This document contains the draft version of the following paper:

A. Thakur and S.K. Gupta. Improving performance of rigid body dynamics simulation by removing inaccessible regions from geometric models, Computer-Aided Design, Volume 44, Issue 12, December 2012, Pages 1190-1204, ISSN 0010-4485, 10.1016/j.cad.2012.06.007.

Readers are encouraged to get the official version from the journal's web site or by contacting Dr. S.K. Gupta (skgupta@umd.edu).

Improving Performance of Rigid Body Dynamics Simulation by Removing Inaccessible Regions from Geometric Models

Atul Thakur[a], Satyandra K. Gupta[b]

[a] *Department of Mechanical Engineering, University of Maryland, College Park, MD 20742, USA*
E-mail: athakur@umd.edu
[b] *Department of Mechanical Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA*
E-mail: skgupta@umd.edu

Abstract

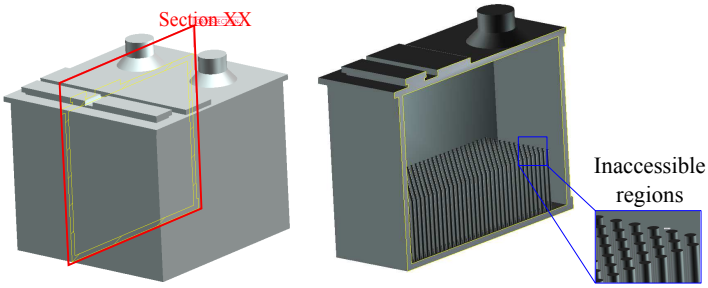
Rigid body simulations require collision detection for determining contact points between simulated bodies. Collision detection performance can become dramatically slow, if geometric models of rigid bodies have intricate inaccessible regions close to their boundaries, particularly when bodies are in close proximity. As a result, frame rates of rigid body simulations reduce significantly in the states in which bodies come in close proximity. Thus, removing inaccessible regions from models can significantly improve rigid body simulation performance without influencing the simulation accuracy because inaccessible regions do not come in contact during collisions. This paper presents an automated pair-wise contact preserving model simplification approach based upon detection and removing of inaccessible regions of a given model with respect to another colliding model. We introduce a pose independent data-structure called part section signature to perform accessibility queries on 3D models based on a conservative approximation scheme. The developed approximation scheme is conservative and do not oversimplify but may undersimplify models, which ensures that the contact points determined using simplified and unsimplified models are exactly identical. Also, we present a greedy algorithm to reduce the number of simplified models that are needed to be stored for satisfying memory constraints in case of a simulation scene with more than two models. This paper presents test results of the developed simplification algorithm on a a variety of part models. We also report results of collision detection performance tests in rigid body simulations using simplified models, which are generated using developed algorithms, and their comparison with the identical performance tests on respective unsimplified models.

Keywords: Model simplification, collision detection, rigid body simulation, contact preserving, physics-aware model simplification, physics-aware model simplification

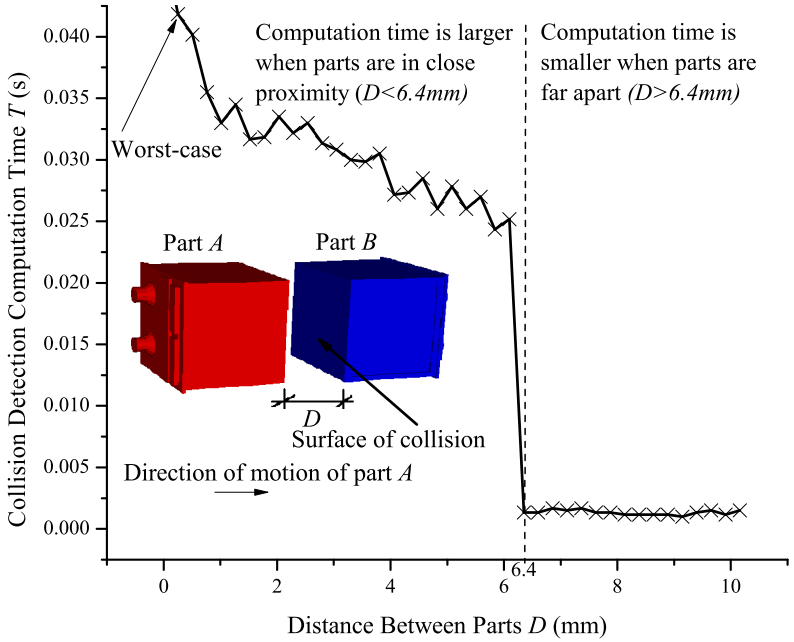
1. Introduction

Rigid body dynamics simulations are nowadays used in a wide variety of interactive virtual environment based applications such as computer games, unmanned vehicle simulations, surgical simulations, assembly process simulations, etc. Many applications also need simulation of compliant parts in addition to rigid body simulations, *e.g.*, in health-care and manufacturing. Such compliance is often modeled using rigid bodies connected by springs (*i.e.*, pseudo-rigid bodies [1]). Pseudo-rigid bodies are also often simulated using rigid body simulation. Thus, rigid body dynamics simulation can be applied to a wide variety of applications involving both rigid as well as compliant parts.

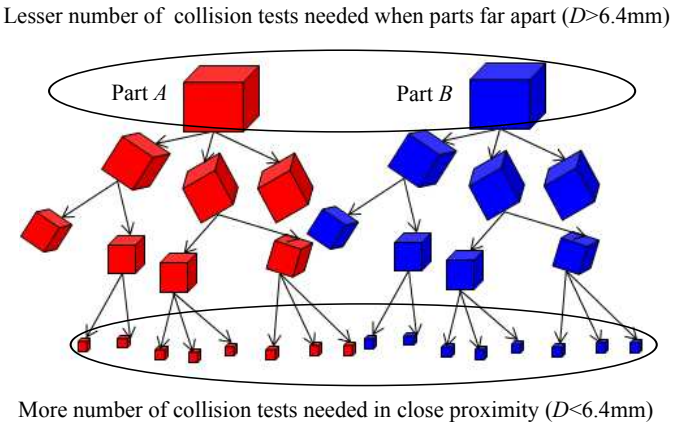
In a rigid body simulator, the majority of computational effort is spent on computing contact points among colliding rigid bodies at each simulation time step. Contact points play an important role in the computation of reaction forces. The reaction forces are integrated to update velocities



(a) 3D CAD model. (b) Cut-through view.

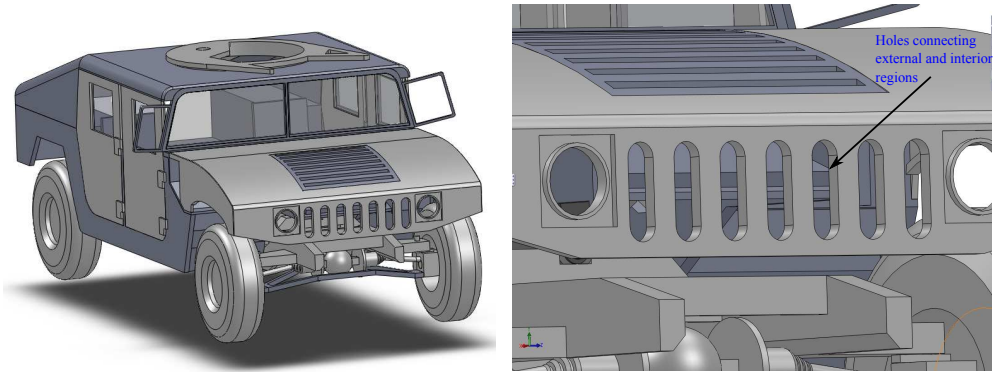


(c) Variation of collision detection time with distance between models.



(d) Lesser number of of collision tests are required when models are far apart and more in close proximity.

Figure 1: Collision computation time drastically increases when models are in close proximity. Collision detection search in hierarchical tree is deep due to inaccessible regions close to the surface of collision. Hierarchical decomposition does not prune the inaccessible facets of a model, which cannot come in contact with other models (RAPID [2] collision detection engine was used for this test).



(a) Vehicle assembly model.

(b) Close up view of vehicle model to show holes connecting external and internal shells prohibiting use of flood filling algorithm to eliminate interior inaccessible features.

Figure 2: In a vehicle model interior inaccessible regions may not be eliminated simply by using flood filling algorithm to separate regions of model exposed to collision with inaccessible regions because of holes connecting external and internal shells.

and positions of rigid bodies. Often, collision detection algorithms are used for computing contact points. Collision detection performed using detailed geometric models may slow down rigid body dynamics simulation as mechanical parts have a large number of detailed features. In general, assemblies are exported as triangle tessellated models in rigid body simulators, which can have large size and can slow down collision computation time. In order to reduce the computation time for collision detection, model geometry needs to be simplified. Many model simplification techniques have been reported in the domain of graphics rendering [24].

A simplification approach may make two types of errors: (1) a region is removed from the model that may contain contact points; (2) a region that cannot contain a contact point is not removed from the model. We are interested in a conservative approach. This means that the approach should not make the first type of errors. On the on other hand, we can tolerate the second type of errors. Model simplification techniques developed for graphics rendering might make the first type of errors and hence cannot be used directly in rigid body simulation applications.

Hierarchical bounding volume based approach is highly successful class of model simplification techniques, which is used for accelerating collision queries to determine contact points accurately. These techniques are excellent for most simulation states in which models are relatively far apart. The collision computation time, however, increases dramatically for the models with inaccessible regions in a close proximity. Due to this increase in the collision computation time, the simulation cannot be guaranteed to run at interactive rates for all states. In order to understand and demonstrate this effect, we performed a collision detection test (see Figure 1) using RAPID engine [2], which is based on oriented bounding box (OBB) decomposition scheme. A point to be noted about the geometry of models A and B as shown in Figure 1(a) and 1(b), is that there are inaccessible regions just behind the collision surface. Figure 1(c) shows the variation of collision computation time T versus the distance D between A and B. In the figure, model A is gradually brought close to another model B with the same geometry and for each relative separation distance D, collision computation time T is determined and plotted. The collision detection time is about $0.001s$ when A and B are far apart ($D \geq 6.4mm$). However, in a close proximity ($D < 6.4mm$) the collision detection time increases dramatically. Figure 1(c) also shows that the worst-case collision computation time for the detailed geometry is about 0.041 s.

The reason behind this dramatic increase in the collision computation time is shown in Figure 1(d). At farther distances, the number of collision tests is smaller, whereas in a close proximity

the number of collision tests increases. During the computations in the lowest level of the hierarchy, a large number of interior inaccessible facets, which are spatially very near to the point of collision, need to be tested. This causes considerable increase in the collision computation time when models are close. In many models similar to the one shown in Figure 2(a) (such as vehicles, boats, etc.), there are many details, which are inaccessible but spatially very close to the point of collision (see Figure 2(b)). If inaccessible facets of a model are removed off-line (*i.e.*, before the simulation) then this can reduce the computation time significantly without affecting the accuracy of the obtained contact points. We would like to note here that although the models described here are assemblies, we have exported them as tessellated models (in the form of STL files). This is required because, in the existing rigid body simulators and collision detection engines, tessellated geometries (polygon soup data as opposed to valid solid models) are used for contact point/force determination.

In some part models, the geometry is split into multiple shells. In such cases a simple flood filling algorithm on the exterior shell can be used for separating interior and exterior shells and then interior shells can be removed to generate simplified models. However, in many cases inaccessible regions are located on the same shell (shown in Figure 2). In all the examples and case studies presented in this paper, inaccessible regions lie on the same shell as the exterior regions. Hence, a simple flood filing algorithm does not work on such models and a more sophisticated model simplification approach for removing inaccessible regions is required for improving frame rates of rigid body simulations.

