for the 3 permutations described above. The benchmark permutation is the first permutation where the cable has a BVH consisting of 1 BV. The second permutation shows that the overhead for updating the cable's BVH is 1.3× larger than the cost of evaluating the steady state dynamics of the cable and box alone. The third permutation shows an extra cost 1.67× that of permutation 2. The extra cost comes from performing a series of AABB collision checks between the cable's BVH and the box' BV plus the more expensive cost of the distance queries between the line segment enveloped by the leaf node AABBs and the polyhedron.

Table 3: *The execution times for the 3 permutations of the cable collision detection system overhead test case.*

Permutation	Execution Time (s)
1	0.168
2	0.218
3	0.365

10.4 BVH vs. stochastic global optimization (SGO)

10.4.1 Setup

The purpose of this test is to demonstrate the differences in execution speeds between the BVH and the SGO cable collision detection methods. The three simulation permutations described in Section 10.3 have been conducted a second time using the SGO method. The SGO method has 3 random search pairs per iteration and remembers previous solutions. It is also using a golden section line search scheme to find the exact minimum separation in cases where the cable is deemed close enough to merit MSD solution refinement. Note that Permutation 1 was not run for this case and was assumed to be identical to that of Section 10.3 to make comparisons easy.

10.4.2 Results

Table 4 lists the execution times for the 3 permutations described above. The benchmark permutation is the first permutation where the cable and box are in the CollisionHandler for the purpose of lock step integration with minimal overhead. The second permutation shows that the overhead for finding the minima for 3 random pairs takes quite a bit more time than the cost of evaluating the steady state dynamics of the cable and box alone. The third permutation shows the negligible extra cost of performing a golden section search for the exact minima on the minimized pruned pairs from the combinatorial optimization stage. This shows that by far the most expensive operation is minimizing the random pairs, which requires crawling the polyhedral mesh and cable until the closest polygon is found.

There is reasonable potential for the use of SGO for resolving MSD problems of concave objects. The results shown here should not be extrapolated to other cases where BVH versus SGO design decisions need to be made. The SGO prototype implementation is likely a poor implementation as alternative, more efficient, subroutines are known to exist. The SGO method was implemented as a prototype first because it could be implemented quickly compared to a BVH based system. This enabled early testing and refinement of cable collision resolution.

It's possible that the expense of maintaining BVHs for more complicated concave geometries, including long winding cables that must be discretised into many sub-pieces,

could lead to larger overheads than a good SGO implementation. However, for this work, the BVH based method is currently outperforming the SGO method. This coupled with the fact that simulations of very long or finely discretised cables (leading to large BVHs) aren't anticipated, SGO based collision detection methods have not be pursued further for this project.

Table 4: *The execution times for the 3 permutations of the BVH vs. SGO test case.*

Permutation	Execution Time (s)
1	0.169
2	2.173
3	2.192

10.5 Cable-ship collision detection demonstration

10.5.1 Setup

The purpose of this test is to demonstrate the functioning of the new BVH based cable collision resolution system. This demonstration consists of two horizontal cable objects each with one end statically held in space, and the other end attached to its own spherical mass (modeled using a `Payload`). The cable and sphere begin the simulation over the side of the ship such that they will swing down to impact the side of the ship under gravity as shown in Figure 17. One of the cable and sphere pairs is assigned to a `CollisionHandler` while the other pair is not. This will demonstrate the behaviour of the system with and without collision resolution.

10.5.2 Results

A video has been submitted as an appendix to this report which shows the animated results of this simulation. Figure 18 shows the two cables and sphere pairs with the pair on the left resolving collisions with the ship while the pair on the right is passing through the ship's geometry. The red lines represent the computed MSD for a single line-segment, encapsulated by one of the colliding leaf node AABBs of the cable's BVH.

