ABSTRACT

Title of dissertation: PHYSICS-AWARE MODEL SIMPLIFICATION FOR INTERACTIVE VIRTUAL ENVIRONMENTS

Atul Thakur, Doctor of Philosophy, 2011

Dissertation directed by: Professor Satyandra K. Gupta
Department of Mechanical Engineering

Rigid body simulation is an integral part of Virtual Environments (VE) for autonomous planning, training, and design tasks. The underlying physics-based simulation of VE must be accurate and computationally fast enough for the intended application, which unfortunately are conflicting requirements. Two ways to perform fast and high fidelity physics-based simulation are: (1) model simplification, and (2) parallel computation. Model simplification can be used to allow simulation at an interactive rate while introducing an acceptable level of error. Currently, manual model simplification is the most common way of performing simulation speedup but it is time consuming. Hence, in order to reduce the development time of VEs, automated model simplification is needed. The dissertation presents an automated model simplification approach based on geometric reasoning, spatial decomposition, and temporal coherence. Geometric reasoning is used to develop an accessibility based algorithm for removing portions of geometric models that do not play any role in rigid body to rigid body interaction simulation. Removing such inaccessible portions of the interacting rigid body models has no influence on the simulation accuracy but reduces computation time significantly. Spatial decomposition is used to develop a clustering algorithm that reduces the number of fluid pressure computations resulting in significant speedup of rigid body and fluid interaction simulation. Temporal coherence algorithm reuses the computed force values from rigid body to fluid interaction based on the coherence of fluid surrounding the rigid body. The simulations are further sped up by performing computing on graphics processing unit (GPU). The dissertation also presents the issues pertaining to the development of parallel algorithms for rigid body simulations both on multi-core processors and GPU. The developed algorithms have enabled real-time, high fidelity, six degrees of freedom, and time domain simulation of unmanned sea surface vehicles (USSV) and can be used for autonomous motion planning, tele-operation, and learning from demonstration applications.

Physics Preserving Model Simplification for Interactive Virtual Environments

by

Atul Thakur

Dissertation submitted to the Faculty of the Graduate School of the University of Maryland, College Park in partial fulfillment of the requirements for the degree of Doctor of Philosophy
2011

Advisory Committee:
Professor Satyandra K. Gupta, Chair/Advisor
Professor Shapour Azarm
Professor Hugh Bruck
Assistant Professor Nikhil Chopra
Professor Amitabh Varshney (Dean's representative)

Dedication

To my parents, my elder brother, and all my teachers.

Acknowledgments

It is hard to overstate my gratitude to my Ph.D. advisor Prof. Satyandra K. Gupta for giving me an opportunity to work with him and mentoring me as a researcher. It is difficult to be unaffected by his unfailing energy and enthusiasm which has always been a source of energy for me to deal with difficult and seemingly unyielding research problems. I am grateful to him for providing me with a research environment like a free intellectual playground to try out different ideas, nevertheless, all the guidance and support when I got lost after too much of play. His academic advise model (so to speak) helped me learn many skills and expanded my horizons in addition to my key dissertation requirements. My vocabulary is not enough to express my gratitude to Prof. Gupta for the innumerable subtle things that he taught me both explicitly and implicitly which helped me become a better researcher.

I would also like to thank Prof. Amitabh Varshney, Prof. Hugh Bruck, Dr. Nikhil Chopra, and Prof. Shapour Azarm for accepting to serve in my dissertation committee. Their valuable suggestions related to my dissertation helped me a lot during my Ph.D. I thank them for being very prompt and responsive about my dissertation needs despite of having unbelievably busy academic and research commitments of their own. I would then like to thank National Science Foundation (NSF) for funding (through my advisor) my research.

Then, I would like to thank Dr. Petr Svec, for being a very good friend and a great colleague. Petr's sharp analytical thinking, attention to details, and

persistence in research pursuit has always been a source of inspiration for me and will hopefully remain with me for the rest of my life. I am indebted to Petr for introducing me to the research area of robot motion planning and motivating me to apply model simplification techniques that I developed as part of my dissertation, which led to two research papers in reputed conferences. I would also like to thank him for meticulously proofreading some parts of my dissertation.

My deepest gratitude goes to my family without whose support none of this could have been possible. I would like to thank my mother for her boundless love, patience and support. Values taught by my father and his wisdom helped me in making right choices in all aspects of my life including research. I would like to thank my brother Dr. Ajay Thakur who kindled an interest in me for academics and research and helped me maintain enthusiasm levels all through. I would like to thank my brother Avinash Thakur, my sister-in-law Kiran Thakur and niece Aastha for their ever smiling faces and voices which never failed to energize me.

I would like to thank my former teachers Prof. S. S. Pande, Prof. P. G. Awate, and Prof. Amitava De from Mechanical Engineering Department, Indian Institute of Technology Bombay for encouraging me to embark on the journey of Ph.D. I am especially grateful to Prof. Pande for providing constant support and encouragement during difficulties. I would also like to thank Prof. P. Prabhu from University of Mumbai who taught me what it means to be a Mechanical Engineer and developed my interest in manufacturing engineering in particular.

My 8316 friends (Dr. Priya Ranjan, Dr. Krishna V. Kaipa, Dr. Saket Navlakha, and Pavan Gajendragad) in US made my transition to this new country

and new culture smooth and I am very thankful to them. I am highly indebted to my friends Anupam Anand, Dr. Sabyasachee Mishra, and Pavan Tallapragada for standing by me and being source of encouragement. Their smile and company made the time fly. I thank my old time friend Sushrut Pavanaskar for the insightful discussions about multiple things including life, research, etc. I am thankful to my friends Vaibhav Arghode, Praveen Noojipady, and Kedar Dimble for their support, encouragement and often needed occasional food invitations. I would also like to thank my friends in India, Amit, Milind, and Pradnya for their support and encouragement.

Then, I would like to thank my colleagues Adam Montjoy, Alexander Weissman, Brian H. Russ, James Koo, Dr. Krishnanand Kaipa, Dr. Madan Dabbeeru, Sagar Chowdhury, and Timothy R. Hall in Simulation-Based System Design Laboratory (SBSD) for their support, encouragement, and for creating an amicable work atmosphere in the laboratory. I would like to thank my past colleagues from SBSD, Dr. Arvind Ananthanarayanan, Dr. Ashis Banerjee, Dr. Peyman Karimian, and Dr. Wojciech Bejgerowski for being great friends.

Last but not the least I thank the Institute for Systems Research staff (Jason Strahan and Beverly Dennis) and Mechanical Engineering administrative staff (Lita Brown, Penny Komsat, Lee Ellen Harper, Amarildo DaMata, and Janet M. Woolery) who were very helpful and prompt in helping me with administrative issues and thereby making my stay at the University of Maryland enjoyable and smooth.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

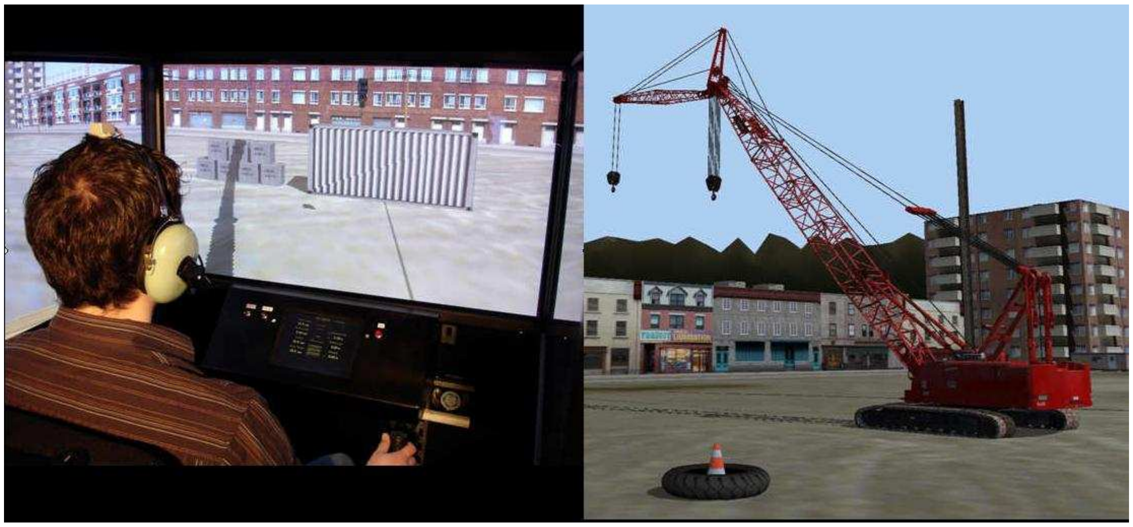
AABB	Axis-aligned Bounding Box
B-rep	Boundary Representation
CAD	Computer-aided Design
CAE	Computer-aided Engineering
CSG	Constructive Solid Geometry
DFT	Discrete Fourier Transform
DOF	Degrees of Freedom
EOD	Explosive Ordnance Disposal
FEM	Finite-element Method
GPGPU	General Purpose computing on Graphics Processing Unit
LOA	Level of Abstraction
LOD	Level of Detail
LPF	Low Pass Filter
MAT	Medial Axis Transform
MDP	Markov Decision Process
NMT	Non-manifold Topology
NURBS	Non-uniform Rational B-spline
OBB	Oriented Bounding Box
ODE	Ordinary Differential Equation
PBD	Programming by Demonstration
PES	Partial Entity Structure
PID	Proportional Integral Derivative
POMDP	Partially-ordered Markov Decision Process
PSM	Progressive Solid Model
PSS	Part Section Signature
RANS	Reynolds-averaged Navier Stokes
STL	Stereolithography
USSV	Unmanned Sea Surface Vehicle
VE	Virtual Environment
VR	Virtual Reality

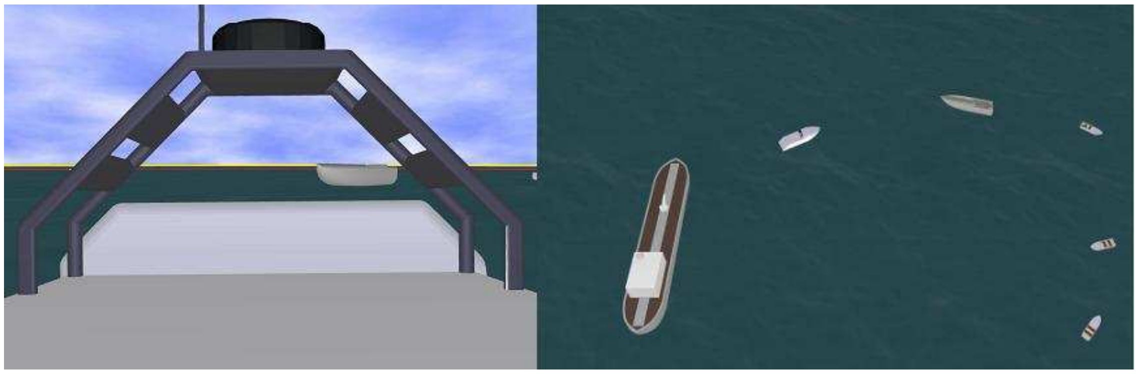
Chapter 1

Introduction

1.1 Background

The role of virtual environments (VE) is crucial in efficient design and operation of robotics systems. VEs are extensively used in operator training for teleoperation [PPB10]. Another application of VEs in autonomous robotics is planning using programming by demonstration (PBD) also known as imitation learning, and hardware and software design [ACVB09, SK08]. VEs are also used in non-robotics applications such as operator training for hazardous operations [YJJ+10], virtual surgery [GNU+09], etc. In all of the applications, VE is supposed to meet two main requirements for immersive realism namely, the accuracy of the results and real-time interactivity. VE for complex systems, such as robots interacting with diverse environments or humans interacting with environments using some virtual reality (VR) tools, require high fidelity physics-based simulations (*e.g.*, rigid body dynamics simulation, computational fluid dynamics, finite element simulation, etc.). Also, in order to be interactive, the VE requires real-time performance of the underlying physics simulation. Physics-based simulation enables the VE to render the realistic motion, forces, visualization and other sensory feedback into various interaction channels for enhanced immersivity [HM07, MH08]. Few examples of VEs are shown in Figure 1.1.


(a) *Vortex crane simulator [vor10].*


(b) *USSV VE simulator [SSTG09].*

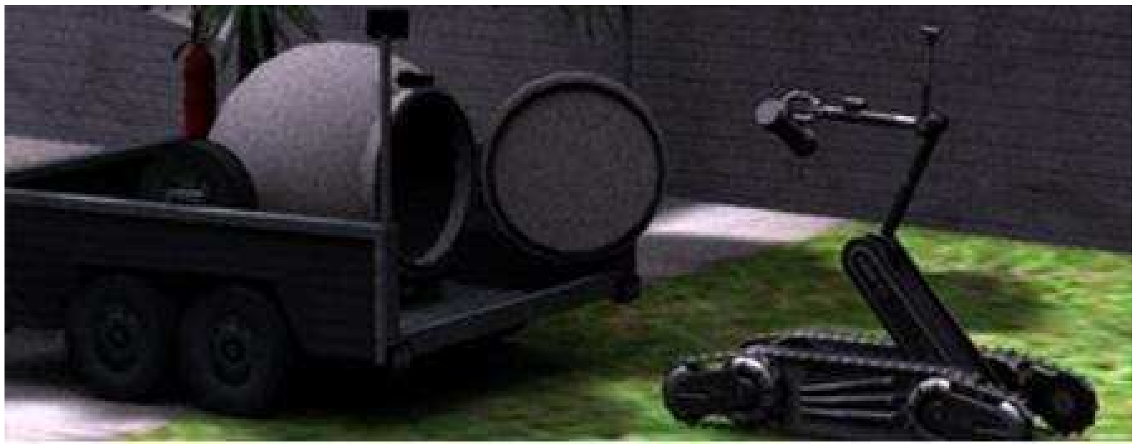
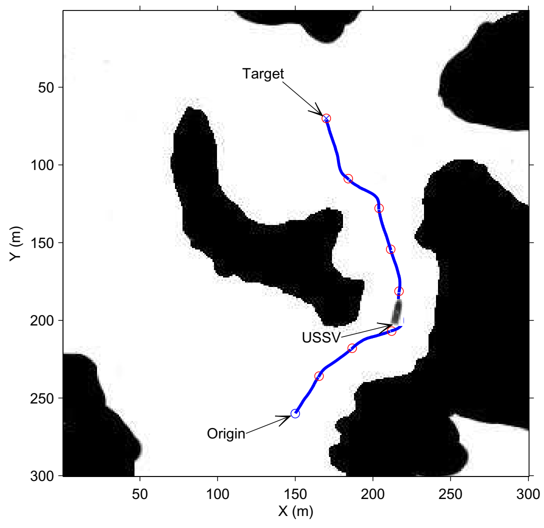

(c) *Battelle's VE simulator for explosive ordnance disposal (EOD) application [bat10].*

Figure 1.1: *Examples of VE based simulators.*

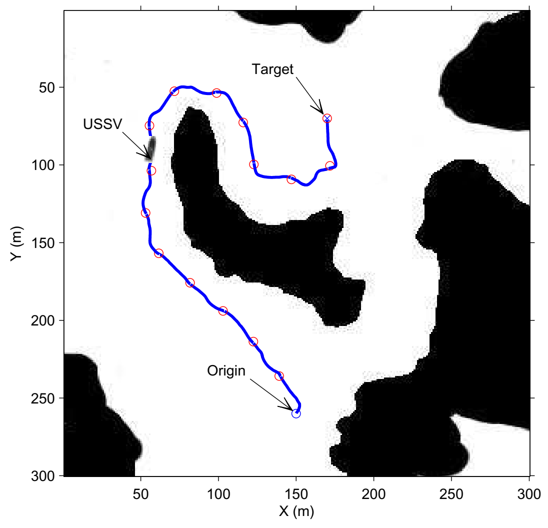
In Figure 1.1(a), VE is used for simulating crane and hoist training system. In order to be realistic training environment, the interaction forces on the crane and hoists must be estimated with sufficient accuracy for the human interacting with the VE. In this system, Vortex rigid body dynamics simulation engine is used for computing the forces that are further displayed on the haptic device to give realistic tactile feedback to the user of the VE. Figure 1.1(b) shows a VE for simulating 6 degrees of freedom (DOF) USSV motion. The USSV simulation environment is used for operator training and PBD. Accurate computation of the forces acting on the USSV is very important for the realistic behavior of the simulated vehicle. Figure 1.1(c) shows an explosive ordnance disposal (EOD) simulation system used for training hazardous EOD tasks. Again the force computation is required for the realistic feedback to the user of the system.