The set of inaccessible facets depend upon the collision context (*i.e.*, pair of models in collision). In many problems, the collision context is known in advance or can be easily determined as the models in collision are known beforehand. This opens up a possibility of storing and retrieving multiple simplified representations of models based on the corresponding collision contexts. This scheme is promising as the memory is relatively inexpensive compared to the real-time computation of contact points for fully detailed models. We plan to utilize the collision context to generate contact-preserving simplified models. This paper presents an accessibility based technique to simplify models with respect to each other in an off-line manner, *i.e.*, before the simulation is performed, in such a way that the possible contact points are preserved. In order to perform a conservative pair-wise model simplification based on accessibility, we report a pose independent data structure called part section signature. We also report a greedy approach to reduce the number of pair-wise simplified models to satisfy memory constraints for a given rigid body simulation scene with several interacting geometries.

In Section 2, we discuss some related work in the area of model simplification. Section 3, presents the problem statement and an overview of model simplification approach to solve the problem. Section 4 to 6 discusses detailed description of the algorithms for model simplification. In Section 7 we discuss the implementation details and results and finally conclusions are presented in Section 8.

2. Related work

Model simplification has been studied for various application domains such as interactive visual rendering, network transmission of models, finite element analysis, and collision detection. In a previous paper, we classified the available model simplification techniques for physics based simulation applications based on the simplification operators into four types namely surface entity based, volumetric entity based, explicit feature based and dimension reduction [3]. These categories are summarized below.

- *Surface entity based techniques :* In this approach, models are simplified by operating on surface entities such as faces, edges and vertices. Under this category, reported techniques are edge collapse technique [4], face clustering [5, 6] and low pass filtering approach such as Fourier Transforms [7].

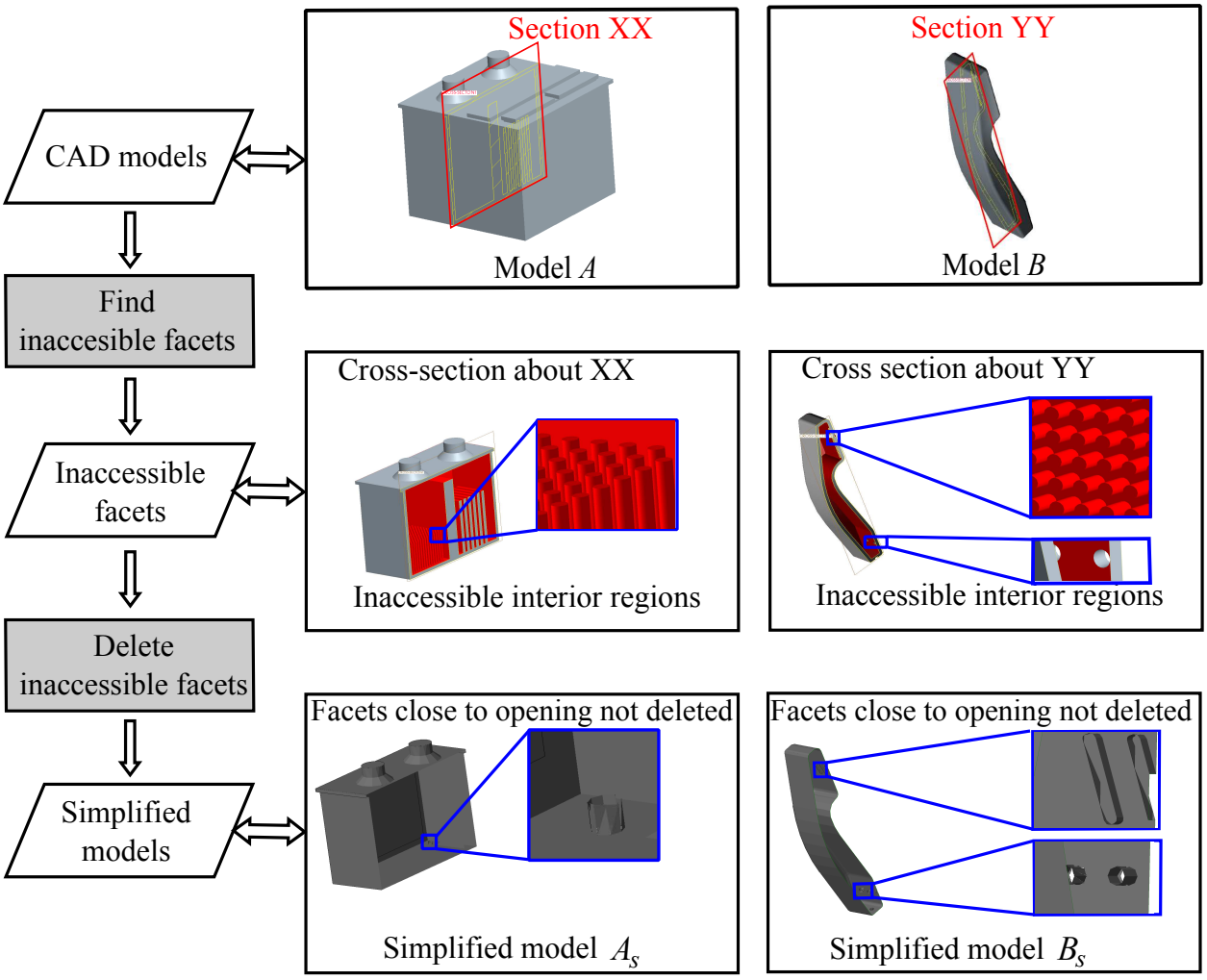


Figure 3: Contact preserving model simplification determines and removes inaccessible facets of a model with respect to another model. Removal of inaccessible facets simplifies models while preserving potential contact points during collisions.

- *Volumetric entity based techniques :* In this approach, models are simplified by operating on volumetric entities from the model. Major techniques reported under this category are effective volume [8, 9, 10], cellular representation [11] and voxel [12, 13] based approaches. Boeing's Voxmap PointShell^{TM} uses voxel based representation for collision/proximity detection, swept volume generation, force modeling, spatial query, occlusion culling, real-time shadows, and volume intersection applications [14].

- *Explicit feature based techniques :* Many reported model simplification techniques are based on recognizing explicit application features prior to simplification. These techniques fall into following categories: prismatic feature simplification, blend simplification, and arbitrary shaped feature simplification based on the type of features covered. In the explicit feature representation based techniques, the feature information and semantics is explicitly extracted from the given CAD model or is present in the model in addition to the geometric and the topological data. The features are user defined and depend on the context of application (*e.g.*, in case of mechanical analysis, holes, slots, steps, pockets, fillets, rounds, etc. are sought features). Feature based modeling and feature recognition techniques are described in details in [15].

- *Dimension reduction based techniques :* In computer-aided engineering analysis applications such as FEA, CFD, etc., dimension reduction techniques such as medial-axis, medial-surface, mid-surface-abstraction, etc. are often used for model simplification [16, 18, 19, 20, 21, 22].

Most of the model simplification techniques described above is used for finite element applications. One of the main limitations of these techniques from the point of view of rigid body dynamics simulation is that the contact points obtained using the simplified models are drastically different than that from the original models. This is undesirable in case of rigid body dynamics simulation as its fidelity depends upon the accuracy of the contact points returned by the collision detection engine. Another class of techniques used for collision detection is mainly of two types, namely, level of detail (LOD) based and occlusion based simplification.

- *Level of Detail (LOD) based simplification :* Level of details based approach is very popular in the graphics community [23, 24, 25]. Several approaches use bounding volume hierarchies or spatial decompositions to simplify models. These approaches approximate objects with simplified bounding volumes or decompose the occupied space, to reduce the number of pairs of objects, which are needed to be checked for collision. In the area of bounding volume techniques, the reported choices for bounding volume are spheres, oriented bounding boxes, axis aligned bounding boxes and k-DOPs [26, 27, 2, 28, 29, 30, 31]. Common spatial decomposition techniques include: octrees, KD- trees, BSP-trees and Shell trees [32, 33], inner sphere trees, [34, 35], and Velocity-aligned discrete oriented polytopes (VADOP) [36]. A general treatment of LOD based techniques as well as the details on GPU based acceleration can be found in [37]. Support plane mapping based collision detection acceleration is reported by Vogiannou *et al.* [38]. Collision detection of millions of rigid bodies on GPU is performed by Liu *et al.* [39].

- *Occlusion culling based approach :* Occlusion based approaches are based on the premise that some parts of objects can never come in contact with other objects in the process of physical interaction. Vanecek adapted the concept of back-face culling, used widely in graphics to the collision detection domain as back-motion culling and developed an algorithm that runs in linear (with respect to the number of facets) time [40]. Kumar presented a sub-linear algorithm for back-motion culling by using hierarchy and coherence [41]. Redon improved the back motion culling technique by employing hierarchical decomposition [42]. Adjacency-based culling technique for continuous collision detection was developed by Tang *et al.* [43].

Most of the hierarchical bounding volume based techniques are very efficient and reduce collision computation time significantly. However, in a close proximity, models with many inaccessible and concave regions (which lie just behind the surface of collision) can increase the collision computation time (as shown in Section 1). In this paper we present an algorithm to perform contact preserving off-line model simplification by removing inaccessible facets of model pairs that interact with each other in a rigid body simulation. The reported simplification approach can be used as a preprocessing step to hierarchical decomposition, which is widely utilized for accelerating collision queries.

3. Problem Statement and Solution Approach

3.1. Problem Statement

We present some definitions before we formally describe the problem statement.

Definition 1 Let A and B be a pair of polyhedral models. A transformation T is called as *non-penetrative touch transform* if:

(i.) T is a rigid body transformation, and

(ii.) $\text{touch}(A, TB) = \text{true}$ and $\text{penetration}(A, TB) = \text{false}$

where,

$$\text{touch}(X, Y) = \begin{cases} \text{true}, \text{int}(X) \cap \text{int}(Y) = \phi \text{ and } \text{bnd}(X) \cap \text{bnd}(Y) \neq \phi \\ \text{false}, \text{ otherwise} \end{cases} \quad (1)$$

and,

$$\text{penetration}(X, Y) = \begin{cases} \text{true}, \text{int}(X) \cap \text{int}(Y) \neq \phi \\ \text{false}, \text{ otherwise} \end{cases} \quad (2)$$

int(.) is the interior of the point set, bnd(.) is the boundary of the point set and ϕ is the null set. It should be noted that we do not directly compute the *touch* and *penetration* based on the definitions presented here. These terms are introduced in order to explain the inaccessibility of facets, which are computed in subsequent sections.

Definition 2 Let A and B be a pair of polyhedral models. We define a set of *inaccessible facets of A with respect to B* as $A_{in,B} \subset A$ such that for all possible non-penetrative touch transforms T, $A_{in,B} \cap TB = \phi$.

Problem Statement : Given a pair of models A and B, generate simplified models A_s and B_s such that, the set of contact points obtained under every non-penetrative touch transform of A relative to B and A_s relative to B_s is the same. This can be mathematically expressed as following. Let $C_{A,B} = A \cap TB$ and $C_{A_s,B_s} = A_s \cap TB_s$, where T be a non-penetrative touch transformation. We want to generate simplified models A_s and B_s, such that $C_{A,B} = C_{A_s,B_s}$. Here, $A_s = A - A_{in,B}$ and $B_s = B - B_{in,A}$.

In this paper we represent all the example models (including assemblies) using triangle tessellated models. This approximation is valid as in most of the publicly available rigid body simulators the input model requirement is triangular tessellation.

3.2. Overview of the Approach

The overall model simplification approach is depicted in the form of a flowchart in Figure 4 and individual computational steps are described below.