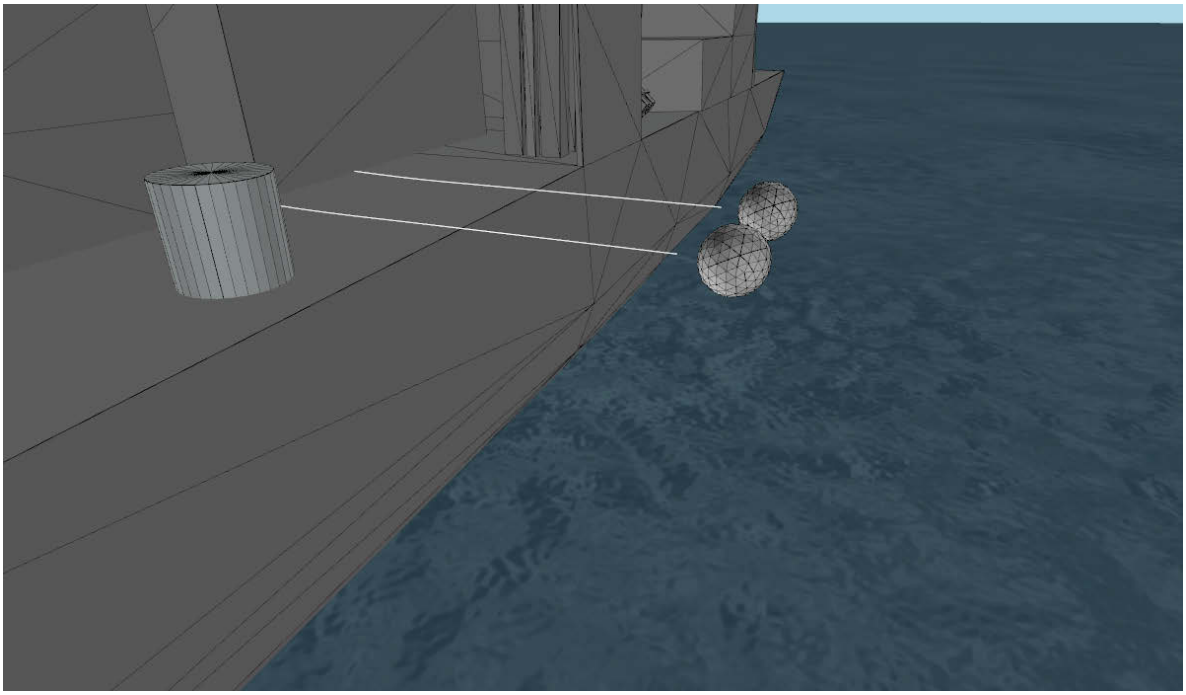


Figure 17: *The initial conditions for the cable-ship collision detection demonstration test.*

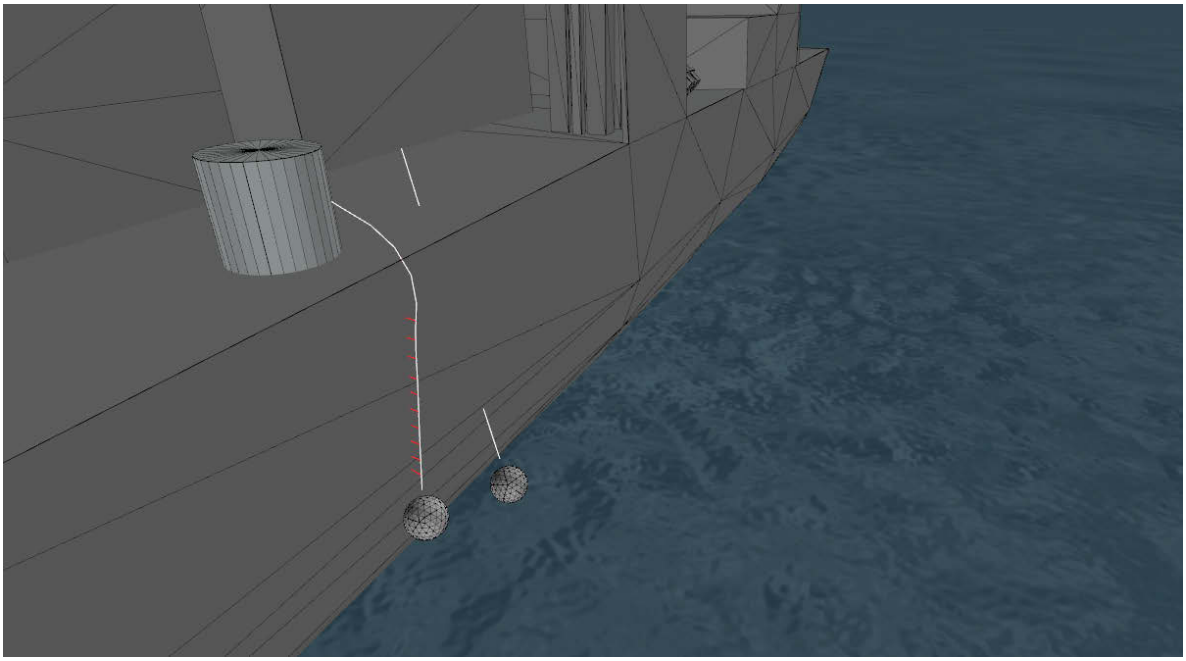


Figure 18: *Cables with and without cable-ship collision detection during demonstration test.*

10.6 Launch & Recovery simulation

10.6.1 Setup

The Launch and Recovery (L&R) simulation consists of a naval frigate in a seaway. The frigate has a forward speed of 1.0 m/s and is excited by a regular wave with period of 10.0 s and height of 1.0 m. On the deck of the naval frigate, as shown in Figure 19, is a rescue boat sitting in its cradle, a Palfinger-like boom crane with a winch (not visualized) and cable for launching and recovering the rescue boat. Also on the deck of the frigate are two deck hands (modeled as winches, thus not shown) holding on to a bow tag line and a stern tag line. The naval frigate's 6 DOF motions in time series form were computed using ShipMo3D. The frigate's motion time series data is used to kinematically drive the frigate in the SMS API simulation.

Figure 19: *The setup of the launch and recovery simulation, showing its actors. Winches are not shown.*

The simulation consists of the following `SimObject`s:

- `CollisionHandler`,
- `MarineVehicle` (naval frigate),
- `Cable` (main cable and 2 tag lines),
- `Boomcrane` (Palfinger-like crane),
- `Payload` (rescue boat).