In addition to the VE simulators (see Figure 1.1) used directly by human users for training and imitation learning [ACVB09], they can even be used for autonomous robot motion planning [TG11, SSTG11, WBK$^+$10, PKV10, LL06b]. One of the current research issues in robot motion planning is to perform efficient physics-aware trajectory planning. Let us consider the case of USSV trajectory planning in Figure 1.2. In order to generate dynamically feasible and optimal trajectory plans, the USSV simulation must be accurate and computationally fast [LaV06]. For the lower sea-state (see Figure 1.2(a)), the system generated trajectory is shorter but leading through the riskier narrow passage whereas for the higher sea-state (Figure 1.2(b)) the generated trajectory is longer but safer [TG11].

In both types of the described systems (see Figures 1.1 and 1.2), the environ-



(a) *Computed at sea-state 3.*



(b) *Computed at sea-state 4.*

Figure 1.2: *Trajectory plan generated using high fidelity simulation for higher sea-state 4 is much more conservative compared to that for the lower sea-state 3 as there is a higher chance of getting deviated and hitting an obstacle in case of the high sea-state environment than calmer low sea-state environment [TG11].*

ment consists primarily of rigid (such as robot arm, USSV hull, etc.) and compliant (such as cable, etc.) bodies. The most widely used simulation technique used for design, called finite element analysis (FEA) [RG01] is seldom directly applied in VEs. The reason is that although FEA is highly accurate in predicting structural deformations and movements, it is computationally very slow, especially for detailed geometries. In other words, the simulations performed must be fast enough to satisfy the refresh rate requirement for VEs [MH08]. Relatively faster performance of rigid body simulation often makes it a practical choice for VEs. Even compliant parts are simulated using the rigid body simulation by using the concept of pseudo-rigid bodies in which compliant parts are modeled as rigid bodies connected by springs [YHL+05]. In order to use rigid body simulations in VEs effectively, an optimal tradeoff must be established between two conflicting constraints namely, the available computing time and the level of accuracy desired from the simulation. The available computing time depends upon the application in which the rigid body simulation will be used, *e.g.*, in the case of visualization applications the required frame rate is more than 20 Hz whereas in the case of haptic display applications the desired frame rate is more than 1000 Hz. In the case of autonomous planning operations, much faster simulation might be needed based upon whether the system is real-time. If the models used for the simulation are more detailed, the results will be more accurate but with higher computation time. On the other hand, models cannot be simplified arbitrarily to reduce the computation time as that will hamper the accuracy of the simulation in an unpredictable way. This gives rise to an important question of how to simplify the models for rigid body simulations in order to satisfy the computational and accuracy requirement. In order to address this problem, manual model simplification is usually performed in most of the VE's so that the computational time is reduced without jeopardizing the simulation fidelity. Manual simplification is expensive and time consuming. Automatic physics-preserving model simplification for rigid body simulation can significantly reduce the cost of development and enhancement (to cater to the wider range of simulation scenarios) of the VEs.

1.2 Motivation

In Section 1.1, we saw that automatic physics-preserving model simplification is required for the rigid body simulations used in VEs. In order to develop techniques for performing automatic model simplification for rigid body simulations, it is useful to understand two aspects namely, the major computational performance bottleneck operations, and the operations that influence the fidelity or the accuracy of the simulations the most. Often, in a rigid body simulation the major amount of computational time is spent in the following two operations:

(i.) computation of forces, and

(ii.) solution of the equations of the motion.

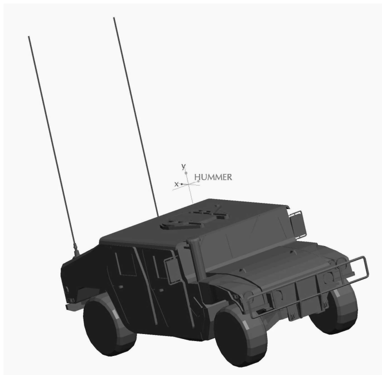
The time taken for the computation of the forces acting on the rigid bodies is considerably high as compared to the time taken solving the equations of motion. The forces acting on any rigid body in the context of rigid body simulation can be classified into two general categories as follows:

(i.) force due to interaction when the rigid bodies come into contact with each other during collision, and

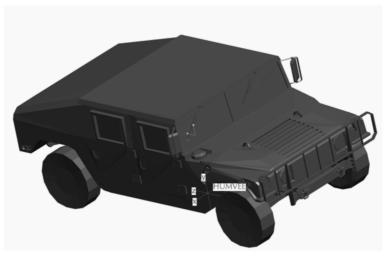
(ii.) force due to the environment and the rigid body interaction such as force due to fluid pressure acting on a rigid body surface moving in a fluid medium.

In order to demonstrate the requirements of the computation time for the case of rigid body to rigid body interaction in a simulation, consider an example of robot motion planning for close contact combat situation on a treacherous terrain of three unmanned vehicles as shown in Figure 1.3. The vehicle and the terrain model can be geometrically very complex. Numerous simulations need to be performed for the motion planning task, which require fast and accurate estimation of reaction and collision forces acting on the unmanned vehicles due to interaction with the terrain and the interaction among the vehicles. The simulation is performed in OpenDynamics Physics Engine [ode08] on a computer with an Intel Quad Core processor. The Hummer model has 7699 facets while the Humvee model has 5381 facets. It is found that on an average (taken over 1000 simulation time steps under close proximity situation), the average computation time taken per simulation time step for contact point determination using in-built collision detection engine inside the OpenDynamics (*dSpaceCollide* function) is about 2.6 ms, whereas the time taken for solving the motion equations (*dWorldStep* function) is about 0.3 ms. Thus, the time taken for contact point determination (and hence contact force determination) is more than the time taken for solving the dynamics motion equations by a factor of about 9. The numbers vary with the size and the complexity of the part models; however, it is evident that the computation time taken for contact point determination is much higher compared to the dynamics equation solution. Another aspect of the simulation is the fidelity which depends directly on the accuracy of the determined contact points (used for computing collision reaction forces).

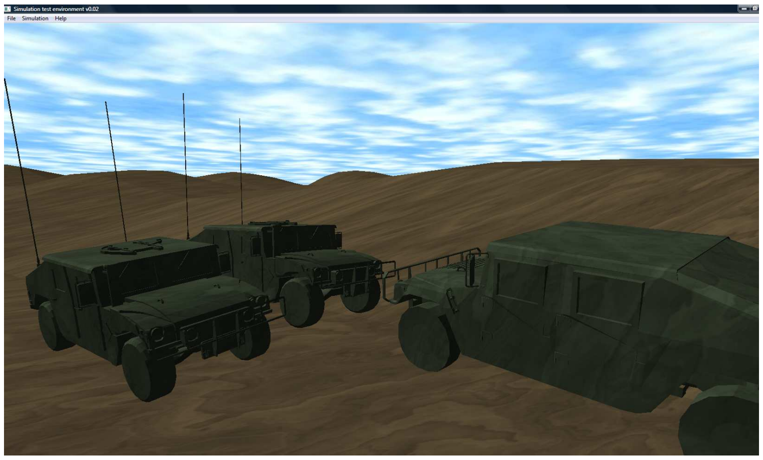
Let us consider another example, where a rigid body interacts with fluid environment, such as a USSV interacting with ocean waves (see Figure 1.4). Based on the simulation technique described by Krishnamurthy *et al.* [KKF05], we developed an ocean wave to boat simulation software and performed a test on a boat model



(a) *Hummer model with 7699 facets.*



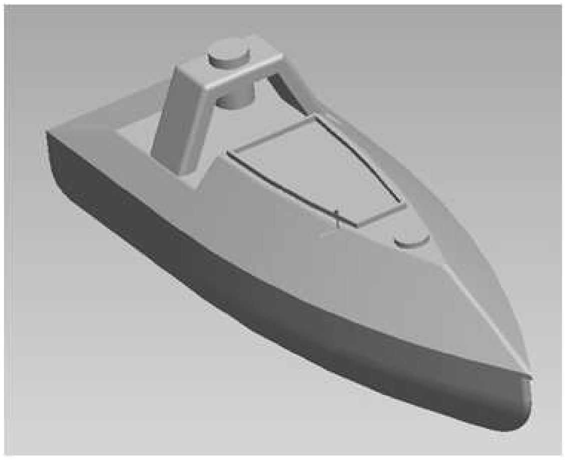
(b) *Humvee model with 5381 facets.*

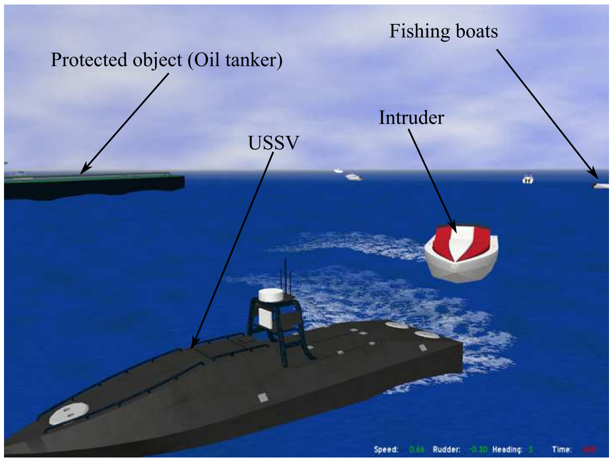


(c) *Rigid body simulation using OpenDynamics simulation library.*

Figure 1.3: *Simulation of close proximity combat situation using OpenDynamics simulation library (average time spent on contact point determination is* 2.6 *ms whereas average time spent on solving motion equation is 0.3 ms).*

with 1793 facets as shown in Figure 1.4(a) [TG10]. The results of the tests showed that the time taken for the computation of the interaction (between the ocean wave and the boat) force per simulation time step is about 995 ms and dynamics equation solution is about 4 ms. Again we see that the computation time for force computation is significantly higher compared to the time taken for solving the dynamics


(a) *Boat model with 1793 facets.*



(b) *USSV simulation environment [TG10].*

Figure 1.4: *USSV Simulation environment and computational performance (computational time required for determining force due to ocean waves is* 995 *ms and dynamics equation solution is about* 4 *ms).*

equations. Also, the fidelity of the simulation is directly influenced by the accuracy of the estimated forces as the forces are integrated to get the position and the velocity of the boats in each simulation time step.

It thus follows that, in order to reduce the rigid body simulation time with known effect on simulation fidelity, it is very important to reduce the time taken for the computation of forces and analyze the influence of simplification on the accuracy of forces obtained by the simplified model. One of the main reasons behind the significantly high amount of time taken for the computation of forces in a rigid body simulation is the dependence of the force computation process on the complexity of the geometry of the parts involved in the computations. Nowadays, highly detailed geometric models are available for simulations due to the advances in shape digitization and data acquisition techniques [Dig11]. Using a faster computer can speedup the computations to some extent, however, the model complexity can always be increased which might require much faster computer. This means using faster computers alone may not be enough or optimal to address the issue of achieving faster simulation speed. One way to address this issue is to simplify the geometry of the parts involved in the simulation in such a way that the simplified models behave similar to the original models in the simulation environment, thereby improving computational performance. A large body of literature exists on the level of detail (LOD) based automatic geometric model simplification for graphics rendering [Lue01, LWCR02]. Unfortunately, the LOD based techniques for graphics rendering cannot be directly applied to the physics-based simulations because the effects of those simplifications on the accuracy of the results of the simulation are unknown. There is a lot of work done in the area of model simplification for the finite element analysis (FEA) [TBG09]. As discussed in Section 1.1, model simplification techniques for FEA cannot be used directly, as FEA is not used in VEs for the requirement of faster simulation. Thus, model simplification techniques for rigid body simulations are needed to improve the refresh rates obtainable in VEs. Another way to improve the computational performance is to use parallelization using multi-core and GPU based hardware for those parts of the simulation program that are highly data parallel. Even in that case, overall simulation speedup will be tremendously improved by using model simplification techniques in conjunction with computing parallelization.

1.3 Goal and Scope

This work identifies the problem of automatic model simplification for rigid body simulations. Development of model simplification algorithms and the techniques for analyzing the relationship between the accuracy of the rigid body simulation results and the computation time with the simplified models forms the central theme of the dissertation. The main research issues this work addresses are as follows:

(i.) *Development of automatic model simplification technique for interaction among rigid bodies.* Most of the model simplification for rigid body simulation performed manually uses the strategy of simplifying parts with respect to each other such that the resulting simplified geometries preserve the contact forces acting on them due to their interaction. The accuracy of the contact force estimation in rigid body simulation is dictated by the accuracy of the estimated contact points caused by the interaction of rigid bodies. In order to compute the contact points between the geometries representing the rigid bodies, collision detection needs to be performed in each simulation time step, which is a computationally expensive operation. There has been a lot of research done in the area of efficient collision detection [KHI+07, JTT01, LG98]. Most of the collision detection algorithms are based on hierarchical data-structures such as oriented bounding box (OBB), axis-aligned bounding box (AABB), etc. Hierarchical data-structure works best when the part models are at larger distances, as the deeper tree nodes are pruned. But when the parts come in close proximity, as it happens frequently in a rigid body simulation scenario, much deeper nodes of the hierarchical data-structure are explored by the collision detection engine leading to slower performance. The regions on the geometric model of a rigid body that are not accessible by the other rigid body because of the geometrical constraints (such as in an unmanned vehicle, the engine, steering mechanism, and other mechanisms which are obscured by the vehicle frame are inaccessible by the terrain and the other vehicles) needs to be removed. This kind of simplification will significantly prune the hierarchical tree data-structure leading to significant speedup in worst case collision computation time. Technique for automating and optimizing the process of simplification, such that the part model can be simplified before the simulation (*i.e.*, off-line) is developed in this dissertation [TG09]. The simplification framework should be such that the forces due to the interaction between the parts are not altered when the simplified geometries are used for the simulation as compared to when the unsimplified geometries are used.

(ii.) *Development of model simplification technique for fluid to rigid body interaction scenarios in rigid body simulation.* The environment influences the motion of rigid bodies in the form of temporally and spatially varying vector fields. Unlike the rigid body interaction case, for a spatio-temporally varying vector field to rigid body interaction scenario, it is almost impossible to come up with a simplified geometry which exactly preserves the computed force. However, part models can be simplified in such a way that the error in computed force with the simplified model is small and predictable. The dissertation will present model simplification techniques to simulate the effect of ocean waves on motion of the boats represented as rigid bodies in six degrees of freedom, in real-time [TG10]. For the convenience of analysis, we will assume that the ocean is completely deterministic.