(i.) In order to make contact with any facet f on the concave portion of A, model B will need to access facet f via some passage or opening on A (described in Section 4 and Figure 5(b) and 6(b)). These passages and openings impose restrictions on the size of B that can reach f (*e.g.*, if B is very large compared to the available passage and opening, then it cannot reach f). Section 4 presents a formal approach in Algorithm 1 to automatically identify and represent concave regions, openings, and passages.

(ii.) We are interested in knowing how deep B can enter into a passage of A. Usually parts with large cross-section with respect to a passage can only enter the passage up to a certain distance before the cross-section becomes the bottleneck. Parts have different cross-sections in different directions. Hence, in order to figure out if a part can enter an opening or passage, we need pose independent measure of its size along its length. Computing this information exactly is not practical. So we have developed method to compute a conservative estimate of this key information. We refer to this measure as part section signature. Section 5 describes Algorithm 2 for computing the part section signature.

(iii.) For a given passage on A and part section signature of B, we analyze if B is small enough to enter a passage of A and reach f or not. Facets on the concave portions of A that cannot be reached by B due to the large size of B with respect to the available passages and openings on A can be removed during the contact determination step. Section 6.1 describes Algorithm 3 to determine and remove the list of inaccessible regions of the model with respect to another model for a given model pair to obtain simplified models.

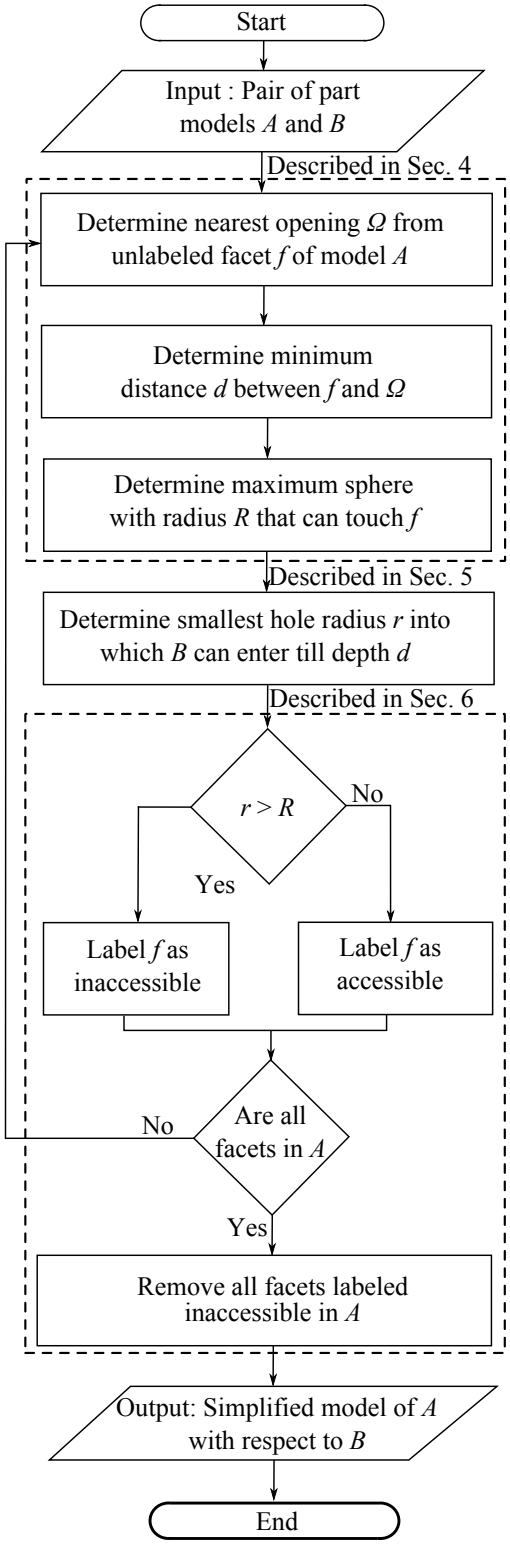
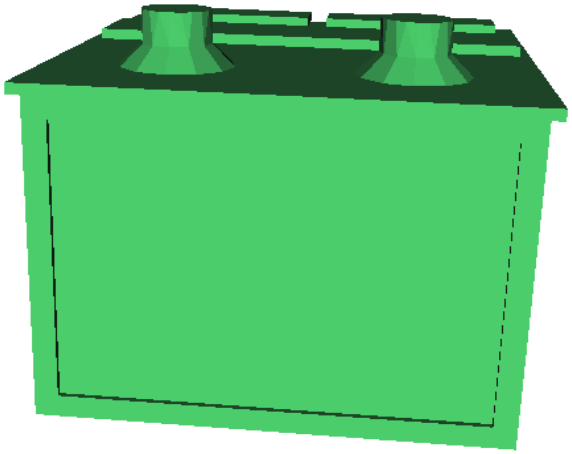
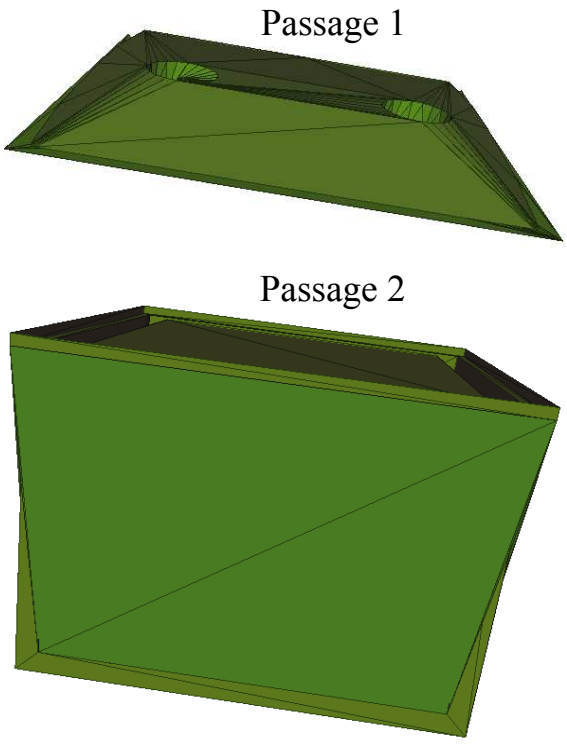


Figure 4: Flowchart of contact preserving model simplification process.

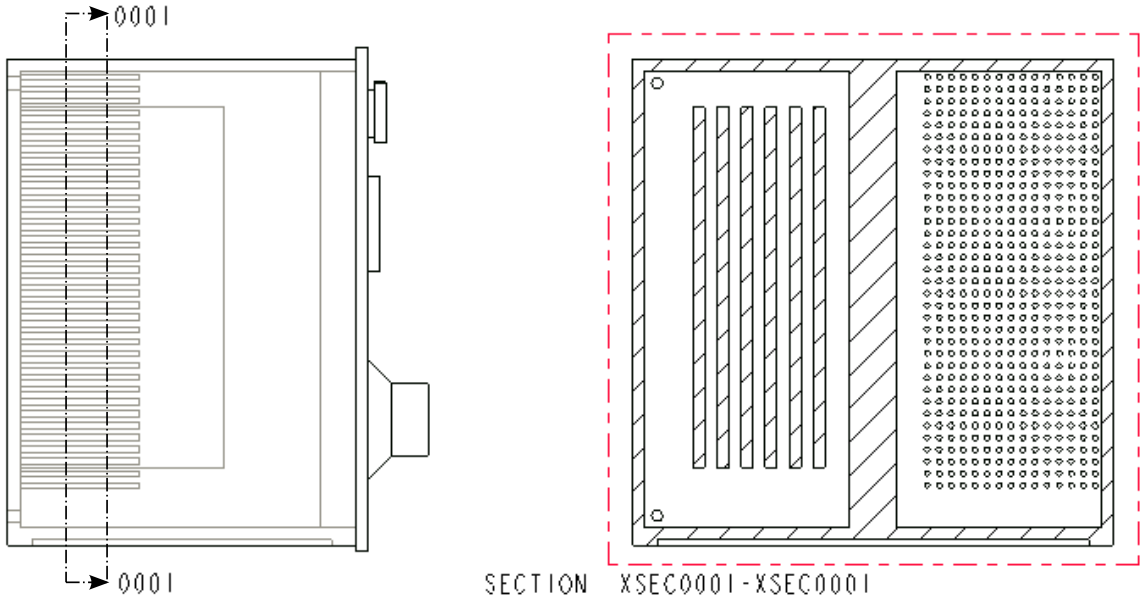
(iv.) Finally, for a given number of models in a rigid body simulation scene, we present a greedy optimization based approach to reduce the number of pair-wise simplified models to satisfy memory constraints. Section 6.2 discusses the approach in Algorithm 4.



(a) 3D CAD model for A.



(b) Passages of A.



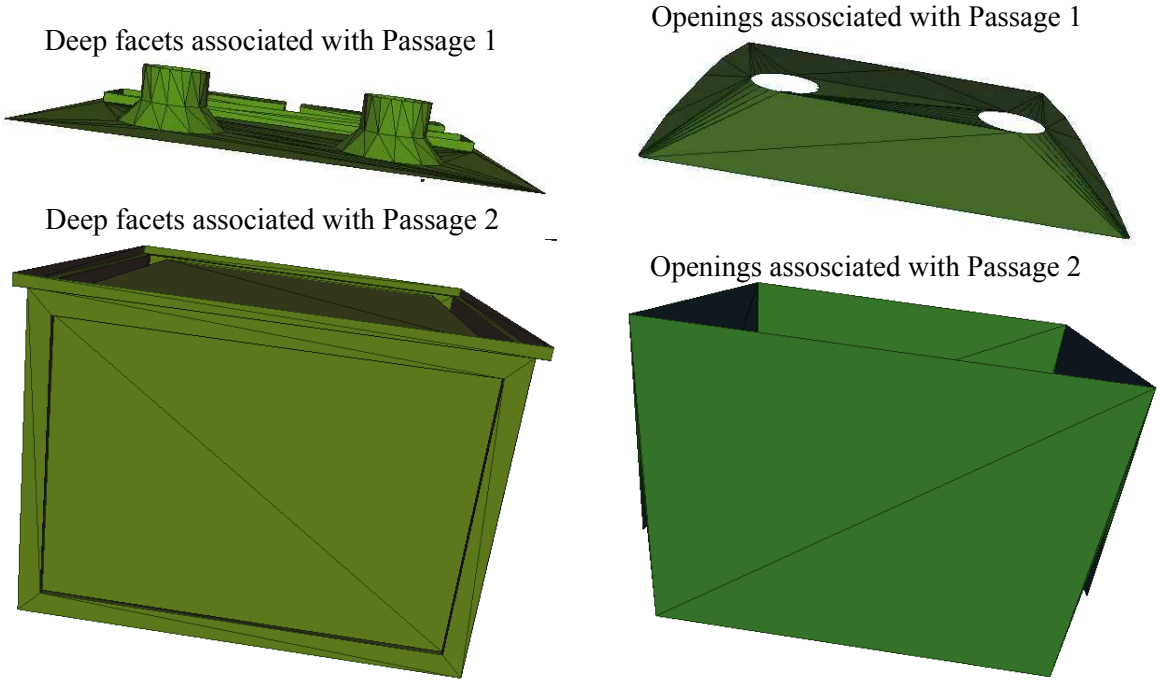
(c) 2D cross-section view of A.

Figure 5: 3D CAD model and passage set (passages of A are obtained by subtracting the A from the convex hull of A). Passages denote regions in space through which facets of a model are touched by another model without penetration.

4. Determining Passages

We first define the notion of passages, deep facets, and openings and then present our approach to determine them for a given model.