The frigate is given a collision geometry to ensure that cables and the rescue boat do not pass through the deck or the hull of the ship. It is represented as a `CollObjConvex`

type of `CollisionObject`. The rescue boat's cradle, made up of 6 collision geometries, are also part of the naval frigate's collision geometry and each is represented by an individual `CollObjConvex` object. Together, they form a `CollObjConvexDecomp` type of `CollisionObject`. These collision geometries are shown in Figure 20. The rescue boat was also given a collision geometry, a convex hull of of the geometry shown in Figure 21. It is also represented as a `CollObjConvex`.

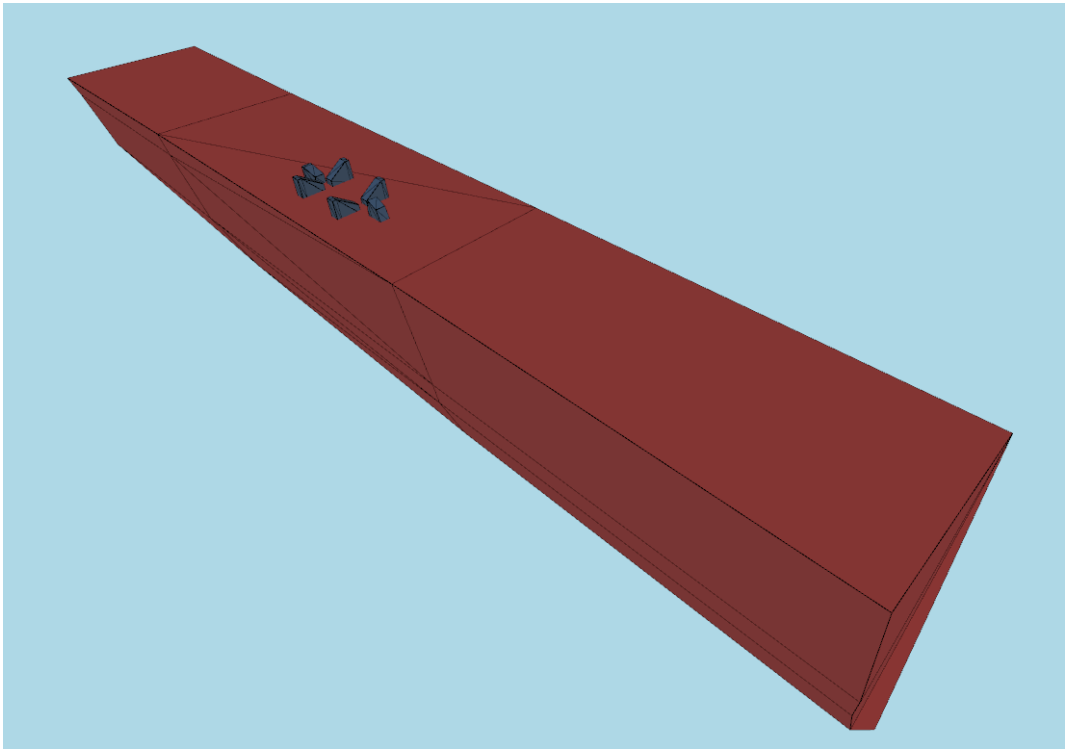


Figure 20: *The naval frigate's collision geometries; the collision geometries consist of 5 individual convex hulls of the rescue boat blocks and Frigate deck and hull shown here.*

The two tag lines are identical except for their locations. One of the end nodes of both lines are fixed to the frigate 1.0 m above its deck, while the other end node begins the simulation free to fall. The free ends of the tag lines will eventually be attached to the rescue boat to provide stability during the launching and recovering periods. Each tag line is assigned a `CollObjCubicSpline` type `CollisionObject` which is discretized into 18 line segments. Each line segment is enveloped by an AABB BV, which is one of the leaf nodes of the cable's BVH.

For this simulation, the rescue boat is allowed to collide with the ship, and the tag lines are allowed to collide with the ship, but the cables can't collide with the rescue boat or themselves.