(iii.) *Model simplification for motion planning in nondeterministic environment.* In practice, environments may be dynamic and nondeterministic, *e.g.* ocean represented using superposition of wave components [TG11, KKF05, TG10, FS04b], where each wave component is specified by wave amplitude, frequency, direction, and uniformly random phase lag depending upon the sea-state of the ocean. The ocean waves once initialized with a given set of wave components evolves deterministically. However, a sea-state is represented by randomly varying similar (but not identical) ocean components. This means that the USSV simulation needs to be performed many times in Monte-Carlo fashion to statistically estimate the USSV motion in a given sea-state. In order to simulate many instances of the simulation simultaneously, GPU computing can be used. Development of model simplification techniques suitable for such simulation is is addressed in this dissertation. The sampled influence of the ocean on the USSV motion is used for generating state transition probability in MDP based framework. The application of the developed model simplification approach for generating state transition probability is demonstrated on USSV trajectory planning problem in nondeterministic ocean [TG11].

Chapter 2

Literature Review

As pointed out in Chapter 1, significant simulation speedup is needed many applications. There are two main lines of research to reduce computational burden of simulations, namely, meta-modeling and physics-aware simplification approaches. Meta-modeling is a general process of abstracting a phenomenon creating model of a model as defined by Blanning and Kleijnen [Bla74, Kle75]. In general, the phenomenon can be a real experiment or a computer simulation. The main advantage of meta-model resides in the fact that a limited number of experimental or simulation runs can be used to construct a meta-model. Intermediate responses can then be estimated very quickly by using the constructed meta-model. Meta-modeling, today is a mature research area with many advances made in last few years and several surveys have been published in [Bar94, SPKA01, BM06, WS07, SW10a, ZX10]. Few of the common meta-modeling techniques used nowadays are splines [TC09], Kriging [Kle08], polynomials or response surfaces [RC10], radial basis functions [SW10b], neural networks [Yan10], proper orthogonal decomposition (POD) [AAP11], etc. Meta-models are function approximation or surrogate models constructed using experimental or simulation results. The main advantage of meta-models is that they can represent expensive simulations with computationally inexpensive approximation functions. Meta-models are commonly used in problems related to multidisciplinary and multiobjective optimization [LLA08, LAFMD06]. However, meta-modeling approaches start exhibiting significant computational challenges when the number of parameters in simulations is very large and simulations are based on highly nonlinear physics.

The current dissertation deals with physics-aware model simplification for rigid body simulations in VEs. It was outlined in Chapter 1 that commonly occurring rigid body simulation scenarios in the areas of robotics and VR belongs to three categories, namely, (1) interaction among rigid bodies, (2) interaction between rigid bodies and deterministic fluid flow, and (3) interaction between rigid bodies and nondeterministic fluid flow. In order to understand the issues related to physics-aware model simplification for the above mentioned rigid body simulation scenarios, literature related to following three areas are reviewed in this chapter.

(i.) CAD Model simplification techniques for physics based simulations (discussed in Section 2.1).

(ii.) Fluid-rigid body interaction simulation (discussed in Section 2.2).

(iii.) Applications of dynamics simulation in robot motion planning (discussed in Section 2.3).

2.1 CAD model Simplification Techniques for Physics based Simulations

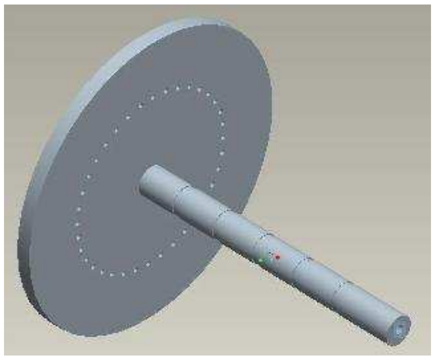
In this section[1] we present a review of existing techniques for physics-based model simplification. Physics-based simulations play an important role during the product realization process. Let us consider few representative examples. Multi-body dynamics simulations are used to determine the sizes of actuators during the design of robots. Finite element simulations are used in structural and thermal analysis of components in the automotive and aerospace industries. Computational fluid dynamics simulation is used in automotive engine cooling system design. These simulations help in reducing the need for expensive physical prototyping and hence shorten the product development time and reduce the product development cost. Apart from these design examples, physics-based simulation is also used in assembly planning, ergonomics analysis, and testing applications. Physics-based simulations are primarily driven by 3D CAD data. The computational performance of simulations depends on the number and complexity of the geometric features present in the CAD model. Features are an integral part of modern CAD model and they are used in virtually all the domains of product life cycle, namely design, manufacturing, analysis and maintenance. Even the presence of a single, relatively small geometric feature can increase the size of the underlying discrete physical simulation problem by as much as 10-fold [WSO03, LAPL05].

[1] The contents of this section was published in the journal Computer-Aided Design in 2009 [TBG09].

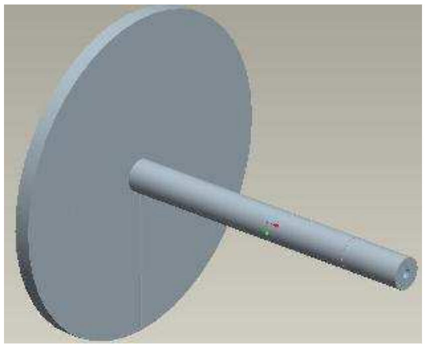
Extremely large computational times limit the usefulness of simulations during the design cycles. Complex models may often lead to ill-conditioned matrices and hence working with non-simplified complex models may produce inaccurate results [Saa03]. Hence, simply utilizing more powerful computers will not solve the problem associated with highly complex models. In order to get accurate results in a timely manner, one must utilize simplified models that retain the important details and eliminate the irrelevant ones.

To illustrate the above mentioned point, let us consider a simple example of a part (see Figure 2.1) subjected to different kinds of simplifications. Figure 2.1(a) shows the solid model of an axis-symmetric part with several grooves and holes. Figure 2.1(b) shows the simplified part model with the tiny grooves and holes removed, which can be used for an application like rigid body simulation where small holes and grooves play a negligible role in determining the inertia tensor and the collision contact points. Figure 2.1(c) shows a simplified 2D model exploiting the symmetry of the part, which can be used for an application such as thermal analysis. Figure 2.1(d) shows the simplified part composed of a beam and a plate element which can be used in structural analysis. All these simplification instances reduce the computational time significantly while affecting the respective simulation results negligibly as compared to the full blown solid model. Currently, such kinds of feature simplifications and idealizations are mostly performed manually. Manual feature simplification, however, requires human expertise and is time-consuming.

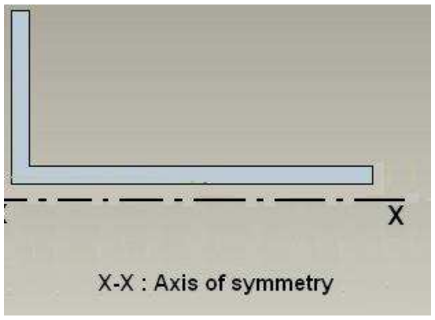
Several efforts have been made over the last few years to automate the model simplification process. It is studied in various contexts such as finite element anal-



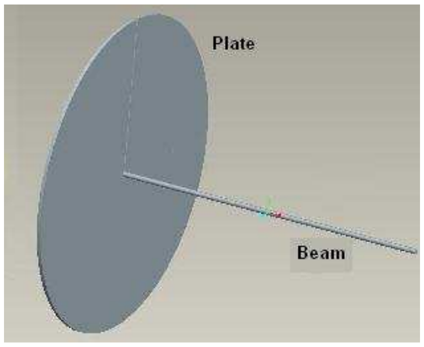
(a) *3D model.*



(b) *Simplified model after removing notches and tiny holes.*



(c) *Simplified model after dimension reduction and exploiting symmetry.*



(d) *Simplified model after dimensional reduction to combination of beam and plate elements.*

Figure 2.1: *Model simplification based on target simulation application leads to different simplified models by removing irrelevant details from the point of view of the simulation application.*

ysis and collision detection within the overall category of physics-based simulation [Arm94]. In the collision detection field, some of the reported work relates to dynamic simplification [YSLM04] and construction of bounding volume hierarchies [TCL99]. These primarily relate to multi-resolution representation for performing collision detection at various levels and do not address the problem of model simplification explicitly and, thus, will not be covered in this chapter for the sake of brevity. Polygonal mesh simplification has been extensively studied by the graphics community [Lue01, LWCR02, CMS98]. We will not focus on the model simplification methods for graphic rendering [ESV98] in this chapter. We have, however, covered some techniques that address model simplification in the context of network model transmission from rendering and occlusion analysis perspective and can also be applied to collision detection [ABA02, HHK$^+$95]. Techniques developed for different contexts have different simplification objectives and hence different simplification outcomes. The purpose of this chapter is to compile a list of existing techniques that are relevant for physics-based simulation problems and to characterize them based on their attributes. The remainder of the chapter is organized in the following manner. In Section 2.1.1, the basic terminology used in model simplification area is explained in details. Sections 2.1.2 through 2.1.13 describe the model simplification techniques. It is difficult to discuss all the reported papers in details and thus for the sake of brevity, we have described only few representative techniques in each category in detail. We have included references to the remaining techniques in each category. Section 2.1.16 discusses a systematic taxonomy of the covered techniques and presents several criteria to aid the readers for selecting a model simplification technique suiting their requirements. Finally, Section 2.1.17 summarizes this chapter by highlighting areas of open research.

2.1.1 Terminology

Many different schemes are used to represent 3D geometric information. The first scheme stores the boundary information for a solid (*i.e.*, vertices, edges and faces together with the connectivity information) and is popularly known as the boundary representation (B-Rep). Another scheme stores the history of applying Boolean operations on a solid and is called a constructive solid geometry (CSG) representation [Lee99, SM95, Hof89]. The third scheme stores solids as an aggregate of simple solids such as cubes (voxels). Solid models described in this manner are known as decomposition models. The fourth scheme explicitly stores feature information in addition to the information about the elementary shape entities (vertices, edges, faces etc.) and is referred to as feature based modeling. Features can be classified into two main types - primary, volumetric features such as holes, pockets, slots etc., and secondary, surface features like blends, fillets and rounds. Secondary features are usually introduced in industrial parts to smoothen the sharp edges of the part to enhance strength, aesthetic appeal, handling safety and ease of manufacturability. The model simplification operators operate on the model representation and perform simplification. Based on the type of simplification operators, the techniques for model simplification are categorized into surface entity based, volumetric entity based, explicit feature based and dimension reduction based type in this chapter. In case of surface entity based techniques, features are simplified by operating on surface entities such as faces, edges and vertices. Under this category, reported techniques are edge collapse technique [LLP02], face clustering [SBB97, She01] and low pass filtering approach such as Fourier Transforms [LL98]. In case of volumetric entity based techniques, the features are simplified by operating on volumetric entities from the model. The major techniques reported under this category are effective volume [LAPL05, CKL02], cellular representation [LLKK04, LMC+04], and voxel [ABA02, HHK+95] based approaches. In the explicit feature representation based techniques, the feature information and semantics is explicitly extracted from CAD model or is present in the model in addition to geometric and 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 the sought features). The feature information is used and operated upon unlike implicit model representation schemes where only geometric and topological data is directly used for model simplification. Feature based modeling and feature recognition techniques are described in details in [SM95]. The reported methods are explained in Section 2.1.9. The dimension reduction scheme deals with expressing models by idealizing the features into reduced dimensional shapes [LPA03, DMB+96, DAP00, Rez96]. The reduced dimensional shapes are application dependent. For finite element analysis, the model features are idealized into one-dimensional beams, point masses, etc. Thus, the dimensional reduction operators convert a model into a reduced dimension model or mixed dimension model based on the level of abstraction desired [RAM+06]. In a mixed dimension model, the model is constituted of higher as well

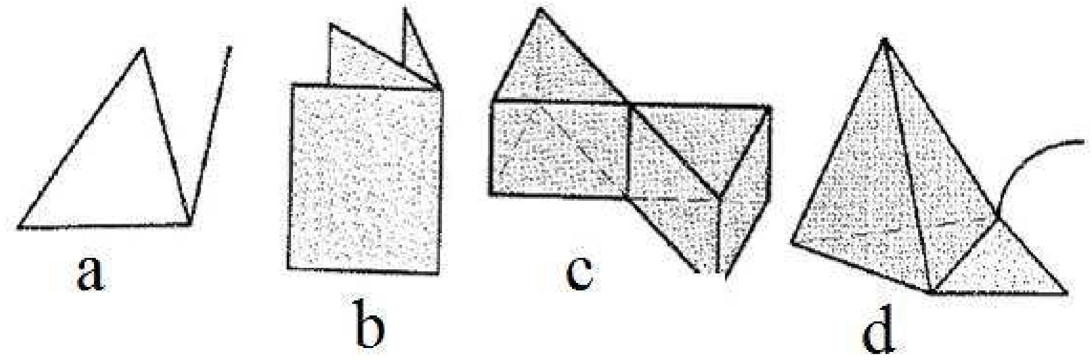


Figure 2.2: *Some examples of non-manifold cell complexes [Mas93].*

as lower dimension features. The remainder of this section introduces underlying foundations for describing CAD model simplifications approaches.

Non-manifold Topology (NMT): NMT is a model representation scheme introduced to address the problems such as absence of multiple representations for concurrent engineering and expensive Boolean operations faced by conventional model representation schemes. It is a representation of wireframe, surface and solid models in a single architecture [Mas93]. Conventional CAD data structures (*e.g.*, B-Rep) can represent only 2-manifold objects. For 2-manifold objects, neighborhood of any point on its boundary is homeomorphic to an open disc in (two dimensional Euclidean space). An object for which the above condition is not satisfied is called non-2-manifold topology or simply NMT [CGP93]. NMT is useful to represent mixed dimensional models (obtained after dimension reduction as explained in Section 2.1.13), stand-alone faces, wireframe edges etc. which cannot be represented using conventional CAD data-structures. Several data structures have been reported that implement NMT, out of which cellular complex and partial entity structure are important from model simplification point of view. Figure 2.2 shows some non-manifold cell complexes.

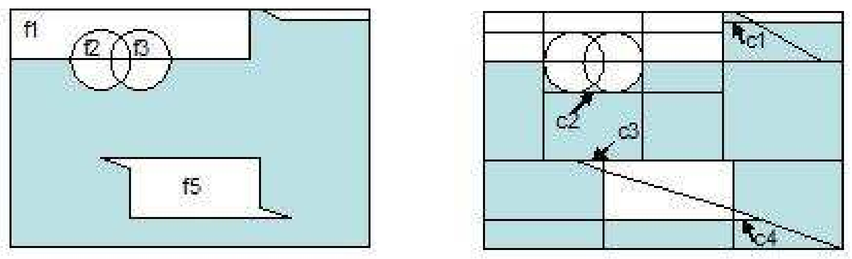
Cellular Representation: Cellular representations have been used for identifying model simplification opportunities [BT93, BdB98, BMNB05, GBT93]. Cellular representations capture both positive and negative spatial regions, and may be viewed as curved voxelized spaces where the exact boundary representation is embedded in the cell boundaries. They are non-manifold geometric representations of the feature model of a product and consist of cell complexes [CCM97]. A cell complex can represent geometric shape relevant to engineering domain effectively, as it includes wireframe, surface and solid models or a combination of them. Mathematically, a cell complex (C) is defined as a set of n-cells, where an n-cells is a bounded subset of $3 - D$ Euclidean space that is homeomorphic to an n-dimensional open sphere. It satisfies the following properties [Mas93]:

(i.) 3D cell complex can be represented by a collection of 0-cell (vertex), 1-cell (edge), 2-cell (face) and 3-cell (volume).

(ii.) The whole boundary of each element consists of lower dimensional elements.

(iii.) No two topological elements intersect each other.