Definition 3 We define a passage set P for a given model A as the set of $3D$ point sets $p_{A,i}$, such that, all points in $p_{A,i}$ belongs to the interior of the convex hull of A but do not belong to the



(a) Deep facets of A corresponding to each passage shown in Figure 5.

(b) Openings of A corresponding to each passage shown in Figure 5.

Figure 6: Deep facets and openings of A.

interior of A.

$$P(A) = A_c - A \tag{3}$$

where, $P(A)$ is the passage set of model A and A_c is the convex hull of model A. Also, $P(A) = \cup p_{A,i}$ and $p_{A,m} \cap p_{A,n} = \phi$ for all $m \neq n$, where, each $p_{A,m}$ represents a mutually exclusive passage of A. The passage set of the model shown in Figure 5(a) are illustrated in Figure 5(b). There are only two passages because of the presence of a hole, which connects the inner and outer shell (see Figure 5(c)).

Definition 4 A *deep facet* F_D of a model is defined as a set points p_i satisfying, $p_i \in int(A_c)$ and $p_i \in bnd(A)$, $\forall p_i \in F_D$.

The deep facets of the model shown in Figure 5(a) are illustrated in Figure 6(a).

In other words, deep facets of a model are those bounding facets of the model that strictly lies in the interior of the convex hull of the model.

Definition 5 An opening Ω of model A is a set of points satisfying following:

(i.) Ω is a connected set, and

(ii.) $p_i \in bnd(A_c)$ and $p_i \notin bnd(A)$, $\forall p_i \in \Omega$.

In other words, an opening of a model A is a connected virtual face that bounds a passage and strictly lies on the boundary of convex hull of A but does not lie on the boundary of A. Openings are actually computed by determining the tessellation (triangular facets) of the virtual face which lies on the convex hull of the model but not on the model itself and the procedure for the same is explained in this section in step 2. Openings of model A are shown in Figure 6(b).

Definition 6 An opening Ω and a deep facet F_D are said to be connected if they lie on the same passage $p_{A,i}$.

Definition 7 *Opening distance* of a deep facet F_D *connected* to openings Ω_j belonging to a passage $p_{A,i}$ is the length of a line segment between a pair of points $p_m \in \Omega_j$ and $p_n \in F_D$, such that the line segment L connecting p_m and p_n satisfies following two conditions, namely,

(i.) all points on L connecting p_i and p_j strictly lies inside the volume of the cell $p_{A,i}$, and

(ii.) L does not intersect any deep facet belonging to $p_{A,i}$.

If a line segment L satisfying above two conditions does not exist then opening distance is equal to infinity.

The largest sphere that can be inscribed inside the volume of $p_{A,i}$ and is tangential to F_D is called as *opening sphere* and the radius of the opening sphere is called as *opening radius*.

Some deep facets of A lying on $p_{A,i}$ may not be connected to any opening. In case of such an orphan deep facet, the opening distance is set as infinity. Opening sphere is determined for such orphan facets by finding out the largest non-intersecting inscribed sphere inside the passage $p_{A,i}$ and tangential to the deep facet.

Thus, each deep facet can have multiple (at least one) opening distances and corresponding to each connected opening there is an associated opening sphere.

Another interpretation of opening sphere is that for a given opening of a deep facet, any sphere with radius larger than the opening sphere cannot *touch* the deep facet without *penetration*.

Following steps are used to determine passage set, deep facets, and openings of a given model and to represent the model as a graph whose nodes contain deep facets and openings while arcs represent the connectivity information.

Algorithm 1 - Construct Connectivity Graph

Input - Polyhedral model A.

Output - Connectivity graph representation of A.

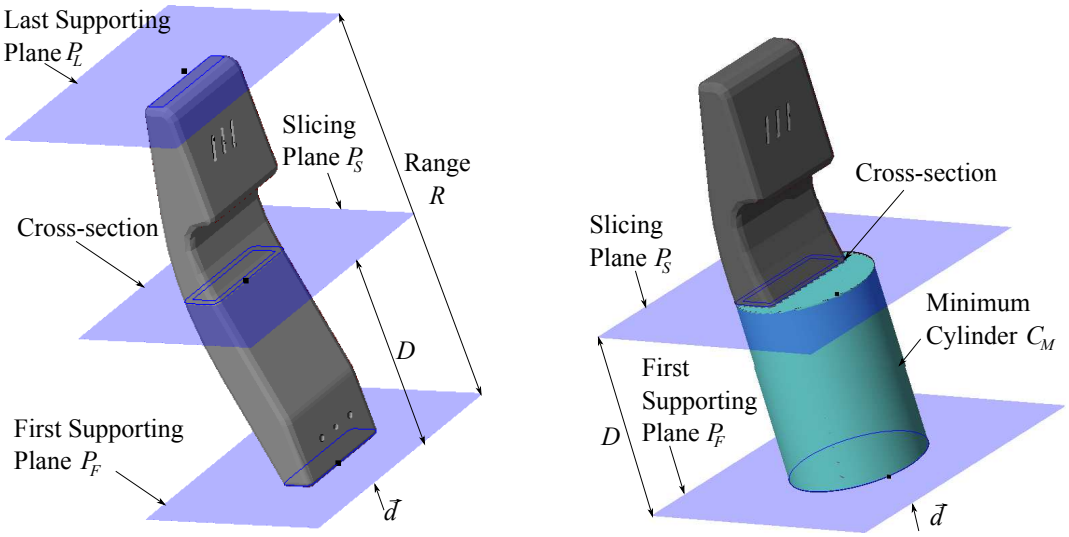
Steps:

(i.) *Find the passage set A* : Determine the convex hull A_c of model A and perform the Boolean subtraction of A_c and A. This results into a non-manifold passage set $P(A)$. The passage set of the model shown in Figure 5(a) is shown in Figure 5(b). We express A and A_c as Nef-polyhedrons as implemented in CGAL [44, 45].

(ii.) *Find openings of A* : Iterate through each bounding facet of passages $p_{A,i}$ of $P(A)$. Label every facet in $p_{A,i}$, that do not lie on bounding facets of A but do lie on bounding facets of A_c as opening facets. Group all contiguous facets marked as opening facets and store as the sets of openings corresponding to the passage $p_{A,i}$. We used CGAL's AABB spatial search structure to accelerate this process [44]. Openings of the model shown in Figure 5(a) is depicted in Figure 6(b).

(iii.) *Find deep facets of A* : Iterate through bounding facets of model A and find all facets of A that are coincident with bounding facets (that are not openings) of the passage $p_{A,i}$, and label them as deep facets.

(iv.) *Construct a connectivity graph for each passage $p_{A,i}$* : Construct a connectivity graph for $p_{A,i}$ and mark each node either as opening or deep as determined in steps (ii) and (iii). Determine connected opening nodes for each deep node and add to the connectivity graph. A node of the graph represents either a deep facet or an opening while an arc represents the connectivity between nodes. Iterate through nodes representing deep facets and determine the bounding sphere for each connected opening and store it as the opening sphere of the connected deep facets. This is because bounding sphere of an opening is always larger than or equal to the *opening sphere* and we need an upper bound on the opening sphere radius. Compute the minimum distance between the deep facet and connected openings and store as opening distance for the deep facet. If some deep facet is not connected to any opening



(a) Supporting planes and range along direction vector $\vec{d}$.

(b) Minimum cylinder along direction vector $\vec{d}$ and for depth D.

Figure 7: Illustration of supporting planes, range, and minimum cylinder.

then use an opening (with the largest bounding sphere) belonging to $p_{A,i}$ for computing the opening sphere. For a deep facet not connected to any opening, set the opening distance to a large value. Return the connectivity graph.

5. Construction of Part Section Signature

In order to answer the question that whether a given deep facet on A is *accessible* by the given model B, we need to represent model B as a function of *minimum opening distance* of deep facets of A returning a lower bound on the size of B. If we have such a function, a comparison of the lower bound on the size of B with the bounding sphere of the connected opening of the deep facet of A can answer the question about accessibility of the deep facet of A by B. We call such a function of minimum opening distance of deep facets of A returning lower bound on the size of B as part section signature or PSS of B.

Definition 8 For a given model M and a direction vector $\vec{d}$, we define a plane P_F as the *first supporting plane*, if it satisfies following conditions (see Figure 7(a)):

(i.) P_F is normal to $\vec{d}$.

(ii.) M lies entirely in the closed half-space of P_F in the direction of $\vec{d}$ and at least one point on M lies on P_F.

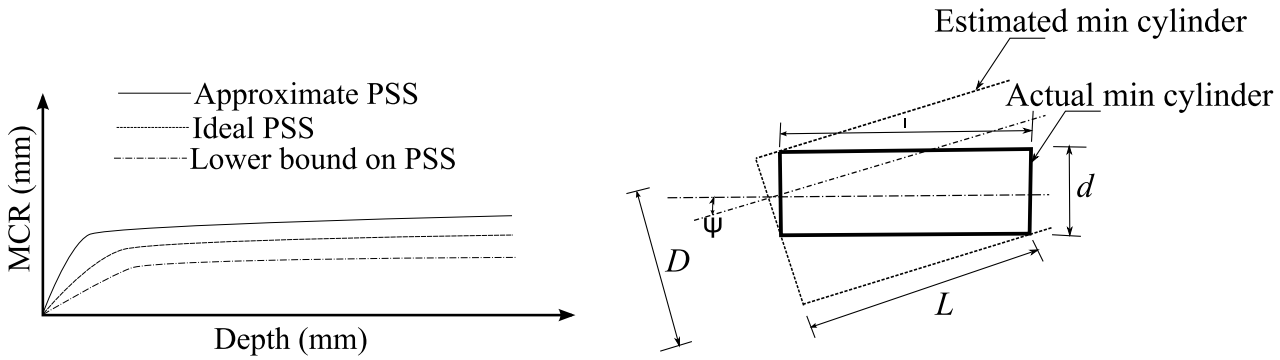
Definition 9 For a given model M and a direction vector $\vec{d}$, we define a plane P_L as the *last supporting plane*, if it satisfies following conditions (see Figure 7(a)):

(i.) P_L is normal to $\vec{d}$.

(ii.) M lies entirely in the closed half-space of P_L in the opposite direction of $\vec{d}$ and at least one point on M lies on P_L.

Definition 10 For a given model M and a direction vector $\vec{d}$, we define *range (R)* as the distance between P_F and P_L (see Figure 7(a)).

Definition 11 For a given model M, direction vector $\vec{d}$, and a distance D where $0 \leq D \leq R$, we define a plane P_S as a *slicing plane*, if it satisfies following conditions (see Figure 7(a)):



(a) Comparison of approximate and theoretical PSS.

(b) Error estimation.

Figure 8: PSS error estimation.

(i.) P_S is normal to $\vec{d}$.

(ii.) Distance between P_F and P_S is equal to D.

Definition 12 For a given model M, a direction vector $\vec{d}$, and a distance D where $0 \leq D \leq R$, we define *minimum cylinder* (C_M) as the minimum possible diameter cylinder (with the axis parallel to $\vec{d}$) that contains all the points on M lying between P_F and P_S of M. The *minimum cylinder radius* (MCR) is thus defined as the radius of C_M, which is a function of model geometry M, direction vector $\vec{d}$ and depth D (see Figure 7(b)).

Definition 13 For a given model M, we define *part section signature* (PSS) as,

$$\text{PSS}(M, D) = \min_{\vec{d}} \text{MCR}(M, \vec{d}, D) \tag{4}$$

where, $\vec{d}$ is the direction of approach and D is distance between the P_F and the P_S for direction vector $\vec{d}$.

In other words, PSS is a $2D$ representation of M that gives the relationship between a given depth D of a blind hole (with minimum possible radius) into which M can enter and reach the blind end of the hole, and the radius of that blind hole. It should be noted, that PSS is a monotonically non-decreasing function of depth D.