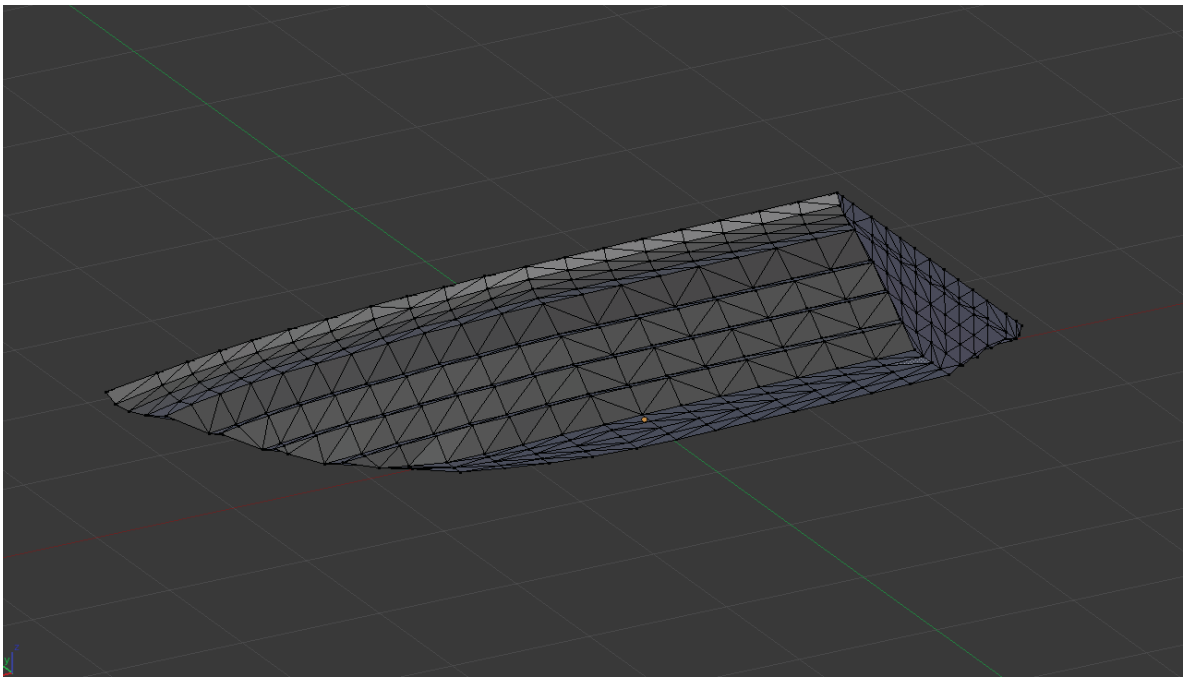


Figure 21: *The rescue boat's collision geometry, the true collision geometry will be a convex hull of the geometry shown here.*

This simulation script has been modified slightly since the simulation conducted in [3]. However, for the most part, the simulation script is identical. The reader is referred to that document for a description of the L&R `Director` script. The simulation is run for a length of 215 seconds.

10.6.2 Results

DSA has completed a L&R simulation with cable collisions being detected. A short video of a visualization of the results has been produced and submitted as an appendix to this report. Some results from the simulation are presented below in the form of screen shots and some plots of the tensions in the cables over the course of the simulation.

10.6.2.1 Screen shots of the simulation

Figure 22 to 26 show the progress of the L&R simulation at various points throughout the simulation. Figure 22 shows the state of the simulation at the beginning of the simulation. Figure 23 shows the state of the simulation after the 3 cables have been attached to the rescue boat and the rescue boat has been lifted out of its cradle. Figure 24 shows the state of the simulation after the boomcrane has moved the rescue boat overboard and lowered it into the water but before the cables are detached. Figure 25 shows the

rescue boat being lowered back into its cradle. Finally, Figure 26 shows the state of the `SimObjects` at the end of the simulation, with the rescueboat back in its cradle with the Palfinger-like boomcrane folded up.

Figure 22: *The L&R simulation at $t = 0$ s showing all of the actors in their initial states.*

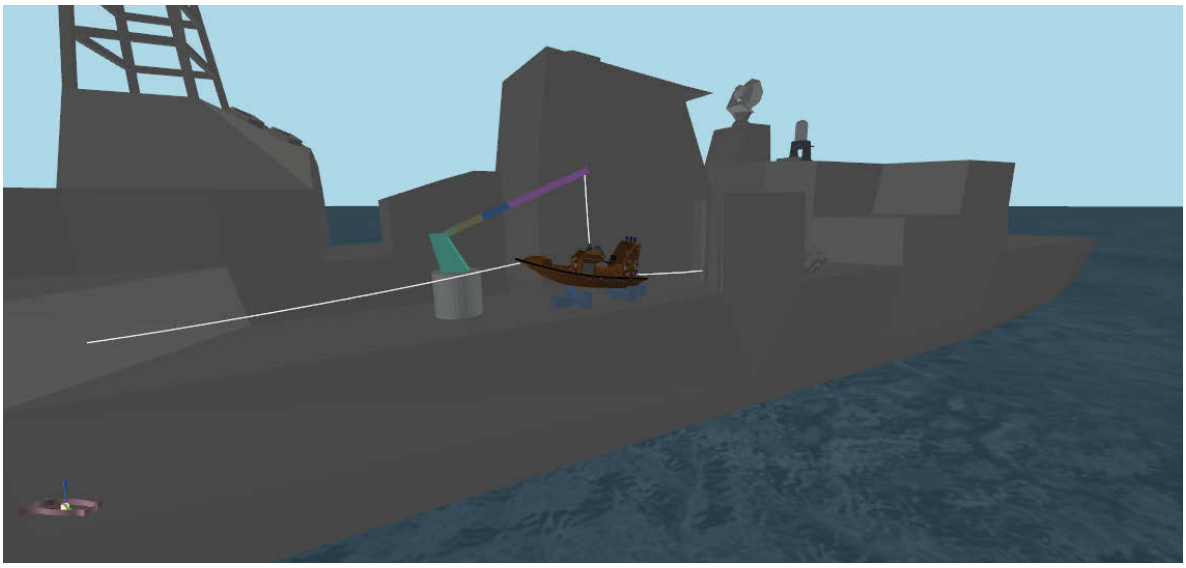


Figure 23: *The L&R simulation at $t = 48$ s showing all cable attached to the rescue boat, which has just been lifted out of its cradle.*

Figure 24: *The L&R simulation at $t = 72$ s showing the rescue boat about to be released in the seaway.*