Thus, a cell complex represents parts as a connected set of volumetric disjoint cells of arbitrary shape, and represents each feature as a connected subset of these cells [BdB98]. The cells defined above represent the part shape exactly. In existing approaches for converting features to cellular representations, the subdivision is determined by the property that two cells may never overlap volumetrically. So,



(a) *A part created by removing five subtractive feature from a block.* **(b)** *Cellular model consisting of 35 cells created by feature volume splitting and controlled half space partitioning.*

Figure 2.3: *An example of cellular decomposition .*

whenever two features overlap, their cells are subdivided such that one or more cells are shared by the two features and the remaining cells belong to either one of them. In a cellular representation, a part is represented as a cellular decomposition of the space, where each cell in the space is either void or filled. Filled cells represent the material that belongs to the part. The void cells represent the surrounding space. An example of a cellular decomposition is shown in Figure 2.1.

The transformation operations on cells are represented using the concept of cell maps [LAPL05, RS98]. In one of the earlier approaches, Bidarra *et al.* discussed application of the concept of cellular complex for developing feature addition and removal operators from the cellular representation of a model [BdB98]. A feature model is partitioned into nonintersecting cells. Each cell stores the list of features to which it belongs to, referred to as the cells owner list. The two modification operators defined on the cellular model in this work are insertion and removal of feature shape. The two main effects of any of these operators on the cellular model are topological modification and owner list modification. In case of insertion of a feature, either the cells just touch each other (no new cell is created in this case) or overlap volumetrically (a new cell is created). The new cell thus created inherits the owner list from its originating cells. In case of feature removal, the cells and their references are removed from the owner lists of other cells. The resulting cells are checked for their owner lists. If the owner list for a cell is found to be empty, that cell is also removed. The cellular representation scheme and the operators developed are useful in handling feature interactions in various applications such as multi-view modeling and feature visualization. This is a general approach for feature addition and removal in a cellular model that can be used for model simplification in addition to multi-view modeling related applications. Effective Volume Operators: Feature rearrangement based on given criteria for Level of Detail (LOD) and Level of Abstraction (LOA) is required for detail removal, leading to a resulting shape different from the original feature shape owing to the non-commutative nature of the union and subtraction Boolean operations. The operators involving combination of Boolean operations giving geometrically and topologically valid resulting features after rearrangement are called effective volume operators. Let V_i denote the volume of the solid primitive of a feature F_i, denote a Boolean operation, and M_n denote the resulting model obtained by applying n Boolean operations between $n+1$ solid models: [Lee05, H.05, LL05, LL06a, LLK06].

$$M_n = \prod_{i=0}^{n} \bigotimes_{i} V_i, \text{ where, } \bigotimes_{0} V_0 = \phi \bigotimes V_0 \tag{2.1}$$

where, ϕ is an empty set. If the j^{th} Boolean operation $\bigotimes_j V_j$ is moved to m^{th}

position, M_n can be represented as follows:

$$M_n = \left[\prod_{i=0,i\neq j}^{m} \bigotimes_i V_i\right]\left[\bigotimes_j \left(V_j - \Sigma_{l=1}^{m-j}\varphi\left(\bigotimes_j, \bigotimes_{j+l}\right)V_{j+l}\right)\right]\left[\prod_{i=m+1}^{n}\bigotimes_i V_i\right] \tag{2.2}$$

where,

$$\varphi(a,b) = \begin{cases} 0, \text{if } a = b \\ 1, \text{ otherwise} \end{cases} \tag{2.3}$$

Equations 2.1 to 2.3 can be explained by the following example [Lee05]. Figure 2.4 shows a sample NMT model created by applying five form features. If the order of features is changed to $F_0 \to F_2 \to F_3 \to F_4 \to F_1$ by moving F_1 to the last location, the Boolean operation sequence corresponding to the new feature order will be $P_0 \cup P_2 - P_3 \cup P_4 - P_1$. However, if Equations 2.1 to 2.3 are applied, the effective zones of F_0, F_1, F_2, F_3, F_4 are $P_0, P_1 - P_2 - P_4, P_2, P_3$ and P_4 respectively. The Boolean sequence becomes $P_0 \cup P_2 - P_3 \cup P_4 - (P_1 - P_2 - P_4)$, and its result is the same as the original part shape as shown in Figure 2.5. Note that the dashed line in the figures represents a hole generated by subtracting a wireframe from a solid.

Voxels: A voxel is a volume element, representing a value on a regular grid in three dimensional space. Voxels can be understood as the three dimensional equivalent of a pixel, which represents two dimensional image data. They are frequently used for visualization and analysis of volumetric data commonly occurring in CAD/CAM, bio-medical, scientific data visualization, etc.

Partial Entity Structure (PES): PES is a compact and fast model representation scheme. It is developed as a non-manifold B-Rep data structure. The topo-

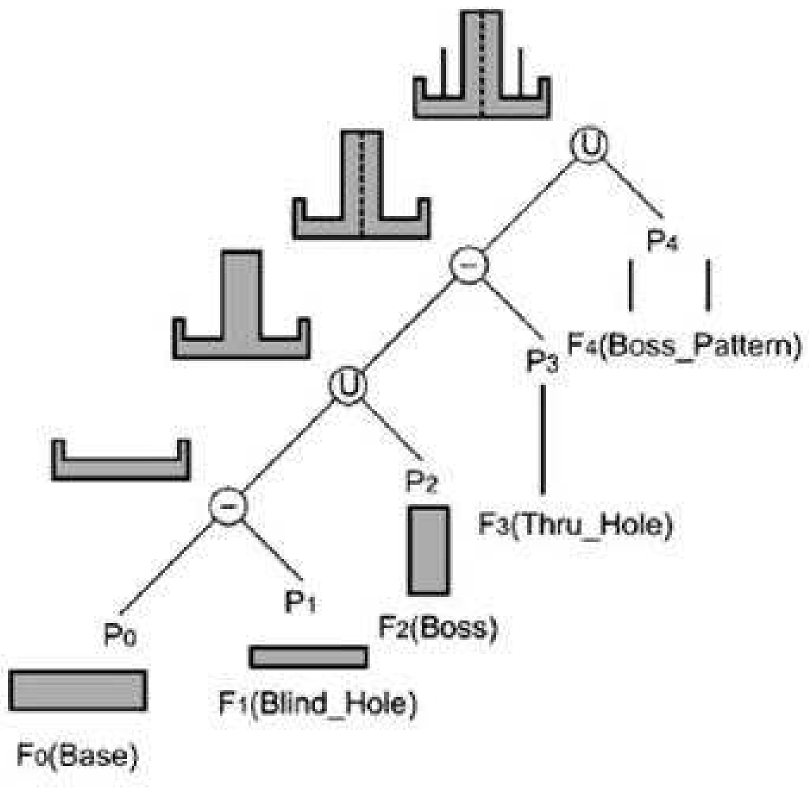


Figure 2.4: *An example of NMT Modeling [Lee05].*

logical entities in PES are of two type namely, primary and secondary entities. The primary topological entities contain 0-cells (vertices), 1-cells (edges), 2-cells (faces) and 3-cells (regions) and their bounding elements (*i.e.*, loops and shells). The secondary topological entities consist of partial vertices, partial edges and partial faces. The partial vertex represents the non-manifold case where more than one two-manifold surfaces are connected to the vertex. A partial edge represents the non-manifold condition where more than two faces are connected to an edge. A partial face is used to represent the non-manifold condition where a face is adjacent to two regions [LL01]. The secondary entities are called partial topological entities or partial entities. Geometrically, partial entities represent adjacency relationships

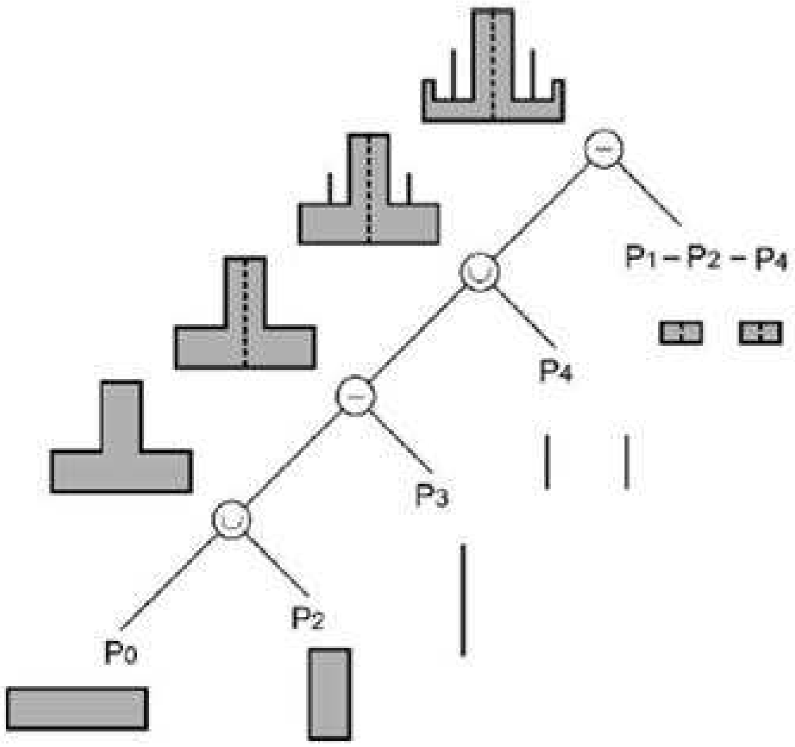


Figure 2.5: *Feature rearrangement based on "effective zone of feature" [Lee05].*

among the primary entities. Figure 2.6 illustrates the different partial entities.

Medial Axis Transform (MAT): The MAT was initially proposed by Blum as a technique for biological shape measurement [Blu67]. It provides a skeleton-like representation of the shape of a model, based on the geometric proximity of its boundary elements. MAT can be described as the locus of the center of a maximal inscribed disk as it rolls around the object interior, where a disk is maximal if it is contained within the object but not within any other disk [DAP00]. Concept of Medial Axis can be easily translated into 3-D domain to Medial Surface by replacing circle with sphere in the former definition. Concept of MAT is very useful in generating FEA idealization models as it provides geometric and topological proximity

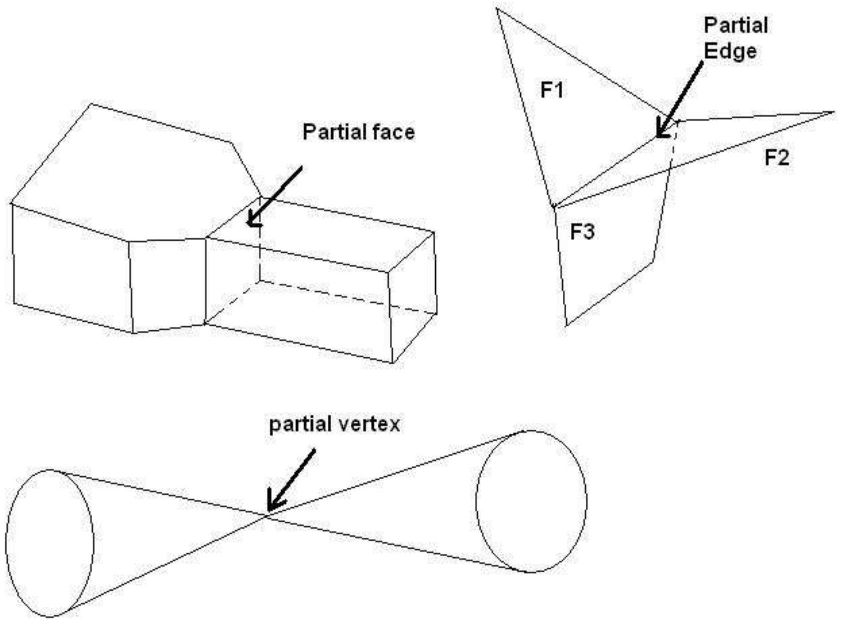
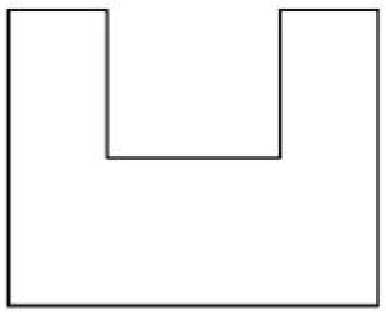


Figure 2.6: *Partial entities.*

information enabling appropriate idealizations to be identified. A 2-D model can be dimensionally reduced to a 1-D beam and a 3D solid can be replaced with a 2D surface, which is comparatively computationally less expensive to analyze. Figure 2.7 shows a channel section and its medial axis transform. One of the modifications to MAT is θ-MAT which is a subset of MAT. For a θ-MAT, separation angle for each point on the MAT is greater than the specified angle θ. Separation angle for a point on an MAT is defined as the angle subtended on it by its nearest neighbors on the polyhederal model. Thus, a θ-MAT for a model approaches the MAT for the model as θ approaches zero. The θ-MAT of a model is computationally more stable than its MAT as it is not affected by fineness of tessellation of the model.

Surface Simplification by Decimation: Decimation is a class of technique to


(a) *A channel section.*

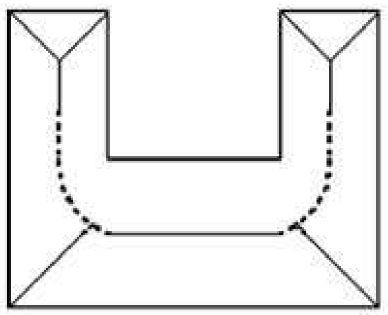

(b) *Medial axis transform of the channel section [RG05].*

Figure 2.7: *An example of medial axis transform.*

simplify the surface geometry of a mesh model by removing or simplifying topological entities (vertex, edge or a face) and making changes in the model to retain the topology of the overall model. In case of vertex decimation, a vertex along with its associated triangles is removed and the hole created is retriangulated to maintain the topology. In case of edge decimation (or edge collapse), edges are collapsed into vertices. In case of face decimation (or face collapse), faces are collapsed into vertices. The entities are selected for removal based on predefined error metrics that evaluate the geometric impact of their removal. One of the reported error metrics used is quadric error [LLKK04, GH97]. Using the fact, that each vertex is an intersection of several planes, the quadric matrix Q (a 4×4 symmetric matrix) is defined for very vertex by aggregating the coefficients of equation of the planes in a way such that the quadric error (e) associated with the given vertex $v = [v_x, v_y, v_z, 1]$ is given by a quadratic form as shown in Equation 2.4.

$$e = v^T Q v \tag{2.4}$$

The quadric error (e) can be used for ranking the vertices for consideration of decimation. In the particular instance of edge collapse $(v_1, v_2 \to v)$, the quadric corresponding to v is calculated as sum of quadrics associated with v_1 and v_2. The approach is quite general and can be applied to face decimation as well, by determining quadric matrix for face using summation of quadrics associated with vertices defining the face. The decimation approaches were originally introduced for obtaining multi-resolution models by simplifying the entire model at a time. Recent techniques involve selective simplification of surface artifacts using the same approach but limiting the simplification process in the region of interest [LLKK04].

2.1.2 Techniques Based on Surface Entities

Boundary representations describe objects in terms of surface entities. Hence, many researchers have developed techniques to perform simplification using surface entities. The techniques where surface artifacts of the given geometric model are simplified are referred to as techniques based on surface entities in this chapter. These reported techniques fall into the following three subcategories low pass filtering, face cluster based simplification and size based entity decimation. We discuss reported work in each category in the following subsections.

2.1.3 Low Pass Filtering

The low pass filtering using Discrete Fourier Transform (DFT) is a technique based on operators acting on surface entities [LL98]. This model simplification technique is applied mainly to Finite Element Analysis (FEA) mesh generation. A shape needs to be digitized before performing discrete Fourier Transform on it. The input model is in the form of a 2-D grayscale digital image of resolution 512 x 512.