To determine, whether a model B can enter an opening Ω on A to a given depth δ without penetrating into A, ideally requires to try out all possible directions of approach $\vec{d_j}$ for B and to find the corresponding set of cross-sections CS_j obtained by intersecting B with the slicing plane for the direction $\vec{d_j}$ and depths $0 \leq D \leq \delta$. If at least one cross-section CS_j is such that it can be contained in maximum inscribed polygon of Ω, we can claim that model B enters the cavity opening Ω to a distance of δ. However the above approach is not practical because of the following reasons:

(i.) There are infinitely many possible orientations and considering all of them is not feasible.

(ii.) The cross-section CS_j is in general non-convex and so are the cavity openings Ω. A containment test of non-convex polygons in other non-convex polygons is computationally expensive operation.

We addressed above problems by computing a conservative estimate of the theoretical PSS by considering a discrete set of directions of approach. The PSS information is used for determining whether a model can enter an opening to a specified depth. If it is found that the model cannot enter the opening to a given depth then the *connected* deep facets are removed to simplify the model.

We are interested in conservative simplification (*i.e.*, we do not want to change contact points due to approximations in PSS). Hence, we computed a lower bound on the theoretical PSS and used it as a conservative estimate of the theoretical PSS. The approximated, theoretical, and the estimated lower bound on the PSS is shown schematically in Figure 8(a). The estimated lower bound on PSS of a model guarantees that the model cannot enter a given hole if the theoretical PSS of the model does not do so. This approach ensures that a facet is deleted only when it is truly inaccessible. However, sometimes a facet may be inaccessible but our approach will not be able to eliminate that due to the use of the estimated lower bound on the PSS.

In our approach, we determine a discrete set of directions along (θ, ϕ) in spherical coordinate system with $\theta = \alpha, 2\alpha, ..., 2\pi$ radians and $\phi = \alpha, 2\alpha, ..., 2\pi$ radians, where α is the angular increment. For each direction, we determine MCR at various depths and then find out an approximate PSS. To estimate how much the actual PSS can be less than the computed approximate PSS, consider Figure 8(b). The ideal cylinder (with diameter d) enveloping a model to a depth of l is shown by a rectangle drawn in solid line in Figure 8(b). The estimated minimum cylinder (with diameter D) enveloping the model is shown by a broken line in Figure 8(b), whose axis is at an angle of $\psi \leq \alpha$ with the axis of ideal minimum cylinder. Now, using the geometry,

$$dcos(\psi) + lsin(\psi) = D \Rightarrow \frac{d}{D} = sec(\psi) - \frac{l}{D}tan(\psi),$$

$\because \psi$ is very small and expressed in radians, $sec(\psi) \approx 1$ and $tan(\psi) \approx sin(\psi) \approx \psi$,

$$\Rightarrow \frac{d}{D} \approx 1 - \frac{l}{D}\psi, \Rightarrow \frac{d}{D} = 1 - \frac{l}{D}\psi \geq 1 - \frac{l}{D}\alpha, \because \psi \leq \alpha,$$

thus,

$$d \geq D\{1 - \frac{l}{D}\alpha\} \tag{5}$$

We use equation 5 to estimate the lower bound on the PSS by multiplying the approximate PSS by factor $e_f = 1 - \frac{l}{D}\alpha$. A few points to note about the error estimation parameter e_f are as follows:

(i.) When $\frac{D}{l} \leq \alpha$, the error estimation parameter e_f becomes negative. Physically, this means that the angle increment α is not small enough. We set e_f to zero under the above stated condition, which makes the lower bound on theoretical diameter to be zero and thus no simplification (as a cylinder with zero diameter can enter hole of any diameter) occurs.

(ii.) Estimated error e_f approaches unity as the value of α reduces. This means that estimated cylinder diameter D is close to actual cylinder diameter d when α is small. However, computation time of PSS increases at inverse square rate of decreasing α and thus a choice of trade off between better estimation and computation time can be made by selecting appropriate value of α.

Following algorithm is used to compute the lower bound on the PSS of model B.

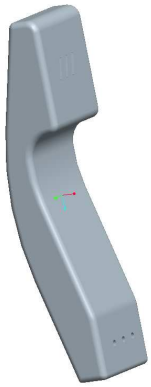
Algorithm 2 - Compute Lower Bound on Part Section Signature (PSS)

Input - Polyhedral representation of model B and angular sampling increment α.

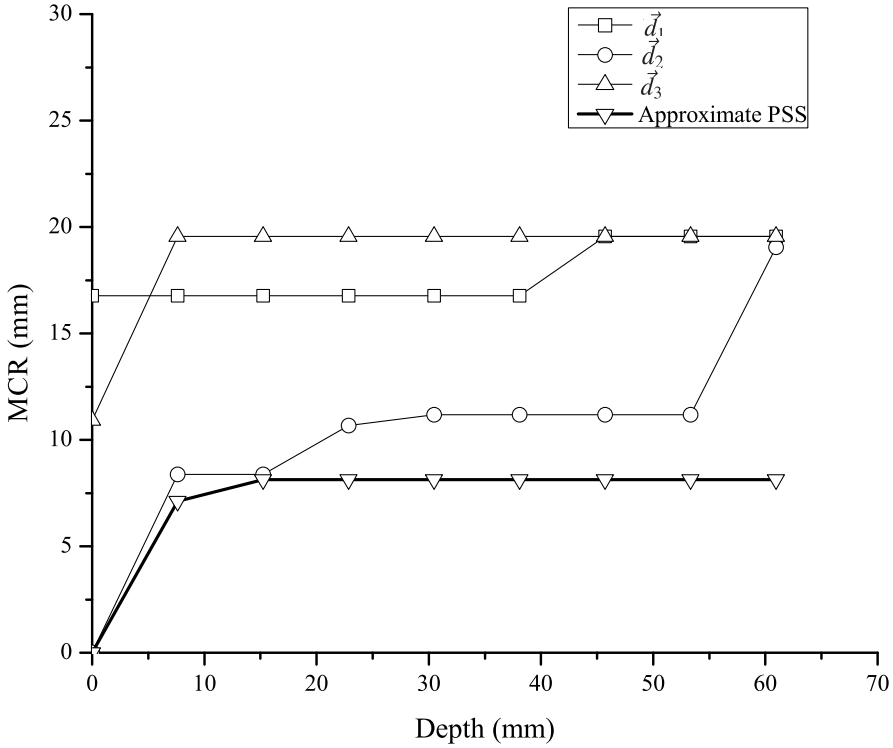
Output - PSS of model B.

Steps:

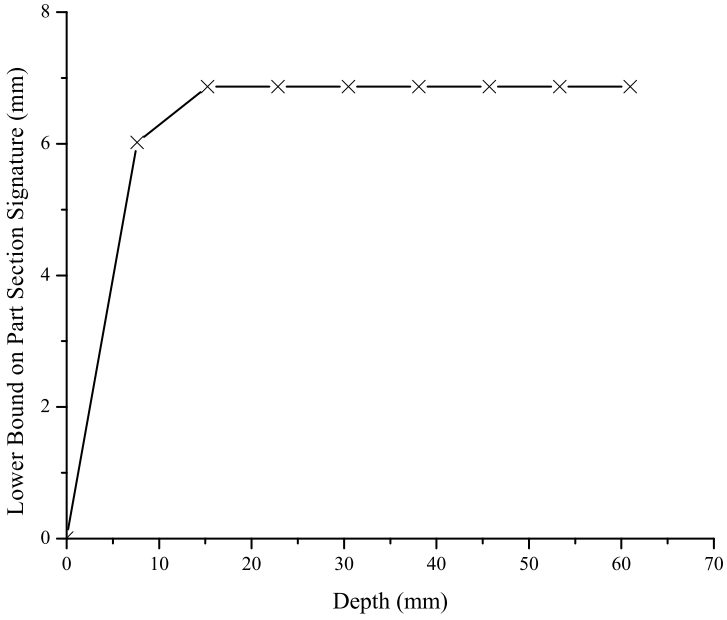
(i.) *Find list of direction vectors* : Determine candidate directions along (θ, ϕ) in spherical coordinate system with $\theta = \alpha, 2\alpha, ..., 2\pi$ and $\phi = \alpha, 2\alpha, ..., 2\pi$, where α is the angular increment. In our experiments, we selected the value of α as 0.26 radians.



(a) Part Model B



(b) Computation of approximate PSS. MCRs along 144 sample directions are computed and the minimum of all the MCRs is called approximate PSS. Only 3 directions namely, d_1, d_2, and d_3 are shown here for clarity.



(c) Part Section Signature (PSS) of model B is determined as lower bound on approximate PSS.

Figure 9: Computation of Part Section Signature (PSS) of model B. PSS is a function that returns the radius of the smallest cylindrical hole for a given depth that a model will protrude without penetration.

(ii.) *Find the slicing plane set along each direction vector* : Find the first supporting plane, last supporting plane and range along each candidate direction. After this, generate slicing planes normal to the direction vector and add to the slicing plane set using the list of depths of deep facets on A. If the depth for a direction is larger than the *range* in that direction then the depth is set equal to the range.

(iii.) *Compute cross-sections along each direction vector* : Intersect the model with each plane in the slicing plane set to obtain the point set lying in the negative half space of the intersecting plane. After this, determine the MCR of each point set. In order to eliminate redundant computations, only compute cross-sections for the range of depths of deep facets obtained in Algorithm 1. For each candidate direction, store the MCR variation along it. Figure 9(b) shows the variation of MCR along three directions ($\vec{d_1}$, $\vec{d_2}$, and $\vec{d_3}$) for the model B.

(iv.) *Compute approximate PSS* : Determine MCR for each direction determined in step (i.). After this, for each direction, find out the minimum among all MCRs corresponding to each interval and generate a lookup table for each direction storing the depth and the minimum MCR. This lookup table is called the approximate PSS. In Figure 9(b) only three representative directions $\vec{d_1}, \vec{d_2}$ and $\vec{d_3}$ are shown for clarity.

(v.) *Apply conservative bound on the approximate PSS* : Multiply factor e_f given in Equation 5 to approximate PSS and determine the lower bound on the PSS. The estimated lower bound on the PSS is shown in Figure 9(c). Return estimated PSS of the model B.

6. Model Simplification

In this section we describe model simplification algorithms based on data-structure developed in Sections 4 and 5.

6.1. Pair-Wise Model Simplification

Each deep facet of the connectivity graph of passages of model A is traversed to find out their accessibility through their corresponding connected openings using the PSS of the model B. The inaccessible deep facets of A with respect to B are subsequently deleted to generate the simplified model A_s. Steps followed for pair-wise simplification are listed in Algorithm 3.

Algorithm 3 - Perform Pair-Wise Model Simplification

Input - Connectivity graph representation of models A and B and PSS of models A and B.

Output - Simplified models A_s and B_s.

Steps:

(i.) *Find inaccessible facets of A with respect to B* : Iterate through openings corresponding to every deep facet of A and determine the smallest opening distance and the largest opening sphere. The procedure for determining accessibility of a facet f of A with respect to B is shown in Figure 10. The PSS (r) of the model B at depth D is shown in Figure 10. The minimum distance from the facet f to the corresponding opening is D. Opening sphere of facet f has radius R. To find whether facet f in A is accessible by B or not, R is compared with r. If $r > R$ then it can be concluded that B cannot touch facet f of A and hence f can be labeled as inaccessible by B (see Theorem 1).

(ii.) *Remove inaccessible facets of A with respect to B* : The list of inaccessible facets of A are then removed from A and a simplified model A_s is generated.

(iii.) *Swap A and B and repeat step (i.) and (ii.) to obtain simplified model* B_s.

(iv.) *Return* A_s *and* B_s.