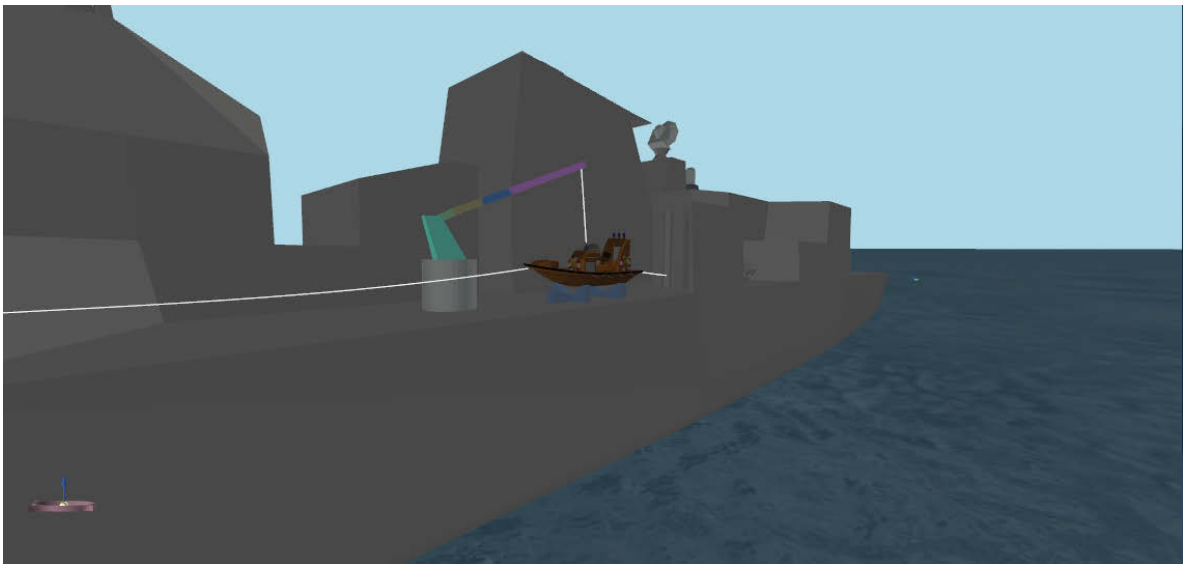


Figure 25: *The L&R simulation at $t = 156$ s showing the rescue boat being lowered back in its cradle.*

Figure 26: *The L&R simulation at $t = 212$ s showing the rescue boat back in its cradle and the Palfinger-like boom crane folded up.*

10.6.2.2 Cable tensions

The tensions in all three cables over the course of the simulation can be seen in Figures 27 through 29. The tensions in the boom crane cable, bow line and stern line are shown in Figures 27, 28 and 29, respectively.

The rescue boat weight is approximately 5000 N, which is apparent in Figure 27 when the rescue boat is first lifted out of its cradle at $t \approx 48$ s. The boom crane cable has a mass on its end weighing ≈ 40 N on its end node. It represents the mass of a hook that might be attached to it. The tension caused by the weight of this hook can be seen at the beginning of the results and at the end of the results when the rescue boat is not attached.

The winches model deck hands holding the tag lines. They are designed to maintain a tension of 400 N. When the cables are attached near $t = 48$ s, it is apparent that they are attempting to maintain their tension in Figures 28 and 29 until the cables are disconnected again around $t = 72$ s.

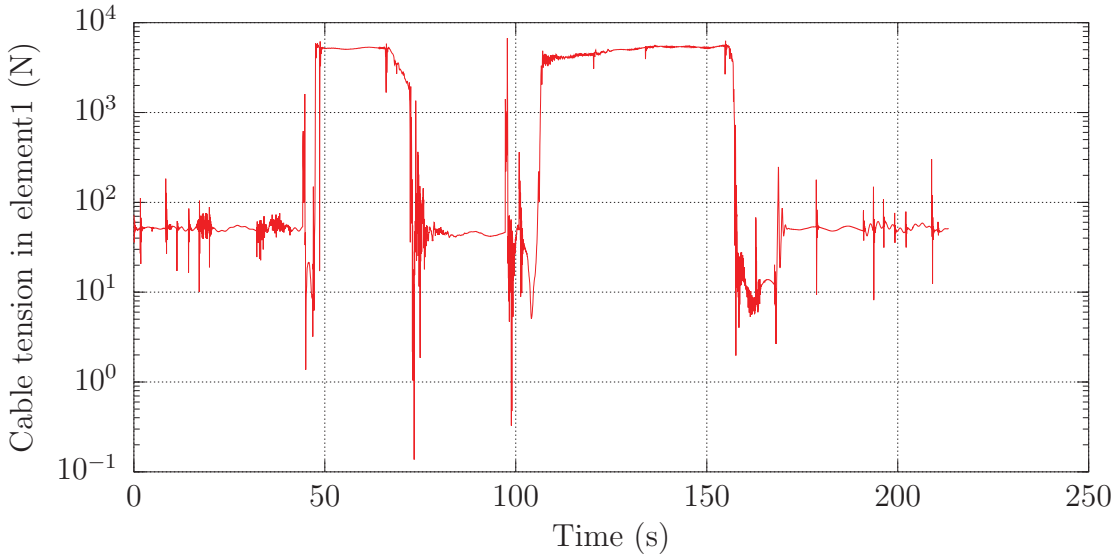


Figure 27: *The tension in the boom crane cable over the course of the simulation.*