The black pixels represent the locus of points satisfying Equation 2.5 and the white pixels represent the surrounding space.

$$h\left(x,y,z\right)\begin{cases}\geq T_0, & \text{if } (x,y,z)\in V\\ < T_0, & \text{if } (x,y,z)\notin V\end{cases} \tag{2.5}$$

where h represents the surface of the model, V represents the volume occupied by the model and T_0 is the threshold representing model boundary. The black pixels are assigned a value of 1.0, whereas the white pixels are assigned a value of -1.0. The transition pixels where the value switches from -1.0 to 1.0 are assigned a value of 0.0 to represent the model boundary. Using these values as the height, a surface can be plotted. After this, Fourier transform is applied to the surface function of the part model. The surface is thus, expressed in the frequency domain, after application of Fourier transform. Authors have stated that the low frequency terms constitute the overall shape while the high frequency terms represent the detailed features in the transform of the surface function of the part model. When the high frequency components are removed from a model, it is called as low pass filtered (LPF) model. In the LPF model thus generated, the sharp edges are converted into smoothed edges. The smoothed edges are not desirable in analysis model as they suppress the important effect of stress concentration. To get the feature suppressed models with sharp edges retained, the original unsimplified model is compared with the LPF model. The average distance between the edges and faces of the original model and LPF model is evaluated as a metric to denote the complexity or detailness of the respective entity. If the metric is below a specified value, the entity is considered detailed and if the metric is greater than the specified value the entity is considered to belong to the overall shape. Authors have also explored the possibility of using wavelets instead of Fourier basis function in this method of low pass filtering. Wavelets have narrow support and, thus, are not suitable for removing detailed features. All the small features influence the overall shape of the model. Thus, the basis function describing a small local feature influences the entire domain rather than a region lying in its vicinity. Use of the metric ranks the features according to size and acts as a measure for determining the order of features to be suppressed, as it represents the deviation between the original model and the filtered model. The computational time for mesh generation is significantly reduced by using this method. It is also considered as a potential technique for feature recognition and is reported to be robust in suppressing details even for complex geometries. The method is not fully automated and some human intervention is required to select the features to be removed. The method is currently developed only for 2D surfaces and not extended to 3D volumes. The methodology is also limited in terms of filling the gaps (the removed faces) if the method is extended to 3D models.

2.1.4 Face Cluster Based Simplification

The face clustering algorithm reported by Sheffer is a technique to cluster the faces in the input model [She01]. The clusters thus formed represent regions of interest that may be considered for simplification. The method is mainly developed for simplifying the models for FEA mesh generation application. There are three main steps followed in this approach: face clustering, finding the collapsible faces and simplification. The model is represented as an adjacency graph with all the faces represented as initial cluster nodes and connecting edges as arcs. The arcs are then contracted to cluster the faces. The face node pairs to be clustered are selected based on the weights assigned to the arcs joining them. The weight depends upon the geometry indices denoting the compatibility criteria (determined heuristically) of non-manifold input. Once the face clusters are made, a collapsibility check is done for all the face clusters. The collapsibility check depends upon criteria like boundary preservation, region size, region smoothness and simplicity of region boundary shape. The metrics for each criterion are defined mathematically in this paper. These metrics are the error measures for evaluating the effects of removing the clusters on the overall model geometry. Finally, decision is taken by the algorithm about which clusters need to be collapsed based upon the metrics evaluated for each cluster. Collapse operators are defined based on virtual topology, where a face is split into different faces equal to the number of adjacent neighboring faces sharing a boundary with the original face (the neighboring faces) [SBB97]. After this, the newly created faces are merged with the respective neighboring faces with which they share the boundary. It can be seen that only the connectivity between the faces changes; however, the geometry is preserved and hence it is called a virtual topology based collapse. The advantages of the above described approach are its applicability to faceted, free form geometries as well as non-manifold models. A curvature-based index is introduced in this paper which can be used for handling curved regions. The curvature-based index is used in the cases where the geometry of clusters is non-planar and planer index cannot be used. The method is applied on several example parts and it is found that the simplified models have significantly fewer elements without loss of accuracy. This is because the simplification process only changes the connectivity but the original geometry is maintained. However, the ranking method for clustering the faces is not systematic and is based on heuristic criteria. Also, since the face clustering algorithm is of the greedy type, the resulting clusters are not necessarily optimal. Inouea *et al.* reported a face clustering technique for FEM mesh generation application similar to virtual topology [IIYF01]. The approach iteratively merged the model faces to obtain face cluster regions having sufficiently large area compared to the mesh element size, smooth enough face boundary and flat surface. Authors defined metrics for mesh area, boundary smoothness and surface flatness and used them for ranking the faces for merging purpose. Dey *et al.* reported local modification of automatically generated meshes [DSG97, SBO98]. They defined a priori error metrics based on aspect ratio and dihederal angle measure and suggested model simplification by iteratively removing elements with poor aspect ratio and small angle metric and remeshing. In their approach, the validity of locally modified mesh is ensured by imposing topology based constraints.

2.1.5 Size Based Entity Decimation

Lee *et al.* present a method to generate progressive solid models (PSM) from feature based models using a cellular topology-based approach [LLKK04]. Here cellular topology is used for generating the PSM and then surface entity based operators are developed to simplify the model. The main concept in this paper is to start with a feature based model as input and generate a sequence of solid models representing the underlying object with various level of details. The intended purpose of PSM is to stream models over a network efficiently. A PSM is defined as an overall shape and a set of details. Authors argue that the problems with traditional approaches in generating such progressive model is the high computational cost associated with applying Boolean operations for transitioning between levels of detail and the cost of storing the various shapes. The authors have come up with cellular topology representation of part model to overcome these difficulties in PSM. Feature shapes have an explicit volumetric representation in terms of cells. PSMs can thus be derived from the face based composition and decomposition of cells that eliminate expensive Boolean computations and storage space. In contrast to the spatial representation schemes (voxel based), cell-based representations are exact in nature. The delta volumes are represented using progressive features in case of cell based technique. The features are simplified or suppressed to produce PSMs. The simplification of features is done using edge contraction techniques. The edges in a feature are ranked based on the geometric error introduced by removing them. The edges are then contracted one by one starting from the lowest cost edge. The positive features are suppressed only using the criteria of feature volume. If the volume of feature is below a certain threshold value, the feature cells are attributed to be dummy cell initially. Later, they are considered as positive cells when the model resolution is increased. The approach described above can handle the exact NURBS representation of the underlying models. The Boolean operations for progressive modeling are replaced by simpler topological entity manipulation at cell level, which is computationally much better. The cell and feature simplification criterion is mainly designed for solid model transmission. Complexity of features or the importance of features with respect to a particular application is not considered in this technique as model simplification criteria. The cell based transformations are used for transforming higher dimensional cells to same or lower dimensional cells *e.g.*, from face to edge. The techniques where operators defined using such cellular transformation acts on surface entities like faces, edges and points for simplification, are studied under the surface entity based category in this chapter. This technique uses cell to cell transformation functions for simplifying models by suppressing features for Finite Element Analysis model preparation [LAPL05]. Authors have presented a general methodology to suppress or reinstate features from a B-Rep model by using invertible cell to cell mapping. The cell to cell mapping functions are implemented in the CADFix software in terms of three surface based operator pairs namely collapse/explode, split/join and insert/remove [CAD05]. In case of collapse operation, the transformation of a cell to a lower dimension takes place. For example, a two-dimensional face is collapsed into a one-dimensional edge. The explode operation is defined as inverse of collapse operator based on collapsing history *i.e.*, the information about collapsed face is used to reinstate the same face when the collapsed edge is exploded. In case of a split operation, a cell within a model having dimension greater than zero (*i.e.*, a point), can be split into two or more cells of similar dimension by introducing one or more cells of lower dimension. For example, a face (2D cell) can be split into two faces by introducing an edge (1D cell). The splitting cell is constrained to lie within the parent cell, so that an edge partitioning a face lies on the face. The same logic applies to splitting an edge by introducing a vertex and a region by introducing a face. Joining two faces again uses splitting history and combines the faces split earlier. In case of an insert operation, lower-dimensional cells are added in the interior of a cell. For example, a vertex can be inserted on an edge. The insert operator is useful in CAE when cellular partitioning is required to represent loading and boundary conditions. In the remove operation, a lower dimensional cell is deleted from the higher dimensional cell. For example, a loop representing hole on a face can be removed using this operator. To suppress features from a B-Rep model, it is first expressed in cellular representation and then the cells are mapped into simplified model by using the operators described above. One of the important requirements in analysis models is to suppress the narrow regions. A narrow region is defined as part of surface where two of the boundary edges come in close contact [LPA03]. The narrow regions pose problems in mesh generation because of huge size differences within the region leading to poor element quality. To suppress the narrow regions, the edges in proximity are identified and then the face is split at each end of the region. The resulting long narrow face is then suppressed by collapsing the short edges to vertices and merging the two long bounding edges. No error measure or threshold has been defined to select the features to be removed. Instead, they are selected interactively by the user for simplification purposes. The main advantage of this methodology is the generality of its implementation. The method can be implemented by explicit cellular representation or direct geometry based operators. An audit trail or analysis history is generated that is useful for analysis to compare various simplification strategies thus obtained by the system. Narrow regions can be suppressed using this approach, which is important for mesh generation. A possible limitation is that the faces are repartitioned in this approach which requires some post processing for generation of meshes of desired density. Focault *et al.* reported a topology simplification technique for finite element mesh generation [FCF+08]. The model for FEA should be prepared in such a way that the mesh generated for it represents the model as closely as possible and at the same time minimizes the computation time for analysis. Another requirement of the simplified model is that the mesh should represent the boundary condition domains, *e.g.*, a point on the part where forces or displacements are applied should be represented by a coinciding node. Also, the mesh edges generated from the simplified model must exactly match the sharp corners of the geometry to minimize the discretization error. The authors have developed a Mesh Constraints Topology (MCT) based model simplification scheme to address the sharp corner matching requirements. MCT entities are defined as composite topological entities created to suit mesh generation requirements stated before. The MC face is a poly-surface, defined as the union of Riemannian surfaces constituting the reference model. The MC edge is a poly-curve, defined as union of Riemannian curves constituting the reference model. The MC entities are used to represent the model because they preserve the exact geometries as they are a higher level representation of the low level B-Rep entities that define the reference model. The MCT models are internally represented using hypergraphs. Hypergraphs are graphs with arcs that connect two or more nodes. MCT is defined using three topological adjacency hypergraphs namely, face-edge adjacency, face-vertex and edge-vertex. The authors have implemented graph based operators for deleting MC edge and vertex, inserting MC vertex in MC edge, collapsing MC edge to MC vertex and merging MC vertices for simplification.

The mesh quality constraint for MC entities is decided by their size and curvature. The size for an MC face is defined as distance between face boundaries while for an MC edge as length of the edge. The size of MC entities must be greater than a set size threshold while curvature (leading to deviation angle) less than a set curvature threshold to satisfy the mesh quality constraint. The deviation angle is the angle between the normals of adjacent mesh segments for a given discretization error. For enlargement of thin MC faces, MC edge deletion operator is used. Redundant edges situated in planar regions need to be deleted using MC delete operator. MC vertex deletion and MC edge collapse operators are used to get rid of small MC edges. Constricted sections in MC faces are suppressed by using MC vertex insertion and vertex merging operators. Thus, the MCT operators and criteria along with the mesh quality constraints are used together to simplify the models. The mesh quality constraints used by the authors are mesh element size and discretization error. The mesh element size is represented using over density ratio which is the ratio of the size map to the effective element size. The discretization error is a measure of the gap between the mesh and the geometry. The simplification method described here, retains the topology of the model and thus no defeaturing errors are introduced after simplification. However, FE numerical errors because of changes in the geometry are introduced and that is controlled by refinement schemes. The method described above retains the topology and creates new geometry adapted to mesh quality constraints such as size-map, maximum over-density, maximum deviation angle, and boundary condition zones. Fine *et al.* introduced idealization operators for Finite Element Analysis [FRL00]. The operators are based on vertex removal and spherical error zone concept. Mobley *et al.* reported an object oriented approach to develop surface based defeaturing operators to suppress small features for FEA model preparation [MCC98]. Date *et al.* reported vertex and edge collapse based technique for mesh model simplification and refinement [DKK+05]. They defined three metrics based on overall geometry error, face size and face shape. The metrics are evaluated for edges to determine their priority index for simplification. Veron and Leon reported shape preserving simplification for a polyhederal model using vertex decimation [VL98]. A metric based on discrete Gaussian curvature and angle subtended by incident edges is evaluated for each vertex in the model. The vertices are then removed based on their rank priority based on above metric. For example, the vertices with curvatures near zero denote that they lie on plane and are removed first. The vertices with curvatures farther from zero (both in positive and negative directions) are considered later for removal. All of the techniques listed in this section are summarized in Table 2.1.

Table 2.1: *Summary of techniques based on surface entity based operators for model simplification*

Method	Input format	Features simplified	Simplification criterion	Advantages	Limitations	Application domain
Fourier Transform based low pass filtering [LL98]	2D image	Small boundary edges and island type features	Based on error measure computed as average distance between original and LPF models	Error metric based LOD selection reduces computation time; can be used in feature recognition	Lack of automation; applicable to 2D surfaces; performance issues in 3D extension	FEM model preparation

continued on next page ...

continued from previous page ...

Method	Input format	Features simplified	Simplification criterion	Advantages	Limitations	Application domain
Face clustering [She01]	Gambit pre-processor	Chamfers, fillets, rounds and spherical surface based features	Metric based on boundary preservation, region size, region smoothness and simplicity of region boundary shape	Applicable to both faceted and free form geometries; curved regions can be handled; geometry is retained; only topology changes	face clustering may not be optimal	FEM model preparation
Face Clustering [IIYF01]	Polygonal Mesh	Protrusions and depressions	Error measure based on cluster area, cluster boundary smoothness and flatness	Mesh quality based metrics are used	Faces with large variation in normal vectors not handled; face clustering may not be optimal	Mesh generation for FEM analysis
Face Clustering [DSG97, SBO98]	Polygonal mesh	Protrusions and depression	Aspect ratio and dihederal angle of elements	Approach is simple to implement	Through hole removal may not be possible	FEM mesh generation
Decimation -Cellular topology progressive modeling [LLKK04]	Feature based model	Freeform features	Based on ascending order of progressive volumes	Applicable for NURBS surfaces; computationally efficient	Application specific feature complexity not considered	Network transmission; FEM model preparation
Decimation-Cell transformation based technique [LAPL05]	B-Rep	Holes, fillets and narrow regions	The features are selected manually by the user for simplification	Analysis history useful for simplification strategy comparison; Handles narrow regions	Post processing needed for mesh generation of desired density; Lack of automation	FEM model preparation
Decimation-MCT based technique [FCF+08]	B-Rep	Freeform	Interactive	Preserves topology and creates new geometry adapted to various mesh quality constraints	Lack of automation	FEM model preparation
Decimation-Idealization operators based on vertex removal and spherical error zone [FRL00]	Polygonal Mesh	Holes	Error measure based on a posteriori analysis using tetrahedral finite elements and Interactive	Takes care of large transformations and simplification of complex areas of polyhedron such as saddle points	Only for polyhedral models; error estimator for hexahedral elements not developed	FEM model preparation

continued on next page ...

continued from previous page . . .