Theorem 1. *Let,* F_D *be a deep facet of model A. Let the minimum opening distance for* F_D *be* lp_{min} *and maximum opening radius be* r_{max}*. If* $PSS(B, lp_{min}) > r_{max}$*, where B is a model, then B cannot touch* F_D*.*

Proof. We can prove this by contradiction. Let us suppose that the model B can touch F_D for a non-penetrative touch transform T through an opening Ω_j with opening distance lp_j.

Let the opening radius corresponding to opening Ω_j be r_j. Let us suppose that transform T orients the model B along vector $\vec{d}$ for the given coordinate system.

$\Rightarrow MCR(B, \vec{d}, lp_j) \leq r_j$

We know, $r_j \leq r_{max}$.

$\Rightarrow MCR(B, \vec{d}, lp_j) \leq r_{max}$

Also, $lp_j \geq lp_{min}$ and using the fact that *minimum cylinder* for depth lp_j is larger or same as *minimum cylinder* for depth lp_{min} for model B.

$\Rightarrow MCR(B, \vec{d}, lp_j) \geq MCR(B, \vec{d}, lp_{min})$

$$\Rightarrow MCR(B, \vec{d}, lp_{min}) \leq r_{max} \tag{6}$$

Now, from the given conditions, $PSS(B, lp_{min}) > r_{max}$ and Equation 4, $PSS(B, lp_{min}) \leq MCR(B, \vec{d}, lp_{min})$, for any arbitrary direction $\vec{d}$.

We have,

$$MCR(B, \vec{d}, lp_{min}) > r_{max} \tag{7}$$

Inequalities 6 and 7 contradict each other and hence under the given conditions, model B cannot touch the deep facet F_D.

Hence the proof follows. □

6.2. Model Simplification for n Models

Let us suppose that there are multiple models in a scene and let, the total number of geometries required for representing all models be n. It is thus evident that if we perform pair-wise simplification alone, we need to store up to n simplified models for each unsimplified geometry (including interactions among models with identical geometry). Thus, for a scene with n different geometries, we need to store n^2 simplified models. However, in practice, we can reduce the number of simplified models to be stored considerably by using similarity of simplified models. Moreover, in order to meet the memory constraints, we can relax the simplified models by merging simplified versions of models that do not change potential contact points. We present a greedy approach to reduce the number of simplified models to satisfy storage memory constraints. Let us suppose that there are n types of models in a scene namely, P_1, P_2, P_3,..., P_n. Let us represent the pair-wise simplified version of a model P_i with respect to P_j as S_i^j $(1 \leq i, j \leq n)$. Let us define an operator *owner*, which returns the unsimplified model for a given simplified model. The algorithm for the model simplification for n models is described below.

Algorithm 4 - Perform Model Simplification for n Models

Input - Pair-wise simplified models S_i^j $(1 \leq i, j \leq n)$ and available memory M.

Output - Reduced set of simplified models.

Steps:

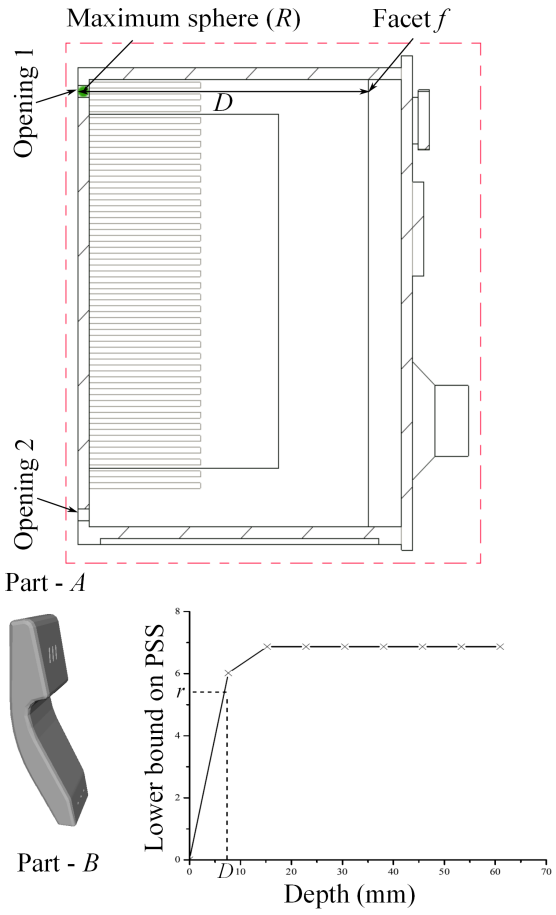


Figure 10: Model simplification procedure (inaccessible facets are detected and then removed).

(i.) Initialize a set $\mathcal{S} = \{\phi\}$ and use the following loop to populate $\mathcal{S}$.

$FOR\ i = 1$ TO n

$FOR\ j = 1$ TO n

$IF\ i \neq j$

$owner(S_i^j) = P_i$

$INSERT\ S_i^j\ IN\ \mathcal{S}$

$END\ IF$

$END\ FOR$

$END\ FOR$

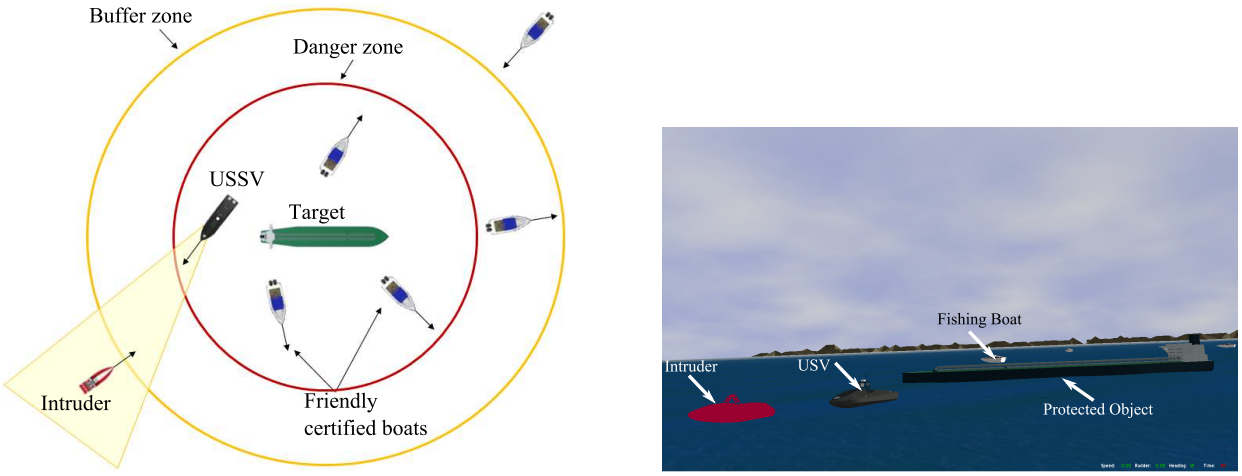
(ii.) Reduce the number of simplified models that need to be stored in order to respect the available memory constraints using following scheme.

$WHILE\ memory\_size\,(\mathcal{S}) > M$

$FIND\ MODEL\ PAIRS\ S_i, S_j \in \mathcal{S}\ SUCH\ THAT\ owner\,(S_i) = owner\,(S_j)$ AND $i \neq j$

$\{m, n\} = argmax_{i,j}((S_i + S_j) - (S_i \cup S_j))$

$CREATE\ S_{m,n} = S_m \cup S_n$



(a) Intruder blocking mission.

(b) Screen-shot of the virtual environment.

Figure 11: USV intruder blocking game.

$$owner\,(S_{m,n}) = owner\,(S_m)$$
$$REMOVE\ S_m\ AND\ S_n\ FROM\ \mathcal{S}$$
$$INSERT\ S_{m,n}\ IN\ \mathcal{S}$$
$$END\ IF$$
$$END\ WHILE$$

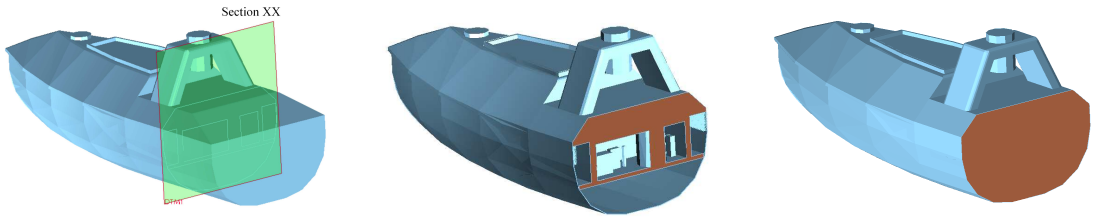
(iii.) Return reduced set $\mathcal{S}$.

In the above algorithm, each pair of simplified versions of a model is tested for uncommon facets. The simplified model pair of same owner, which has the largest number of uncommon facets, is fused to generate a relaxed simplified model. This process is repeated until, number of simplified versions of model satisfy the storage memory constraint.

7. Implementation and Results

We developed a prototype software implementation of the contact preserving model simplification algorithm discussed in this paper. The programming platform was chosen as VC++ (version 8) using CGAL 3.6 on Windows Vista operating system [44]. Our implementation takes STL files as input and after simplification, generates output in the same format. Also, we represent assembly models in stl format. In this section we present some results of model simplification approach developed in this paper, mainly in two robotics applications, namely, unmanned surface vehicle (USV) simulation and ground vehicle simulation. We also present some more examples to demonstrate the utility of the technique developed in this paper.

We have been developing a virtual environment to simulate surveillance operation for USVs [46, 47], in which we used the simplified models generated using the approach discussed in this paper. The virtual environment simulates a maritime mission in which a remote controlled boat tries to protect a valuable target from an intruding boat. Figure 11(a) shows the schematic view of the mission, where the intruder boat attempts to reach to the protected object by crossing the buffer zone and the danger zone. The USV's task is to block the intruder. The level of aggression, with which the USV blocks the intruder depends upon the position of the intruder (*i.e.*, whether the intruder lies in the buffer zone or the danger zone). The USV should block the intruder and at the same time should avoid collision with the other dynamic obstacles, which represent the harmless



(a) USV model with 60030 facets.

(b) Cut-through view (about plane XX) of unsimplified USV model with 60030 facets.

(c) Cut-through view (about plane XX) of simplified USV model with 1120 facets.

Figure 12: Boat model used in the virtual environment.