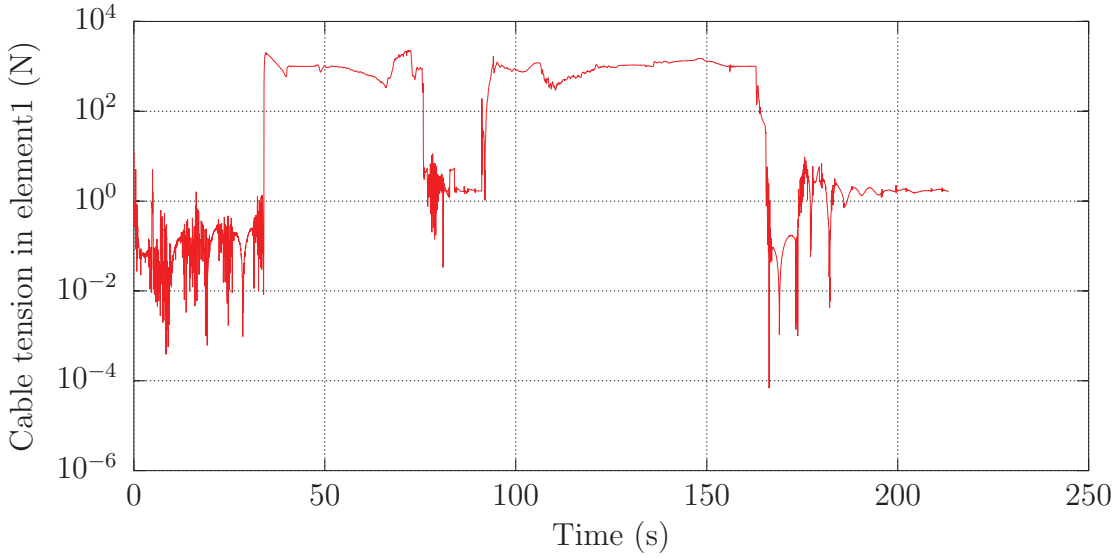


Figure 28: *The tension in the bow tag line over the course of the simulation.*

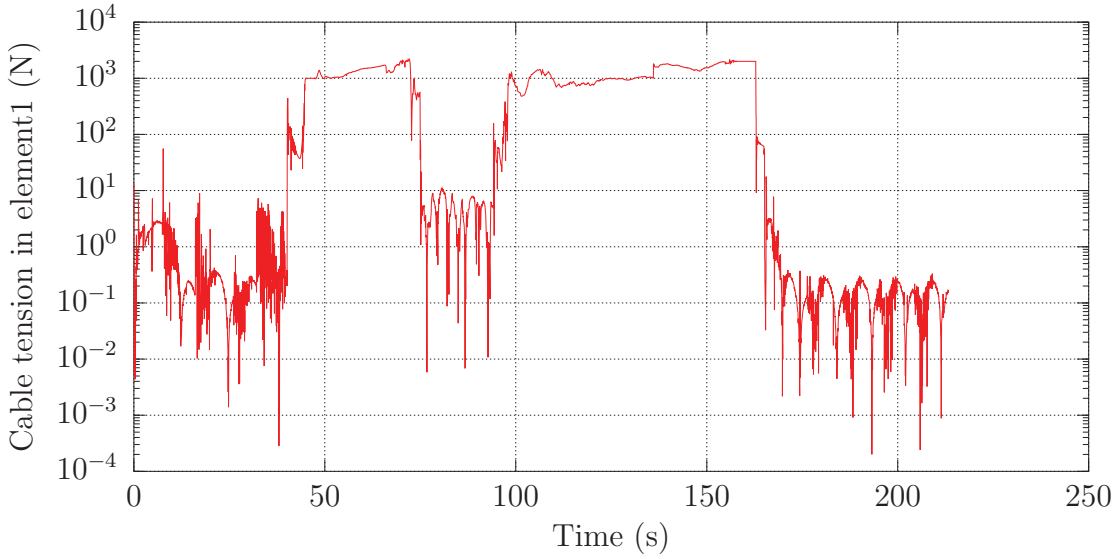


Figure 29: *The tension in the stern tag line over the course of the simulation.*

11 Future Work

This section discusses areas of improvement for the SMS API that could be addressed in the future. In general, future work should focus on simulating observed launch and recovery operations, or validation of the entire simulation. Improvements to the code may be necessary based on the requirements of such work. There are a few areas in the simulation models where recommended improvements have been identified.

The winch model has both a constant tension mode and a constant velocity mode. It is a kinematic model as the inertia of a winch's drum is not taken into account. A dynamic model of the winch would improve the fidelity of the winch used by boomcranes. For modelling a deck hand, the current winch model qualitatively works well for controlling the rescue boat. However it does not model a deck hand with a high degree of fidelity. A deck hand has mass and can be displaced by a large tension. They can only apply a certain level of tension before they are forced to let go. These aspects are not reasonably being modelled.

The cable's contact dynamics model currently does not include a friction model. Some time should be spent adding one, otherwise a cable laying on the deck of a ship is likely to slide around as there is no resistance to tangential motion.