Method	Input format	Features simplified	Simplification criterion	Advantages	Limitations	Application domain
Decimation-OOP approach for de-featuring [MCC98]	Polygonal Mesh	Loops, coincident edges, near tangencies	Error measure related to global defeaturing tolerance	Scalable object oriented software design	Protuberances are not handled	FEM Model preparation
Edge decimation based approach [DKK+05]	Polygonal Mesh	Protrusions and depressions	Error measure based on overall geometry error, mesh size and mesh shape	Mesh quality based error metrics are used	Not applicable for assembly models	Mesh generation for FEM analysis
Decimation-Shape preserving polyhederal simplification [VL98]	Polygonal mesh	Protrusions and depression	Shape based error zone and discrete form of Gaussian curvature	Object shape preserved	May not handle large geometric transformation in complex area (such as saddle points)	Mesh generation for FEM analysis

2.1.6 Techniques Based on Volumetric Entities

The techniques where volume based artifacts are removed for model simplification are called volume entity based techniques. Such techniques are further classified into two subcategories voxel based and effective volume based techniques.

2.1.7 Voxel Based Simplification

Andujar *et al.* proposed Trihederal Discretized Polyhedra Simplification (TDPS) based topology reducing surface simplification technique [ABA02]. The aim of this work is finding a reduced valid polyhedral approximation of a solid model such that the maximum solid Hausdorff distance is below an error value. There are three steps followed in this approach: discretization, reconstruction and face reduction. The model is discretized using Maximal Division Classical Octree (MDCO). It subdivides the cubic universe into eight octants and arranges it into an 8-*ary* tree, the leaves of which are labeled black or white based on whether they fall completely in or out of the model volume respectively. Nodes that fall partially in the model volume and partially out of it are labeled as gray. The black and white nodes are not subdivided further, whereas the gray nodes are subdivided till their parts become white or black or the depth of the tree reaches a set level. In the reconstruction step, the MDCO is compared with the original model and polyhedral surface (S) based on the solid Hausdorff distance as error measure is extracted. The surface (S) satisfies the conditions that all the black nodes are completely within S and all the white nodes are completely outside R and all the border nodes (defined as nodes whose at least one of the twenty six neighbors is white) intersect with S. Finally, in the face reduction step, the surface obtained from the previous step representing the original model is further simplified using edge collapse technique. Although it appears that the simplification operator used is surface entity based, the error measures and topology preserving simplification strategy are based on volumetric operators. Hence we have chosen to list this method under volumetric entity based model simplification category. The major advantage of this method is its applicability on individual parts as well as assemblies. Also non-manifold inputs can be effectively handled by this approach to produce 2-manifold solids. One of the limitations in the methodology described above is the performance issue. Small error thresholds require larger subdivision levels for the MDCO. The cost of TDPS largely depends on the subdivision level of MDCO and thus volume based technique is computationally intensive for small error thresholds. Another limitation is the requirement for preprocessing of input model before simplification to align it with the axes so that octree represents the part more accurately. This is because of the isothetic nature of the octree. TDPS method is applicable to collision detection, occlusion analysis, multi-resolution robust Boolean operations, indirect illumination, acoustic modeling and query acceleration. One of the earliest approaches based on operators acting on volumetric entities is reported by He *et al.* [HHK$^+$95]. They developed a voxel based object simplification technique using marching cubes algorithm to generate multi-resolution triangle mesh hierarchy.

2.1.8 Effective Volume Based Simplification

A feature based non-manifold modeling system is developed by Lee *et al.* to address the needs of both CAD and CAE applications simultaneously from a single model [Lee05, H.05, LL06a, LLK06]. This system supports feature based multi-resolution and multi-abstraction modeling capabilities. The main advantage of non-manifold model is its capability to represent any combination of wireframe, surface, solid and cellular models in a unified data structure. Partial entity structure is used to store the model information [LL01]. The detail removal and dimensional reduction at various levels of detail and abstraction requires features to be rearranged. This rearrangement of features may result in different final shapes owing to the non-commutative nature of the union and difference operators. Authors have developed the concept of effective volume (explained in Section 2.1.1) of features and presented theorems for exchange and rearrangement of features. The detail removal process involves three steps that are briefly stated as follows. Firstly, all the idealization features having application domain as design are extracted from the master model. Secondly, the extracted idealization features are rearranged according to LOD criteria. The LOD criterion selected by authors is based on decreasing volumes. If the volume is below a threshold, the corresponding feature is considered for simplification. Thus, the error measure for selection of features for simplification is the volume of the feature. Finally, the LOD is set interactively to remove the features below the specified level. The multi-abstraction of the model on the other hand is performed by applying LOA criteria to abstract the features. LOA criteria are application dependent. In case of structural analysis applications, aspect ratio of the feature is used to set the abstraction level. In case of injection molding simulation application, the mesh size is used to set the abstraction level. The approach described above is particularly advantageous for multi-resolution modeling and capable of LOD simplification of features. The model representation scheme using partial entity structure enables use of the same model in extracting CAD and CAE information leading to CAD-CAE integration. The model editing operations consisting of a given sequence of Boolean operations is carried out for a CSG tree containing only a subset of the primitives in the merged set, and selectively filters out these primitives from the final result without actually removing them from the merged set. The feature based modeling capabilities for both feature deletion and feature interaction detection are thus computationally efficient and simple. Some of the limitations of the approach described above are explained as follows. When the shape or semantics of a feature are altered due to feature interactions, the predefined abstract model of the feature may become invalid. Authors have pointed out that this problem can be solved by adopting automated medial axis/surface transformation method. The physical constraints and properties such as boundary conditions and mass properties applied to CAD models have also not been transferred to CAE models. All of the techniques discussed in this section are summarized in Table 2.2.

2.1.9 Techniques Based on Explicit Features

Many reported model simplification techniques are based on recognizing explicit application features before simplification. These techniques define class of explicit features related to a particular application such as manufacturing, FEM etc. and evaluating some metric based on which simplification decision can be taken. The techniques fall into the following subcategories prismatic feature simplification, blend simplification, and arbitrary shaped feature simplification based on the type of features covered.

2.1.10 Prismatic Feature Simplification

The three steps followed in the model simplification from polygonal mesh by Date and Nishigaki includes feature recognition from the input mesh, mesh simplification and feature recovery [DKKN06]. The steps are explained as follows. In the feature recognition step, mesh segmentation technique is used. The features considered in this work are blind holes, through holes and bosses. The dihedral angle between two faces shared by the common edge is used for extracting the feature edges. These sets of edges are used to extract regions of interest and segment the mesh into regions. The regions with area larger than the threshold set by a user are classified as base surfaces. After this, the triangles that are not coincident with base surfaces are extracted as Feature Construction Triangles (FCT). The feature type identification and feature parameter extraction is then done using three rules:

Table 2.2: *Summary of techniques based on volumetric entity based operators for model simplification*

Method	Input format	Features simplified	Simplification criterion	Advantages	Limitations	Application domain
Voxel based-Discretized Polyhedra Simplification based topology reducing surface simplification [ABA02]	Polygonal Mesh	Holes, protuberances and edge features	Hausdorff distance between MDCO and original model	Handles assembly models; non-manifold inputs can be handled	Pre-processing required	Collision detection, occlusion analysis, multi-resolution robust Boolean operations, indirect illumination acoustic modeling and query acceleration
Voxel based object simplification [HHK+95]	Polygonal Mesh	Small protuberances and cavities	Frequency threshold for low pass filtering	Handles individual as well as collection of objects	Large number of redundant triangles generated for low surface curvature regions	Efficient antialiased Rendering; can be used for collision detection
Effective volume based technique [Lee05, H.05, LL06a, LLK06]	Polygonal Mesh	Freeform features	Volume threshold	Capable of multi-resolution modeling	Depends on feature volume, not complexity; when shape/semantics of feature are altered, predefined abstract model may become invalid; physical constraints and properties applied to CAD models not transferred to CAE models	CAD-CAE integration and network model transmission

the FCTs with two loops are identified as through holes, the FCTs with one convex loop are identified as blind holes and the FCTs with one concave loop are identified as bosses. The feature parameters are extracted by fitting a least square plane on the feature boundary and finding the depth, width and height for each kind of feature. The location of each feature is determined using the centroid of feature boundary on the least squares plane. In the mesh simplification step, edge collapse is used to eliminate the vertices that are redundant. The main consideration during edge collapse is to keep correspondence between the feature boundary edges (FBE) and the boundary of feature meshes so that features are recoverable. This is done by creating a metric to evaluate the approximation error of FBEs and using it to decide whether the edge needs to be collapsed. In the last step of feature recovery, the feature removing triangles (FRT) are replaced with feature FCTs to reinstate the suppressed features. The condition that has to be satisfied in order to recover a feature is that the feature boundary vertices (FBV) should match with the boundary vertices of the removed feature mesh. If this condition is not met, then the local LOD method is used. In the local LOD method, the mesh is represented as a binary tree containing a set of parent nodes representing the collapsed nodes and corresponding children nodes consisting of the vertices that are collapsed to make the parent node. This binary tree is traversed to recover neighborhoods iteratively by vertex split. The features suppressed can be recovered based on the Level of Detail tree. The approach described above is capable of simplifying simple features (with one base surface) like blind and though holes and bosses. However, nested features such as hole in a pocket cannot be simplified. Another reported technique to recognize and suppress features for finite element model preparation is given by Ribelles *et al.* that utilizes face clustering technique for feature recognition [RHG+01]. Feature based methodology for finite element model idealization has been discussed by Dabke *et al.* by extracting axis-symmetric, plane and solid features from input B-Rep model [DPS94].

2.1.11 Blend Simplification

The approach by Zhu *et al.* deals with simplification of fillets and round features in a B-Rep model [ZM02]. The main application of model simplification in this work is to prepare the model for further feature recognition purposes by removing fillets and rounds. From the input B-Rep model, convexity of all the topological entities namely faces, edges and vertices are characterized, which in turn is used to identify the trace faces. Trace faces are the smooth faces obtained by blending sharp edges while modeling. The trace faces are then used to identify the fillets and rounds in the model. After identifying them, the topological entities pertaining to these features are cleaned and the gaps thus produced are filled up by a method developed by the authors named as incremental knitting process. The authors have underlined three main considerations while suppressing the fillets and round features, namely topological consistency, geometric consistency and reversibility. By preserving the topological consistency, it is meant that the resulting model should be a manifold model. When the fillet faces are removed from a model, gaps are created. These gaps may make the model topologically invalid. The gaps should be filled by topological entities to keep the model topologically valid. By geometric consistency, it is meant that the edge replacing the fillet surface should lie on the planes of supporting faces of the suppressed fillet. Reversibility means that the attributes of the suppressed fillet should be preserved to roll back the model if needed. The trace faces are classified by the authors into three types as toroidal, cylindrical and spherical. The toroidal faces are generated by blending circular edges. The cylindrical faces are generated by blending straight edges. The spherical faces are generated by blending a convex vertex. Several trace faces are connected together to form face chains. Face chains forming closed loops are called closed face chains; otherwise, they are known as open face chains. Homeomorphic equivalence is used to map the face chains to a disc or a ring. The disc or ring representation is used to test the occurrence of gaps when fillet faces are removed. The gaps are then knitted using KR (knitting on ring) or KD (knitting on discs) algorithms developed by the authors. KR and KD algorithms are used to introduce new topological entities after removing fillets to maintain topological and geometrical consistency. To ensure the reversibility of the model simplification operation, the properties of the fillet such as radius is added as an attribute to the topological entity which is introduced in place of that fillet. In this approach, the recognized fillets are selected interactively by the user for further application of simplification operations. The approach described above is general enough to handle various topological configurations (ring and disc type) of fillets and rounds. Also, the suppressed fillets and rounds can be reinstated. The approach, however, only deals with the constant radius blends and needs to be extended to handle variable radius fillets which are not uncommon in mechanical parts. The trace faces are identified by rule based approaches by exhaustively characterizing all the topological entities (convexity checks), which can be computationally challenging when the model size is large. Venkataraman and Sohoni discussed simplification of blend type of features by recognizing the topological entities representing the blends and then removing the entities [VSR02]. Tautges discussed model simplification for finite element mesh generation using hydraulic diameter based feature size metric and recognized blends and bridge features for removal from input B-Rep model [Tau01].

2.1.12 Arbitrary Shaped Feature Simplification

Venkatraman and Sohoni implemented the delete face operation to remove a set of faces corresponding to a particular feature [VS02]. Once the feature is removed by deleting the faces, the gaps produced are filled up by extending or contracting the neighboring faces of the removed feature. In this case, both the additive and subtractive features (in terms of volume) are considered as sets of faces. It is assumed by the authors that once the faces corresponding to the feature are removed from the model, it is possible to patch up the gaps by just contracting or extracting the neighboring faces without adding new faces. The faces that constitute a particular feature are called feature faces. The faces that are neighboring to the feature faces *i.e.*, which share at least one edge or one vertex with the feature faces are called external faces. The edges that are along the boundary of the feature faces are termed as boundary edges, and vertices along the boundary are termed as boundary vertices. The edges that are not along the boundary but touch the boundary vertices are called external edges. Each external edge has two faces sharing it, termed as the left and the right face. When a feature is suppressed, the feature faces are removed and external edges are extended to a face known as opposite face. The opposite face is detected by finding the minimum Euclidean distance between the external edge and each external face. The vertex sense is another important attribute that is required to be determined for deleting the face set. The vertex sense is related to the sense of Gaussian curvature as positive or negative on the edge. The point where the external edge intersects the opposite face is called opposite vertex. The vertex sense of opposite vertex is determined by finding the sign of Gaussian curvature. The features suppressed using the above approach are pockets and slots. There is no error measure defined to select the features for simplification and the features are selected interactively by the user. The technique described above is useful in suppressing both protrusion and depression types of features. The algorithm was tested on a large number of cases such as multiple boundary loops and degeneracies and found to be successful. Operations like extension, contraction and merging of neighboring faces to suppress a feature are handled effectively. The assumption that operations on neighboring faces suffice to construct the feature volume without requiring additional faces may not cater to cases where new faces need to be created for constructing feature volumes. Such problems are usually under-constrained with multiple solutions. Authors mention that additional heuristics would be required to deal with such cases. Another limitation is lack of model simplification history storage and processing and, thus, in this system, the simplified features cannot be reinstated. The features suppressed in the approach used by Joshi are holes, fillets and bosses from a B-Rep surface model representing sheet metal components [JD03].