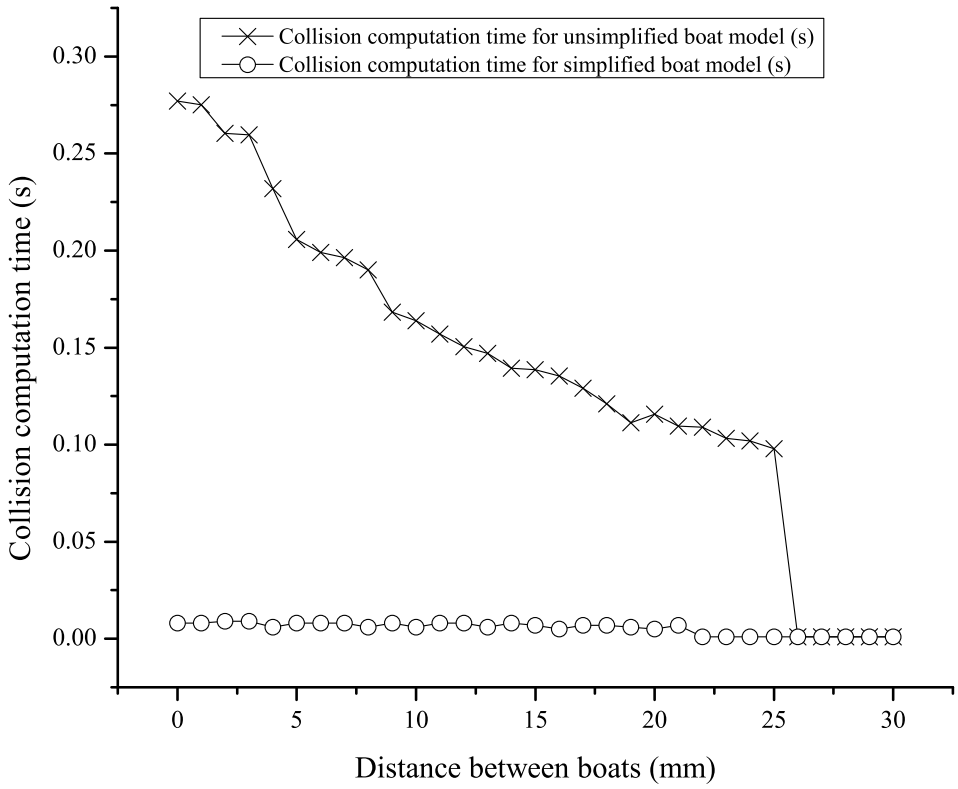
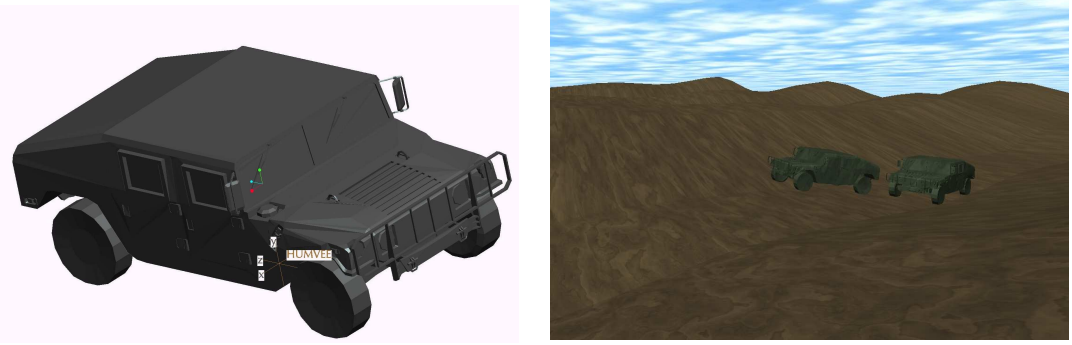


Figure 13: When the USVs are far apart ($D > 25mm$) collision computation time is very small because of the efficient hierarchical data-structure used by the existing collision detection approaches but in close proximity ($D < 25mm$) model simplification technique developed in this paper in conjunction with hierarchical data-structure reduces collision computation time by a factor of 34.6.

artifacts such as fishing boats. An important maneuver in this operation is the active blocking of the intruder boat by the remotely controlled USV by steering towards the front of the intruder boat in the danger zone. This application requires interactive rigid body dynamics simulation to train operators driving boats using tele-operation. A meta-model is used to determine forces (such as ocean wave, gravity, buoyancy, etc.) acting on boats due to the water [46]. A screen-shot of the virtual environment is shown in Figure 11(b). We chose RAPID collision detection engine for collision queries [2].

In our test, we imported a boat model (with 60030 facets) to represent both the USV and the intruder and used this detailed model for the purpose of visualization as well as the collision detection. In the test, boats were gradually moved towards each other and the time taken for the



(a) Ground vehicle geometric model with 10686 facets.

(b) Simulation environment developed in OpenDynamics [48].

Figure 14: Ground vehicle simulation environment developed in OpenDynamics [48].

collision query in each time step was recorded. The worst-case collision detection time obtained with the detailed model (with 60030 facets) used as both the visualization and collision model was found to be 0.277 s. Using our model simplification approach, the simplified boat model obtained had 1120 facets (as shown in Figure 12(c)). Regions such as ribs, inner parts, motors, batteries, etc. are removed automatically using the model simplification approach as shown in Figure 12(c). Other concave regions such as facets just beneath the deck, although are deep facets are not labeled as inaccessible as it is connected to a large opening. It should be noted that only ribs lying just behind the collision surface contribute significantly to the collision detection time. Other regions that are very far from the point of collision have negligible effects on the collision detection time. We repeated the simulation using the simplified model (with 1120 facets) as collision model and the detailed model (with 60030 facets) as visualization model. The worst-case collision detection time obtained with the simplified model as the collision model was found to be 0.008 s. The contact points obtained in both cases (with unsimplified and simplified collision models) were *exactly* same. The variation of collision detection time with respect to the distance between boats is shown in Figure 13. The speed-up in the worst-case collision detection time by using the simplified models was thus, by a factor of 34.6.

We present another example (see Figure 14) application of developed simplification technique in the domain of ground vehicle simulation. Figure 14(a) shows geometric model of a vehicle with 10686 bounding facets. Figure 14(b) shows a snapshot of an environment developed in OpenDynamics [48] engine with vehicles represented by the model shown in Figure 14(a). The vehicle moves on a treacherous terrain and interact with the other vehicle and the terrain during the mission. We applied the model simplification approach developed in this paper to this problem and obtained simplified model of the vehicle with 2086 facets. In this case too, the inaccessible regions such as engine, steering mechanism, etc. contribute to a large number of facets, which gets removed by the application of the model simplification techniques developed in this paper. The worst-case collision computation time in close proximity was found to be 0.021 s for unsimplified model, whereas 0.007 s for the simplified model of the vehicle. Thus, using the contact preserving model simplification approach, speed-up of about a factor of 3.0 was obtained (see Figure 15) for the worst-case collision detection time.

We also tested our implementation on several different pairs of models (shown in Figure 16) that have interior inaccessible regions. Representative results of model simplification are shown in Table 1. The table shows the reduction in the facet count of models without changing their potential collision contact points. For the reported test models we found that the reduction in the number of facet count ranges from factor of 3.2 to 22.7 depending on the model complexity and

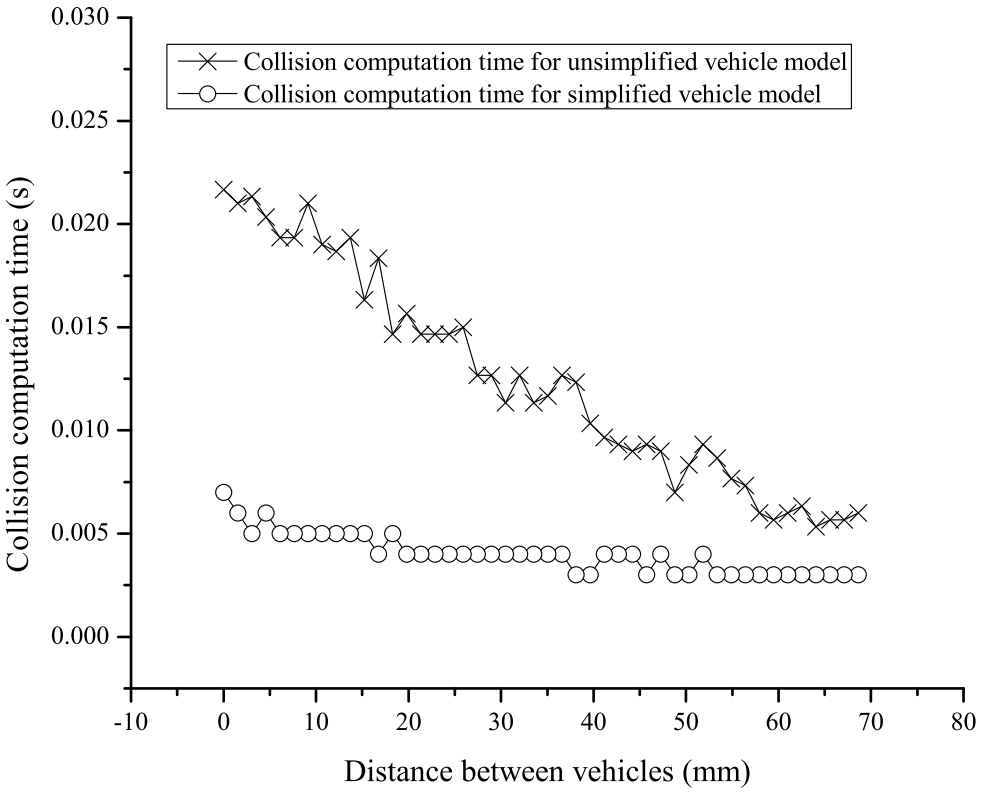


Figure 15: When the ground vehicles are far apart ($D > 60mm$) collision computation time is very small because of the efficient hierarchical data-structure used by the existing collision detection approaches but in close proximity ($D < 60mm$) model simplification technique developed in this paper in conjunction with hierarchical data-structure reduces collision computation time by a factor of 3.0.

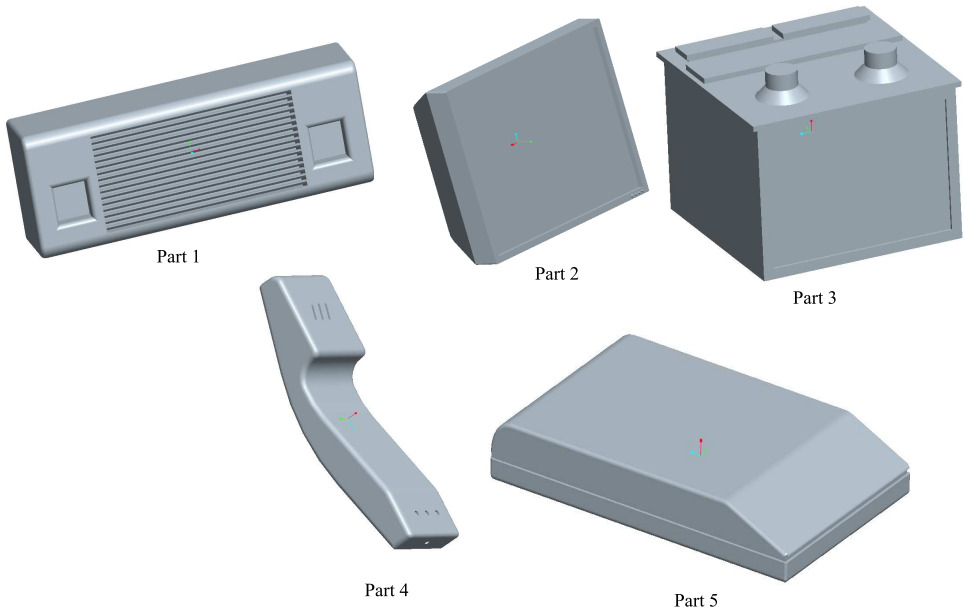


Figure 16: Example models.

the collision context.

We repeated the test described in Figure 1 for each of the model pairs in Table 1. The worst-

Table 1: Representative simplification results.