A real naval frigate displaces the fluid while moving through it, disturbing the fluid particles near its hull. When launching a rescue boat with forward speed, the rescue boat is likely to be significantly affected by the disturbed fluid particle field. This effect is currently not modelled.

Tighter integration between ShipMo3D and the SMS API will be required if forces or articulated masses need to be transferred between the SMS API models and the ShipMo3D frigate. A federated simulation would likely be beneficial for accomplishing this.

When the collision detection system is in use, every object managed by the `CollisionHandler` is forced to integrate at the same timestep. `SimOject`s that would normally integrate with larger time steps must integrate at the same timestep as the `SimObject` with the smallest time step requirements. This results in many more unecessary dynamics evaluations for those objects. To speed up execution speeds of SMS API, the decoupling of the object integration rates should be investigated.

Currently, only regular waves can be simulated by the SMS API. To model more realistic wave loading cases, the ability to simulate a random seaway should be implemented.

It would be useful to update the software manual to reflect some of the recent additions and modifications made to the SMS API.

12 Conclusions

A new collision detection infrastructure has been created to handle collisions between any number of `SimObject`s. Collisions between `Cable`s and `RigidBody` based `SimObject`s are now being resolved using a BVH based approach. Both a BVH and an SGO cable collision detection method were implemented and a comparison was made between the two. The BVH based approach was shown to be most efficient thus it was chosen to be the cable collision detection method. Because the SGO based method implementation was less than optimal it is unclear whether or not an SGO method could outperform a BVH based approach. More research is required to make such a determination.

A Hunt-Crossley contact dynamics model was used to determine the normal contact forces acting between a cable and the ship's deck. No friction model is currently used to handle tangential contact forces.

The launch and recovery simulation from previous projects has been updated to include the new developments. The tag lines now no longer interfere with the frigate's geometry, producing higher fidelity simulation results.

References

[1] Lotan, I., Schwarzer, F., Latombe, J. C., and Halperin, D. (2002). Efficient Maintenance and Self-Collision Testing for Kinematic Chains. In *Symposium on Computational Geometry*, pp. 43–52. Barcelona, Spain.

[2] Carretero, J. A. and Nahon, M. A. (2005). Solving Minimum Distance Problems with Convex or Concave Bodies Using Combinatorial Global Optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, **35**(6), 1144–1155.

[3] Roy, A., Steinke, D., and Nicoll, R. (2012). Improved Simulation of Ship Mechanical Systems. (DRDC Atlantic CR 2012-093). Defence Research and Development Canada – Atlantic.

[4] Hunt, K. H. and Crossley, F. R. E. (1975). Coefficient of Restitution Interpreted as Damping in Vibroimpact. *Journal of Applied Mechanics*, **42**, 440–445.

[5] Ericson, C. (2005). Real-Time Collision Detection, The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann Publishers.

[6] Lin, M. and Manocha, D. (2003). Collision and proximity queries. *In Handbook of Discrete and Computational Geometry, Chapman and Hall/CRC*.

[7] Gilbert, E. G., Johnson, D. W., and Keerthi, S. S. (1988). A Fast Procedure for Computing the Distance Between Complex Objects in three Dimentional Space. *IEEE Journal of Robotics and Automation*, **4**(2), 193–203.

[8] Mirtich, B. (1998). V-Clip: fast and robust polyhedral collision detection.. *ACM Transactions on Graphics*, **17**(3), 177–208.

[9] Nahon, Meyer (1994). Determination of the Interference Distance Between Two Objects Using Optimization Techniques. In *Proceedings of ASME Conference on Advances in Design Automation*, Vol. 1, pp. 1–6. Minneapolis, Minnesota, USA.

[10] Uppuluri, R. and Carretero, J. A. (2004). A Novel Optimization-Based Pruning Strategy for Concave Minimum Distance Problems. In *Proceedings of the 2004 CSME Forum, London, Ontario*, pp. 581–591. London, Ontario, Canada.

[11] Carretero, J. A., Nahon, M. A., and Ma, O. (2002). Using Genetic Algorithms with Niche Formation to Solve the Minimum Distanct Problem Amongst Concave Objects. In *Proceedings of the 2002 ASME Design Engineering Technical Conferences, 28th Design Automation Conference*.

[12] Thomas, Frederico and Torras, Carme (1994). Interference Detection between Non-Convex Polyhedra Revisited with a Practical Aim. In *Proceedings of the 1994 conference on Robotics and Automation*.

[13] Jiménez, P., Thomas, F., and Torras, C. (2001). 3D Collision Detection: A Survey. *Computers and Graphics*, **25**(2), 269–285.

[14] Yoon, S.-E. and Manocha, D. (2006). Cache-Efficient Layouts of Bounding Volume Hierarchies. *EuroGraphics*, **25**(3).

[15] Lin, M. C. and Manocha, D. (1995). Fast Interference Detection Between Geometric Models. *The Visual Computer*, **11**, 542–461.

[16] Johnson, David E. and Cohen, Elaine (1998). A framework for efficient minimum distance computations. In *Proceedings IEEE Intl. Conf. Robotics & Automation*, pp. 3678–3684. Leuven, Belgium.