The two steps followed in this paper are recognizing the feature faces and replacing them with new surface(s) to suppress the feature. The hole is recognized by determining the closed loops formed by free edges. The free edges are those which are not encountered more than once when each surface patch is traversed along their respective edges. The loop with the largest perimeter is identified as the outer boundary, while other loops as regarded as holes. Once the holes are recognized, they are suppressed by introducing a surface patch between the recognized holes boundary or just by removing the boundary edge constraints from the NURBS representation of the surface containing the hole. The fillets are identified by determining the curvature of iso-parameter curves at various points on the surface patch. If the curvature of all such iso-parameter curves is same, then the surface is identified as a constant radius fillet; else it is characterized as a variable radius fillet. Spring and cross edges are then identified by comparing the curvatures along the edge and perpendicular to the edge for the face under consideration and its adjacent face sharing the same edge. If the curvature along the edge is equal on both faces and corresponds to the radius of the fillet, then the edge is identified as a cross edge. If the curvature in a direction normal to the edge is greater than a threshold and also greater than the curvature of the adjacent face in the perpendicular direction, the edge is classified as a spring edge. The threshold is determined based on the bounding box of the object. After this, the edges of the fillet are chained and sequenced. Once the fillets are recognized and sequenced, they are suppressed as follows. The edges of the fillet surface that are common with the support surface are extended in tangential direction of the fillet and the intersection curve is determined. Thus, the fillet surface is replaced with two new planar surfaces sharing edges with the support surface and the newly generated intersection curve. The boss is recognized after the fillets are already recognized and fillet chains and rings are identified. The algorithm assigns an attribute to each face specifying the fillet chains and rings contained in it. For all planar faces the adjacent faces on its outermost loop are checked and if they belong to the same fillet ring, the planar face is marked as boss top and the fillet ring is marked as top ring face. After this, adjacent faces of the top ring are checked to identify the bottom ring. All these surface patches are recognized as parts of the boss. To suppress the boss all the components of the boss like the top face, the top ring and the bottom ring are removed and replaced by a smooth continuous flat surface patch. To do this, the edges between the bottom ring and base surface are chained to form a closed loop and a flat surface patch is created for this loop in the same way as it is done for a hole. The generated edge may not be the one that existed before the blend generation. The reason for this is that the effect of curvature of freeform base surface is not accounted for while calculating the intersection curve. This approach can sometimes recognize a surface as a blend even if the surface is not intended to be so. The concept described above is applied for mesh generation for Finite Element Analysis, particularly for design of forming dies and molds where small holes and bosses are not very important in the early stages of analysis. The quality of mesh is analyzed with respect to the number of elements, aspect ratio, warping and the mean value of element included angle. It is shown that for several test cases, the mesh quality for the given criterion improves after feature simplification. The approach is particularly advantageous in handling freeform surfaces. The method can tackle arbitrary shaped holes and variable radius fillets and rounds. Compound features such as darts and beads are reportedly handled by this approach which is useful for a completely new family of parts having such features. The features cannot be recovered once they are removed from the model database as no simplification history is maintained. Moreover, maintaining a history of operation and reinstatement of feature may be difficult to incorporate in this architecture. Kim *et al.* reported a system for multi-resolution feature simplification using three operators, namely wrap-around, smooth out and thinning [KLH$^+$05]. The method is applied to finite element model preparation and network transmission multi-resolution modeling. The features are first recognized using rule based techniques and then various operators are applied to suppress them. In case of wraparound operator, the model is considered to be wrapped by a thin plastic cover. The parts of model hidden by the cover are considered for simplification or removal. The wraparound operators are explained in the earlier papers by Koo and Lee [LL02] and Seo *et al.* [SSK$^+$05]. This operator is especially suitable for handling concave features. The operator is applied to features that are detected as blends, chamfers and concave features. Authors have applied certain rules to recognize feature. For example, the existence of cylindrical surface on concave edge is recognized as a fillet and the existence of inner loops of convex edges connected to more than one face is recognized as a concave feature. The recognized features are then simplified using wrap-around operator. The limitation of wrap-around operator is its volume-additive nature in case of a protrusion type of feature. If the protrusion feature is very small, a volume removal approach is more useful. To address convex features such as bosses and ribs, authors propose smooth out operator, which can be considered as smoothing operation carried out by a sand paper or chisel. The bosses are identified by finding concave inner loops connected to more than one face. The ribs are identified by finding face pairs which are much smaller in size than the other faces and contain two concave edges. If two faces are connected by a convex edge, then the pair is identified as a rib face set. Once the features are identified, the smooth-out operator removes the faces corresponding to each boss and rib and the neighboring faces are extended and stitched to fill the gaps produced by removal of feature. Smooth out operator cannot be applied to the concave features. Finally, thinning operator is applied to model already simplified by wrap-around and smooth out operators. In the thinning process, the dimension of features is reduced to 2D or 1D using mid-surface representation. The model is searched for face pairs having same geometry (the geometries supported are plane, cylinder, sphere, cone, torus, and offset face). If the distances between pairs of faces are found to be very less as compared to the face dimensions, the pairs are identified as candidate for dimension reduction using thinning operators. The thinning operator introduces an additional face between the faces in the pairs recognized for thinning and the introduced face is trimmed to the shape of bounding box of the face pair. After this, the thickness information is attached to the introduced face and it is stitched to the model. In each of the operators explained, the history of simplification is stored and the reinstatement of the simplified features in each case can be obtained using the history information. The major advantages of the method described above are its applicability to assembly models and interoperability. The implementation is developed in Parasolid kernel (used in many commercial CAD systems) and hence the algorithm is portable. The rule based feature recognition is one of the computational overheads and authors propose that feature based models can be used to overcome this limitation. In case of simplification of assembly models, the interface between mating parts is not considered and hence the mating information is not preserved. All the techniques mentioned in this section are summarized in Table 2.3.

Table 2.3: *Summary of techniques based on explicit feature simplification operators*

Method	Input format	Features simplified	Simplification criterion	Advantages	Limitations	Application domain
Feature simplification from triangular meshes [DKKN06]	Polygonal Mesh	Regular shaped blind holes, through holes and bosses	Quadric error based metric	Suppressed features can be recovered based on the LOD tree	Nested features not addressed	FEM model preparation
Feature removal from polyhedral model [RHG+01]	Polygonal Mesh	Cavity and protrusion types of features	Quadric error based metric	Uses general definition of feature	Implemented only for polyhedral models	FEM model preparation
Finite element based feature removal [DPS94]	B-Rep	Axis-symmetric, plane, solid	Interactive	FE based features used	FE features like beam, pipe, shell, etc. not defined	FEM model preparation
Automatic fillet/round simplification [ZM02]	B-Rep	Constant radii fillets and rounds	Interactive	Handles both ring and disc type topology; reinstatement of features possible	Variable radii fillets not covered; computationally intensive feature recognition step	Feature recognition
Blend removal from B-Rep [VSR02]	B-Rep	Blends	Automatic	Complex blend networks with interacting features are handled	Variable radii blends not covered	Feature recognition

continued on next page . . .

continued from previous page . . .

Method	Input format	Features simplified	Simplification criterion	Advantages	Limitations	Application domain
Detail Reduction for mesh generation [Tau01]	B-Rep	Blends	Hydraulic diameter of surface and volume	Size based metric; Portable across modeling systems	Roll-on and face edge blend not covered; bridge removal not covered	FE model preparation
Feature simplification using *face delete* operation [VS02]	B-Rep	Arbitrary shaped protrusion and depression	Interactive	Simplifies both protrusion and depression types of features	Could not handle cases where extension and contraction of adjacent faces can't patch the gap produced; under-constrained problems with multiple solutions not considered	Feature recognition
Feature simplification for freeform surface models [JD03]	B-Rep	Arbitrary shaped holes, fillets (constant and variable radii), free form surface features	Interactive	Handles freeform surfaces; arbitrary shaped holes and variable radius fillets and rounds	Suppressed features cannot be recovered; cannot simplify features such as notches, lances, etc.	FEM model preparation
Feature simplification using wrap-around, thinning and smoothing operators [KLH+05, LL02, SSK+05]	B-Rep	Blends, chamfers, passages and concave regions	Based on summation of area of faces contained by features	Applicable to assembly models; interoperability and portability to commercial system	Rule based feature recognition results in overhead; mating information not preserved while simplifying assembly models	FEM model preparation, network model transmission and multi-resolution viewing

2.1.13 Techniques based on Dimension Reduction

In many applications, reducing the dimensions of CAD models is beneficial. For example, consider a long round slender bar of uniform diameter. If we model this long slender round bar as a 1D beam, there will be negligible effect on the accuracy of the analysis applied to the beam but the computational time will reduce dramatically. Hence, dimensional reduction is studied by the physics-based simulation community (especially finite element analysts) for model simplification purposes.

2.1.14 Medial Axis Transform based Dimension Reduction

One of the well established techniques for dimension reduction is medial axis transform (MAT). There is a large body of literature on MAT and it is difficult to include all of them here. We, thus, suggest the interested readers to refer to the survey paper by Attali *et al.* for details about medial axis transform based techniques [ABE09]. In this section, we will discuss few representative papers that use MAT based model simplification. In addition we will also discuss techniques based on θ-MAT which is a modified approach for MAT and mid-surface abstraction. Donaghy *et al.* used Geometry Idealization pertaining to dimension reduction to simplify a model for analysis and other downstream operations [DAP00]. The technique used for dimension reduction used by authors is Medial Axis Transform (MAT). The process of development of MAT follows three steps. In the first step, Delaunay triangulation of the object is performed. In the second step, the circum-circles for each triangle generated are determined. In the last step, the circum-centers are fitted on a curve. The fitted curve is called the medial axis. To determine whether the dimension of the object is to be reduced, aspect ratio and taper criteria are used. The lower bound for aspect ratio is determined as the ratio of the length of the shortest edge bounding the region and the maximum disk diameter in the local region. The taper is determined as the maximum rate of change of diameter with respect to medial edge length. The high aspect ratio or low threshold indicates variation in slenderness property of the object. If an object region has an aspect ratio greater than the threshold or a taper value lower than the taper threshold, then the region is suitable for modeling with 1D element. If not, then the region is either kept in its original dimension for meshing with 2D elements, or dimensionally reduced to an equivalent $0D$-point element. After dimensional reduction has been performed, physical properties are imparted to the reduced dimension model. The main parameters related to physical properties discussed by the authors in the context of FEA are the moment of inertia tensor and the position of the neutral axis. This approach is used for dimension reduction of 2D planar or 3D shell model. The technique is not applicable for simplification of other types of 3D solids. Sud *et al.* presented an algorithm for homotopy preserving medial axis simplification of polyhederal models with a linear computational complexity and proposed applications to mesh generation and shape analysis [SFM05]. Foskey *et al.* utilized the concept of θ-MAT for dimension reduction and model simplification of polyhederal mesh models [FLM03]. θ-MAT is more stable algorithm compared to MAT computationally. The potential applications of the approach suggested by Foskey *et al.* are in the area of mesh generation and shape analysis [FLM03].

2.1.15 Mid-surface Abstraction

Rezayet presents a technique to abstract the part model in terms of mid-surface [Rez96]. The main applications of this approach are in FEA model preparation and feature recognition. Authors have pointed out several benefits of mid-surface abstraction in comparison with medial axis transform such as volume preservation, non-creation of MAT branches, effective simplification of features and detail removal, reflection of part form and use of geometric reasoning to define the shape. There are four steps involved in the generation of mid-surface, namely pairing surfaces, topology based adjacency graph creation, mid-surface patch generation and sewing the patches based on adjacency information. In the surface pairing step, all the faces excepting the end-caps (faces on the edges of solid model) and orphans (faces not on the thin wall sections) are paired. The pairing process involves arbitrary selection of a face as the seed face and casting of a ray in the material direction from the seed face. If the ray hits another face, the faces are paired and all the related faces *i.e.*, the faces sharing common edges with the paired faces are tagged and processed. From the remaining faces, again a seed face is selected and the process is repeated till all the faces are processed. The processed faces are then used to create adjacency graph. The nodes of the graph are the faces and the arcs are the relationships between the faces. The two types of relationships defined are pairing relationship and the common edge relationship. The patterns in the graph indicate particular types of features. The paired faces are then interpolated to create mid-surface patches, which may be intersecting. The topological adjacency information is used to trim the patches. In case of plane faces, a 2×2 grid is used, while in case of non-plane faces, a 15×15 grid is used to create the interpolated points. The thickness distribution is then assigned to the mid-surface thus generated. The thickness of the part at an arbitrary point P, on the mid-surface between two parent surfaces S_1, and S_2 is T, where,

$$T = dist(P, S_2) + dist(P, S_1) \tag{2.6}$$

The mid-surface abstraction is applied to the entire model simultaneously and no error threshold is defined to select the parts of the model to which abstraction is to be applied. In this approach, small features such as holes are simplified automatically when the mid-surface is generated. Material distribution is effectively represented in this approach by introducing thickening in the direction of the draft and by introducing thinning in the direction of the undercut. In the area of dimension reduction, other reported work is by Chong *et al.*, who presented a technique to decompose the solid model into parts and then applied mid-surface abstraction for simplification for finite element model preparation application [CKL04]. All the techniques listed in this section are summarized in Table 2.4.

Most of the model simplification techniques described above are developed for finite element applications. Another class of model simplification techniques used in collision detection (inspired by model simplification techniques in graphics domain) are described next.

- *Level of Detail (LOD) based simplification approach:* Level of details based approach are very popular in the graphics community [Lue01, LWCR02, ESV98]. Several approaches use bounding volumes hierarchies or spatial decompositions to simplify the models. These approaches approximate the objects with simplified bounding volumes or decompose the occupied space, to reduce the number of pairs of objects that need 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-

Table 2.4: *Summary of techniques based on dimension reduction operators*

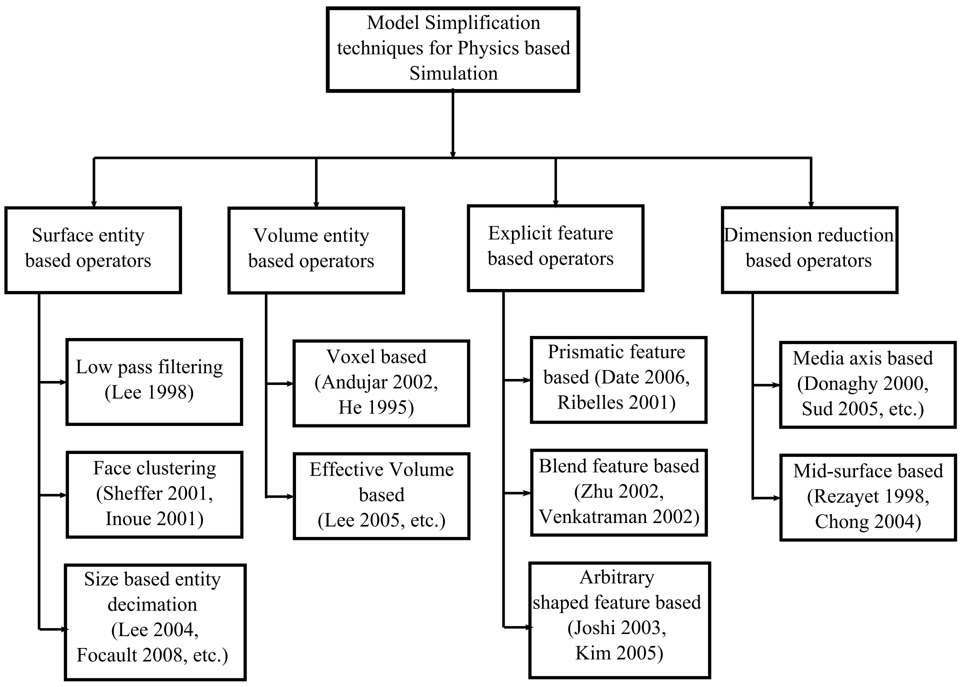
Method	Input format	Features simplified	Simplification criterion	Advantages	Limitations	Application domain
Medial axis transform [DAP00]	B-Rep	Shell based features	Automatic	High aspect ratio regions can be identified; simplified models computationally efficient and comparable with original model w.r.t. FEA results	Only applicable for shells	FEM model preparation
Homotopy preserving medial axis simplification [SFM05]	Polygonal Mesh	Freeform features	Automatic	Results into homotopically equivalent medial axis	Pruning of unstable parts of medial axis may not be optimal	Mesh generation and shape analysis
θ MAT based technique [FLM03]	Polygonal Mesh	Protrusions and depressions	Based on separation angle	Computationally stable	Homotopy is not preserved	Mesh generation; shape analysis
Mid-surface abstraction [Rez96]	B-Rep	Prismatic features	Automatic	Material distribution effectively represented; volume preserved even in reduced model	Complex geometries like freeform bumps and depressions not handled	FEM, fluid flow simulation and feature recognition
Feature decomposition and selective mid-surface abstraction [CKL04]	B-Rep	Freeform features	Interactive	Model decomposition also performed for efficient mixed dimension modeling	Mid-surface extension and stitching operations may lead to errors in generated mid-surface	FEM model preparation

DOPs [TCL99, Hub96, GLM96, Ber97, PG95, KHM+98]. Popular spatial decomposition techniques include: octrees, k-d trees, BSP-trees and Shell trees [JG97, KGL+98]. A general treatment of all the LOD based techniques as well as details on GPU based acceleration can be found in [Eri04].