Models	Facet counts of unsimplified models	Facet counts of pair-wise simplified models					Facet counts of relaxed simplified model
		P_1	P_2	P_3	P_4	P_5	
P_1	17500	1616	1616	1616	1616	1616	1616
P_2	29304	2712	2712	2730	2730	2650	2730
P_3	13156	4070	4070	4002	4002	4002	4070
P_4	12836	752	752	752	752	752	752
P_5	30768	711	711	1356	1356	1356	1356

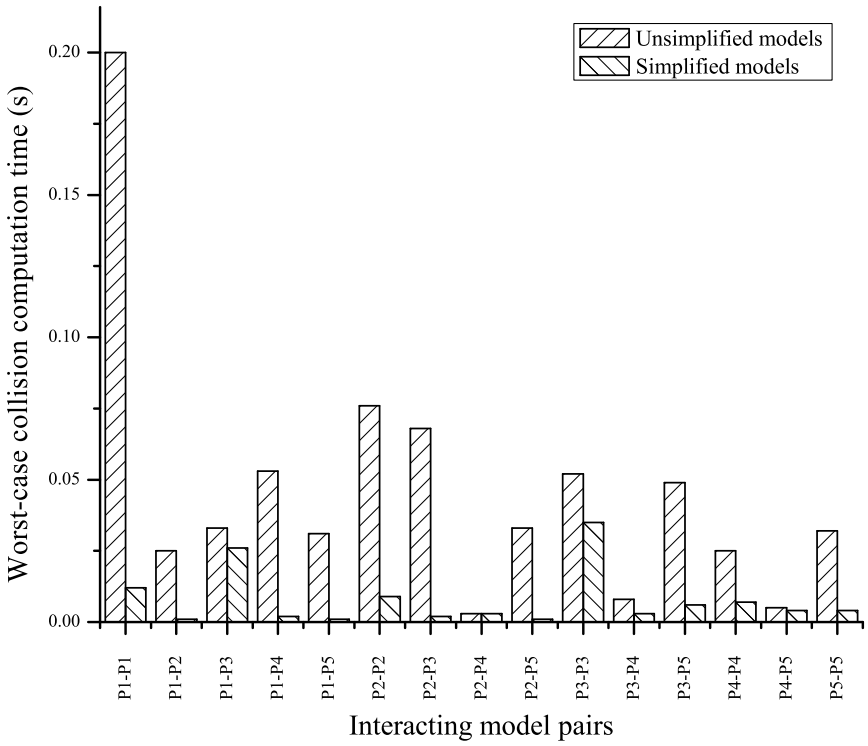


Figure 17: Worst-case collision computation times for pair-wise collision obtained using simplified and unsimplified models.

case collision detection time obtained by using unsimplified and simplified models are shown in Figure 17. The collision detection time reduced by factors in the range of 1.0 to 34.0 by utilizing simplified models over the unsimplified models. It can be seen that the speed-up factor for the model pair P_2-P_4 is unity, as there are no inaccessible facets in the close proximity of collision surface. The speed-up in worst-case collision detection time directly depends on the number of inaccessible facets near collision surface and not the total number of model bounding facets. We ran our model simplification implementation on a computer with Quad-core processor and 8 Gigabytes of RAM and it took 30 minutes to simplify the five example models for each possible interactions among themselves (including identical model interaction). So, on an average, for the models of the complexity level in Figure 16, one model took about $\frac{30}{25} = 1.2$ min for the simplification.

8. Conclusions

This paper reports an off-line contact preserving model simplification approach that removes inaccessible facets from a model with respect to another colliding model in a rigid body simulation. The main contributions of this paper are listed below.

(i.) We developed a conservative model simplification approach which guarantees that the contact points are not altered after simplification under all possible collision configurations.

(ii.) We introduced the concept of accessibility based on the part section signature and openings.

(iii.) We demonstrated reduction in the collision detection time obtained by using simplified models in a USV simulation environment (by a factor of about 34.6), ground vehicle simulation (by a factor of about 3.0), and several general examples of model pairs (upto about 34.0 depending upon the model complexity).

Since the approach helps to reduce collision detection time without altering the potential contact points, the possible applications besides the rigid body simulation include robot motion planning, assembly simulation, etc. Also, the speed-up obtained by the technique introduced in this paper can significantly improve the interactivity of simulations. If applied in a purely combinatorial manner, the presented pair-wise simplification approach will require n simplified models generated for every model in a scene consisting of n models. In order to satisfy memory storage constraints, we have developed a greedy approach to reduce the number of simplified models by generating relaxed simplified models.

A limitation of this work is the conservative approximation of approaching models by bounding cylinders that sets a lower bound on the model cross-section. This can simplify a given model with respect to another model, which can potentially collide in a contact-preserving sense but might leave some facets, which should be removed. This way, however, we guarantee that models are never over-simplified.

Acknowledgments

This research has been supported by the NSF grants CMMI-0727380 and OCI-0636164. Opinions expressed in this paper are those of the authors and do not necessarily reflect opinions of the sponsors.

[1] Y. Q. Yu, L. L. Howell, C. Lusk, Y. Yue, and H. M. Gen. Dynamic modeling of compliant mechanisms based on the pseudo-rigid-body model. *Journal of Mechanical Design*, 127(4):760–765, 2005.

[2] S. Gottschalk, M. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive technique*, 1996.

[3] A. Thakur, A. G. Banerjee, and S. K. Gupta. A survey of CAD model simplification techniques for physics-based simulation applications. *Computer Aided Design*, 41(2):65–80, 2009.

[4] G. Foucault, J. Cuillière, V. François, J. Léon, and R. Maranzana. Adaptation of CAD model topology for finite element analysis. *Computer Aided Design*, 40(2):176–196, 2008.

[5] A. Sheffer, T. Blacker, and M. Bercovier. Clustering: automated detail suppression using virtual topology. In *AMD 220. Trends in Unstructured Mesh Generation. ASME.*, Evanston. IL. USA., 1997.

[6] A. Sheffer. Model simplification for meshing using face clustering. *Computer-Aided Design*, 33:925–934, 2001.

[7] Y. G. Lee and K. Lee. Geometric detail suppression by the Fourier transform. *Computer-Aided Design*, 30(9):677–693, 1998.

[8] Lee S. H. Feature-based multiresolution modeling of solids. *ACM Transactions on Graphics.*, 24:1417–1441, 2005.

[9] S. H. Lee and K. Lee. Simultaneous and incremental feature-based multiresolution modeling with feature operations in part design. *Computer-Aided Design*, 44(5):457 – 483, 2012.

[10] D. H. Choi, T. W. Kim, and K. Lee. Multiresolutional representation of B-Rep model using feature conversion. *Transactions of the Society of CAD-CAM Engineers.*, 7(2):121–130, 2002.

[11] J. Y. Lee, J. H. Lee, H. Kim, and H. S. Kim. A cellular topology-based approach to generating progressive solid models from feature-centric models. *Computer-Aided Design.*, 36(3):217–229., 2004.

[12] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel based object simplification. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 296, 1995.

[13] C. Andujar, P. Brunet, and D. Ayala. Topology-reducing surface simplification using a discrete solid representation. *ACM Trans. Graph.*, 21(2):88–105, 2002.

[14] Voxmap pointshell. http://www.boeing.com/phantom.

[15] J. J. Shah and M. Mantyla. *Parametric and feature based CAD/CAM: Concepts, techniques, and applications.* John Wiley & Sons, Inc., New York, NY, USA, 1995.

[16] K. Y. Lee, M. A. Price, and C. G. Armstrong. CAD-TO-CAE integration through automated model simplification and adaptive modeling. In *Proceedings of International Conference on Adaptive Modeling and Simulation*, Barcelona. Spain., 2003.

[17] S. H. Lee. A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques. *Comput. Aided Des.*, 37:941–955, August 2005.

[18] R. J. Donaghy, W. McCune, S. J. Bridgett, C. G. Armstrong, and D. J. Robinson. Dimensional reduction of analysis models. In *Proceeding of 5th International Meshing Roundtable*, Pittsburgh. PA. USA., 1996.

[19] J. Donaghy, C. G. Armstrong, and M. A. Price. Dimensional reduction of surface models for analysis. *Engineering with Computers.*, 16(1):24–35, 2000.

[20] M. Rezayat. Midsurface abstraction from 3D solid models: general theory and applications. *Computer-Aided Design*, 28:905–915, 1996.

[21] M. Ramanathan and B. Gurumoorthy. Interior Medial Axis Transform computation of 3D objects bound by free-form surfaces. *Computer-Aided Design*, 42(12):1217–1231, Dec. 2010.

[22] D. Sheen, T. Son, D. K. Myung, C. Ryu, S. H. Lee, K. Lee, and T. J. Yeo. Transformation of a thin-walled solid model into a surface model via solid deflation. *Computer-Aided Design*, 42(8, SI):720–730, Aug. 2010.

[23] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics Applications*, 21(3):24–35, 2001.

[24] D. P. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney. *Level of Detail for 3D Graphics.* Elsevier Science Inc., New York, NY, USA, 2002.

[25] J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):133–144, 1998.

[26] T. Tan, K. Chong, and K. Low. Computing bounding volume hierarchies using model simplification. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 63–69, 1999.

[27] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[28] G. V. Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[29] I. Palmer and R. Grimsdale. Collision detection for animation using sphere trees. *Computer Graphics Forum*, 14(2):105–116, 1995.

[30] J. T. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[31] F. Luo, E. Zhong, J. Cheng, and Y. Huang. VGIS-COLLIDE: an effective collision detection algorithm for multiple objects in virtual geographic information system. *International Journal of Digital Earth*, 4(1):65–77, 2011.

[32] D. Jung and K. Gupta. Octree-based hierarchical distance maps for collision detection. *Journal of Robotic Systems*, 14:789–806, 1997.

[33] S. Krishnan, M. Gopi, M. Lin, D. Manocha, and A. Pattekar. TI: Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum*, 17:315–326, 1998.

[34] R. Weller and G. Zachmann. Inner sphere trees for proximity and penetration queries. In *2009 Robotics: Science and Systems Conference (RSS)*, Seattle, WA, USA, June 2009.

[35] R. Weller and G. Zachmann. Inner sphere trees and their application to collision detection. In Guido Brunnett, Sabine Coquillart, and Greg Welch, editors, *Virtual Realities*, pages 181–201. Springer Vienna, 2011.

[36] D. S. Coming and O. G. Staadt. Velocity-aligned discrete oriented polytopes for dynamic collision detection. *IEEE Transactions on Visualization and Computer Graphics*, 14:1–12, 2008.

[37] C. Ericson. *Real-time collision detection.* The Morgan Kaufmann Series in Interactive 3D Technology., 2004.

[38] A. Vogiannou, K. Moustakas, D. Tzovaras, and M. G. Strintzis. Enhancing bounding volumes using support plane mappings for collision detection. *Computer Graphics Forum*, 29(5):1595–1604, 2010.

[39] F. Liu, T. Harada, Y. Lee, and Y. J. Kim. Real-time collision culling of a million bodies on graphics processing units. *ACM Trans. Graph.*, 29:154:1–154:8, December 2010.

[40] G. Vanecek. Back-face culling applied to collision detection of polyhedra. *Journal of Visualization and Computer Animation*, 5(1):55–63, 1994.

[41] S. Kumar, D. Manocha, W. Garrett, and M. Lin. Hierarchical back-face computation. *Computers & Graphics*, 23:681–692, 1999.

[42] S. Redon, A. Kheddar, and S. Coquillart. Hierarchical back-face culling for collision detection. *Intelligent Robots and System.*, 3(3):3036–3041, 2002.

[43] M. Tang, S. E. Yoon, and D. Manocha. Adjacency-based culling for continuous collision detection. *The Visual Computer: International Journal of Computer Graphics*, 24(7):545–553, 2008.

[44] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[45] P. Hachenberger and L. Kettner. Boolean operations on 3D selective Nef complexes: optimized implementation and experiments. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 163–174, New York, NY, USA, 2005. ACM.

[46] A. Thakur and S. K. Gupta. Real-time dynamics simulation of unmanned sea surface vehicle for virtual environments. *Journal of Computing and Information Science in Engineering*, 11(3):031005, 2011.

[47] M. Schwartz, P. Svec, A. Thakur, and S. K. Gupta. Evaluation of automatically generated reactive planning logic for unmanned surface vehicles. In *Performance Metrics for Intelligent Systems Workshop*, September, 2009.

[48] OpenDynamics rigid body dynamics engine., November 2008. http://www.ode.org/.