[17] Uppuluri, R. L. (2005). Distance Determination Algorithms With Novel Pruning Strategies for Complex Dynamics Simulations. MScE thesis. Department of Mechanical Engineering, University of New Brunswick, Fredericton, New Brunswick, Canada. Fredericton, New Brunswick.

[18] Raghupathi, L., Grisoni, L., Faure, L., Marchal, D., Cani, M.-P., and Chaillou, C. (2004). An intestine surgery simulator: Real-time collision processing and visualization. *IEEE Transactions on Visualization and Computer Graphics*, **10**(6).

[19] Carretero, J. A. and Buckham, B. J. (2004). Simulation of Submerged Slack Tethers and their Interaction with the Environment. In *Proceedings of the 23rd ASME International Conference on Offshore Mechanics and Arctic Engineering (OMAE)*, Vancouver, British Columbia, Canada.

[20] Roy, A. R., Carretero, J. A., and Buckham, B. J. (2007). Detecting Tether Self-Collisions in Tethered ROV Simulations. In *The proceedings of the 26th International Conference on Offshore Mechanics and Arctic Engineering (OMAE)*, San Diego.

[21] Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnenat-Thalmann, N., Strasser, W., and Volino, P. (2005). Collision Detection for Deformable Objects. *Computer Graphics Forum*, **24**(1), 61–81.

[22] Brown, J., Latombe, J.-C., and Montgomery, K. (2004). Real-time Knot-Tying Simulation. *The Visual Computer: International Journal of Computer Graphics*, **20**(2), 165–179.

[23] Gottschalk, S., Lin, M. C., and Manocha, D. (1996). OBBTree: A Hierarchal Structure for Rapid Interference Detection. In *SIGGRAPH '96 proceedings of the 23rd annual conference on Computer graphics and interactive techniques*.

[24] Redon, S., Kim, Y. J., Lin, M. C., and Manocha, D. (2003). Fast Continuous Collision Detection for Articulated Models (TR03-038). Technical Report. University of North Carolina at Chapel Hill.

[25] Hippmann, G. (2003). An Algorithm For Compliant Contact Between Complexly Shaped Surfaces in Multibody Dynamics. In Ambrósio, J. A. C., (Ed.), *IDMEC/IST*.

[26] Drozdek, Adam (1996). Data Structures and Algorithms in C++, PWS Publishing Company.

[27] Ma, O. and Nahon, M. (1992). A General Method for Computing the Distance Between two Moving Objects Using Optimization Techniques. In *ASME Advances in Design Automation*, Vol. 1, pp. 109–117. New York: ASME.

[28] Ma, O. (1995). Contact Dynamics Modelling for the simulation of the Space Station Manipulators Handling Payloads. In *IEEE International Conference on Robotics and Automation*, pp. 1252–1258.

[29] Roy, Andre, Carretero, Juan A., Buckham, Bradley J., and Nicoll, Ryan S. (2009). Continuous Collision Detection of Cubic-Spline-Based Tethers in ROV Simulations. *J. Offshore Mech. Arct. Eng.*, **131**(4), 041102–10.

[30] Gilardi, G. and Sharf, I. (2002). Literature Survey of Contact Dynamics Modelling. *Mechanism and Machine Theory*, **37**(10), 1213–1239.

[31] Roy, A. R. and Carretero, J. A. (2012). A damping term based on material properties for the volume-based contact dynamics model. *International Journal of Non-Linear Mechanics*, **47**(3), 103–112.

[32] Gonthier, Y., McPhee, J., Lange, C., and Piedboeuf, J.-C. (2004). A Regularized Contact Model with Asymmetric Damping and Dwell-Time Dependent Friction. *Multibody System Dynamics*, **11**, 209–233.

[33] Flores, Paulo, Machado, Margarida, Silva, Miguel T., and Martins, Jorge M. (2011). On the continuous contact force models for soft materials in multibody dynamics. *Multibody System Dynamics*, **25**, 357–375.

[34] (2013). Ergonomics - Pushing and Pulling (Online). Canadian Center for Occupational Health and Safety. `http://www.ccohs.ca/oshanswers/ergonomics/push1.html` (22 March 2013).

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Protected.)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Dynamic Systems Analysis Limited
101-19 Dallas Road, Victoria, BC, Canada, V8V 5A6

2a. SECURITY MARKING (Overall security marking of the document, including supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC DECEMBER 2012

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Simulation of Launch and Recovery of Small Craft Including Cable Collisions and Cable Tension from Deck Personnel

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

Roy, A.; Steinke, D.; Nicoll, R.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
August 2013	66	34

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Atlantic
PO Box 1012, Dartmouth NS B2Y 3Z7, Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
11ge02	W7707-135587/001/HAL

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Atlantic CR 2013-136	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

This report describes the simulation of launch and recovery of a small boat from a ship using a boom crane. The simulation was developed using the existing Ship Mechanical Systems Application Programmer Interface (SMS API). The SMS API was extended to include modelling of collisions between cables and solid objects, which can include a ship or a small boat. The influence of tag lines being handled by deck personnel was modelled, with a winch model being used to approximate the cable tension being applied by a person on deck. Simulation components were verified by comparison with a number of test cases. A complete launch and recovery scenario was simulated and visualized.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

collision dynamics; cranes; launch and recovery; ship motions; simulation; time domain; waves

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE DÉFENSE

www.drdc-rddc.gc.ca