- *Occlusion culling based approach:* The occlusion based approaches are based on the premise that there are some parts of objects never contacted by other objects in the process of physical interaction. For example, in case of visualization, back faces are occluded by the front face of the objects and thus could be removed while rendering. Similar approach is employed in real-time simulation scenario such as collision. Since, the faces having normals away from the relative velocity of colliding parts never takes part in collision, they can be culled. 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 [Van94]. Kumar presented a sub-linear algorithm for back-motion culling by using hierarchy and coherence [KMGL99]. Redon improved the back motion culling technique by employing hierarchical decomposition [RKC02].

2.1.16 Discussion

We studied existing model simplification techniques that are useful from physics-based simulation point of view and classified them broadly into four main categories based upon the type of simplification operators used in the respective techniques, *i.e.*, surface entity, volumetric entity, explicit feature and dimensional reduction.



Model Simplification techniques for Physics based Simulation
Surface entity based operators
Volume entity based operators
Explicit feature based operators
Dimension reduction based operators
Low pass filtering (Lee 1998)
Face clustering (Sheffer 2001, Inoue 2001)
Size based entity decimation (Lee 2004, Focault 2008, etc.)
Voxel based (Andujar 2002, He 1995)
Effective Volume based (Lee 2005, etc.)
Prismatic feature based (Date 2006, Ribelles 2001)
Blend feature based (Zhu 2002, Venkatraman 2002)
Arbitrary shaped feature based (Joshi 2003, Kim 2005)
Media axis based (Donaghy 2000, Sud 2005, etc.)
Mid-surface based (Rezayet 1998, Chong 2004)

Figure 2.8: *Taxonomy of model simplification techniques.*

Figure 2.8 shows the taxonomy of different model simplification techniques in the form of a tree containing four hierarchical levels. The solid lines denote direct inheritances of different techniques from parent classes.

Analysis of the four tables (Table 2.1 to Table 2.4) presented in the preceding sections clearly reveals that there is no single technique that can solve all the model simplification problems. For example, a majority of the techniques are applicable for FEA model preparation, whereas others are suitable for fluid flow problems, collision detection or even as pre-processing steps in recognizing complex features. Again, some of the techniques can effectively handle prismatic features, while others are useful in dealing with shell-based, freeform, cavity or protrusion type of features. The level of automation, input model format and the type of operators used vary quite a bit as well. All these factors, namely, application domain, types of features

Table 2.5: *Model simplification technique selection criteria*

Input format	Application Domain	Features present in the model	Level of automation/flexibility desired	Suitable simplification techniques
B-Rep	FEA model simplification, fluid flow problems	Prismatic	Automatic	Mid-surface [Rez96], Wraparound/thinning/smoothing operators [KLH+05], Detail Reduction for mesh generation [Tau01]
B-Rep	FEA model simplification, fluid flow problems	Prismatic	Interactive	Cell transformation based [LAPL05], Finite element based feature removal [DPS94]
B-Rep	Feature recognition preprocessing	Freeform features	Interactive	Face delete operators [VS02], Blend removal [VSR02], Fillet/Round simplification [ZM02]
B-Rep	FEA model simplification	Shell based features	Automatic	Medial-axis transform [DAP00, FLM03]
B-Rep	FEA model simplification	Free-form features	Interactive	Selective mid-surface abstraction [CKL04], Feature simplification [JD03], MCT based [FCF+08], Homotopy Preserving medial axis technique [SFM05]
Mesh	FEA model simplification	Cavity and protrusion type	Automatic	Feature removal [RHG+01], Feature simplification [DKKN06] Surface entity based operators [SBB97, She01, IIYF01, VL98, DKK+05] Local modification in a mesh [DSG97, SBO98]
Mesh	Collision detection	Protuberances and cavities	Automatic	Voxel based object simplification [HHK+95], TDPS [ABA02]
Mesh	FEM model simplification	Freeform features	Automatic	Effective volume [Lee05, H.05, LL06a, LLK06]
2D Image	FEM model simplification	Boundary edges and island type features	Automatic	Fourier transform based [LL98]
Native feature	FEM model simplification	Freeform	Automatic	Cellular topology based PSM [LLKK04]

handled, input format, level of automation and type of operators used, need to be taken into consideration before selecting a particular technique. Table 2.5 has been drawn to summarize our findings. We believe that this will aid potential users in choosing the appropriate technique quickly based upon the characteristics of their problem.

2.1.17 Summary

This chapter summarizes various model simplification techniques available in the open literature that are applicable for physics-based simulation applications. To the best of our knowledge, this is the first attempt towards classifying and organizing the various techniques into well-defined categories. Comparative study of all the techniques in each of the categories has been performed in order to delineate the types of features handled, relative merits, weaknesses, and potential applications clearly. Based on this classification and comparative study, we have also presented a broad selection criterion for different classes of engineering problems commonly encountered in practice. This literature survey clearly reveals that there are many open research issues that merit serious attention in the future. They are listed as follows:

- Lack of formal analysis of computational complexity: It is possible to identify certain methods to be computationally less intensive than others based on computational experiments on test data. However, very few of them have formally enumerated the complexity in terms of input parameters. In this case, the overall complexity can be decomposed into two categories: one pertaining to the recognition or identification of features and the other pertaining to the actual simplification. The former will be associated with the number of vertices or faces based on the type of solid model used (*e.g.*, mesh-based methods will relate it to vertices, whereas B-Rep-based methods will link it to the number of faces or surface patches). The latter on the other hand will deal directly with the number of features implicitly or explicitly represented in a particular method. Thus, formal derivation of asymptotic computational complexity is often a complex task. However, to really understand the nature of the underlying algorithms and make further progress, it will be very useful to know the asymptotic time complexity of the algorithms.

- Lack of application-specific (physics-dependent) error measures: A vast majority of methods utilizes indirect, geometry based error metrics to characterize their performance. These errors are used as thresholds to arrive at various decisions regarding classification and consequent simplification of individual features. However, these metrics usually cannot directly address the error that will be introduced in terms of the physical behavior and properties of the system under various possible external loading conditions. Hence, such metrics need to be developed using a mathematically rigorous framework to estimate the errors accurately and efficiently.

- Lack of standardized set of test parts: Graphics community has developed several standard test cases to assess the performance of any newly developed simplification technique for rendering (*e.g.*, Stanford bunny, sculpture of David by Michelangelo [Sta]). However, current the physics-based simulation community lacks a standard set of test parts to test simplification algorithm performance. So the research community needs to come up with a basic, test set of solid models that is acceptable to everyone working in this field.

- Lack of formal investigation of robustness: Robustness is an important issue in geometry computing applications. Discrepancies in floating point representation of geometric entities may result in erroneous results. Hence, often significant effort is devoted to ensuring robustness of the proposed algorithms. The problem of robustness has not yet received significant attention in model simplification applications.

2.2 Fluid-Rigid Body Interaction Simulation

Fluid-rigid body interaction simulations are computationally expensive because of the coupling between the fluid flow and the rigid body motion. There are two types of coupling, namely, (1) influence of rigid body on the fluid in which it moves, and (2) the influence of the fluid on the rigid body motion. Simulation approach in which both the couplings are solved explicitly in each time step are called two-way coupling solution; whereas, if one of the couplings is replaced with some faster model then it is called one-way coupling solution. In this section we shall review some common techniques for fluid-rigid body interaction simulation and existing approaches for simulating unmanned boat motion.

2.2.1 Techniques

Some of the common techniques, wherein the two-way coupling problem between a fluid and a floating rigid body simulation are: Euler's momentum equation, Navier-Stokes law, Smoothed Particle Hydrodynamics (SPH) technique, and Lattice Boltzmann Method (LBM).

- *Euler's Momentum Equation Based Approach :* In Euler's momentum equation based techniques, the momentum equation in continuum mechanics is solved for fluids numerically. Batty *et al.* reported a computation time of 25 s per frame using a grid size of 60×90 [BBB07].

- *Navier-Stokes law based Approaches :* In this class of techniques, interaction between a viscous fluid and rigid bodies are simulated by numerically solving the Navier-Stokes equations. Carlson *et al.* reported a computation time of 27.5 s per frame for a domain of size 64×64 [CMT04].

- *Smoothed Particle Hydrodynamics Based Approach :* In SPH technique, the fluid is assumed as a collection of particles and the motion of fluid particles and their effects on a floating rigid body is modeled based on a kernel function weighted by the distance of the particle from the floating rigid body. Becker *et al.* reported a computation time of 3.47 s per simulation step for simulating fluid with 850000 particles [BTT09].

- *Lattice Boltzmann Method:* In LBM, the fluid flow is represented as motion of fluid particles where each particle follows a velocity distribution function and moves in discrete time steps and can collide with other particles (which behave in the same way). The collision rules are such that the statistical particle motion (or fluid flow) obtained is consistent with the continuity conditions. Garcia *et al.* developed LBM based fluid-structure interaction approach [GGR11]. Geist *et al.* developed a real-time approach for wave surface generation and attained 25 frames/s for grid of size 1024^2 [GCTW10]. Geveler *et al.* developed LBM based approach for simulating laminar flow with free fluid surface on multi-core CPU and many-core GPU processors and reported a factor of 8 speedup on GPU code with respect to multi-threaded CPU code [GRGT11].

It was discussed in Chapter 1, that one of the main application of rigid body simulation is in VEs for robot simulations. VE for USSV is one of the systems that require fluid-rigid body interaction. A survey on boat simulation was reported by Beck and Reed [BR01]. Craighead *et al.* reported another recent survey on open source boat simulators [CMBG07]. Some of the key USSV simulation techniques are RANS based techniques, strip theory based techniques, kinematic model based techniques, and potential flow theory based techniques.

- *Reynold Averaged Navier Stokes Equation Based Approach:* In recent years, RANS based techniques for fluid flow around boats for simulating boat motion are becoming popular because of their accuracy in the problems involving boundary layer effects, turbulence, wake *etc.*, [Gor02]. The limitation of RANS code based techniques is the slow computation. In one of the implementations of RANS code by Kim, the reported computation time for 360 time steps is about 24 hours using 84 processors on Mauis IBM-SP3 computer [Kim02]. Some of the other implementations of RANS code can be found in [KGM$^+$03, WWS05, CWNS07].

- *3-DOF Model Based Approach:* There are host of research papers reported in the area of the underactuated controller design for the USSVs that utilize 3-DOF simplified models which neglect the rolling, pitching, and heaving motions [AMM10, KGZ97, LFP00, MPN02, DJP02, LPN03].

- *Strip Theory Based Approach:* The Strip theory is mainly used for slowly moving slender geometries [FS04b, BB08, HD06].

- *Potential Flow Theory based Approach:* In potential flow theory, fluid flow is assumed to be irrotational and inviscid [New77]. Potential flow theory based techniques are used by several researchers to perform the motion simulation of USSVs [KKF05, TG10, SS07].

2.2.2 Summary

In this section we reviewed some common fluid-rigid body interaction simulation techniques and also some boat simulation techniques. In nutshell:

- Euler's equation, Navier-Stokes, and RANS equation based techniques yield highly accurate results but one limitation is the dependence of computation time on the domain size and the slow speed of computation.

- SPH technique results into good quality animations but the problem with the SPH is requirement of large number of particles to simulate the fluid which in turn increases the computational time.

- The 3-DOF simulations are computationally very fast for the obvious reasons of neglecting the effect of fluid flow on the roll, pitch, and yaw and as a result of the same fact are not accurate enough.

- Strip theory based techniques are not suitable for taking non-linear effects due to the hull geometry and wave interactions. This is because, in strip theory, the hull geometry is approximated to the nearest ideal shape (such as ellipsoids, spheres, etc.). This idealization might yield significant errors in hydrodynamic and hydrostatic force estimations.

- The accuracy obtained by the potential flow based technique are not as good as the RANS but are computationally faster and much easily amenable to the 6-DOF computations and hence much accurate compared to the simplified 3-DOF models.

2.3 Physics-aware Robot Motion Planning

One of the main applications of rigid body simulation is found in robot simulation, which is used in robot motion planning. There is a large body of literature in robot motion planning and we do not make an attempt to summarize that [LaV06]. We shall mainly focus on research related to robot trajectory planning considering robot dynamics or in other words physics-aware robot trajectory planning.

2.3.1 Techniques

The literature for trajectory planning under differential constraints falls into five main categories [GKM10] namely, (1) state space sampling based trajectory planning, (2) decoupled trajectory planning with minimum distance path, (3) finite-state motion model or the maneuver automaton (MA), and (4) mathematical programming, and (5) Model Predictive Control (MPC).

- *State Space Sampling:* In this technique the robot state space is discretized and searched for the low cost collision free trajectory. Several schemes for state space discretization have been reported. In simple grid based approach, the state space is discretized into regular cells and trajectory is searched in the discretized state space [DXCR93]. In navigation function based approach, a navigation function is defined over the discretized state space and determined using using algorithms such as value iteration and then trajectory plan is determined based upon that. Interpolation is used [LK01a] to make the planning domain continuous. In rapidly exploring random trees (RRT) a stochastic search is performed in robot's body centered frame tree is expanded through random sampling in configuration [FDF01, LK01b, HKLR02].

- *Decoupled Approach :* In decoupled approach, planning takes place in two phases. In first phase, a discrete path or set of way points is determined using graph search technique on grid such as A*, Voronoi approach, probabilistic road map, etc. by considering only kinematic constraints of the robot. In the second phase, dynamically feasible trajectory is determined by solving two-point boundary value problem between consecutive way-points using optimization approaches. Suzuki *et al.* used A* approach for way-point generation and RTABU search based optimization approach for trajectory generation [Suz05]. Scherer *et al.* reported an evidence grid with a Laplacian-based potential method for path planning, an obstacle avoidance based on reactive planning, and velocity controller for trajectory generation [SSCS07].

- *Maneuver Automaton Approach :* In MA, the action space is discretized into action automatons to reduce the search from infinite dimensional space to finite dimensional. Frazzoli *et al.* presents rigorous definition of MA in [FDF99].

Some other related works can be found in Refs. [PKK09, KSA$^+$06, BLOY01].

- *Mathematical Programming Approach :* In this, trajectory planning problem is posed a numerical optimization problem with robot dynamics as constraints and solved using techniques such as mixed integer linear programming, nonlinear programming, and other constrained optimization techniques [RH02, BSRB06, ED05].
- *Model Predictive Control :* In this, the trajectory planning problem is posed again as optimization problem, but optimized over finite horizon. This way the solution obtained is suboptimal but takes lesser computation time than optimizing over infinite horizon [CVR09, HK07].

In presence of motion uncertainty, optimal trajectory planning can be done by solving a Markov Decision Process (MDP) using the dynamic programming (DP) algorithms [RN09]. However, since the state space of a planning problem under motion uncertainty is usually very large, most of the practical algorithms that have been developed [LGT04, FS04a, SGDS09] to compute an optimal or close-to-optimal solution to the problem by running the value iteration over a carefully chosen subset of the state space.

In the USSV trajectory planning domain a three layered architecture for Dijkstra algorithm based global planning and A* based local planning is presented by Casalino *et al.* [CTS09]. They used a simple kinematic model with no environmental disturbances. Benjamin *et al.* developed a technique for collision avoidance and navigation of the marine vehicles respecting the rules of the roads [BCN06].

Soltan *et al.* developed nonlinear sliding mode control based trajectory planner for a 3-DOF dynamics model [SAM09]. Xu *et al.* reported a receding horizon control based trajectory replanning approach where the global plan is determined using predetermined level sets from experimental runs [XKS09].

2.3.2 Summary

Most of the trajectory planning algorithms described above assumes deterministic environmental conditions or conservatively approximate uncertainties. A conservative approximation of motion uncertainty due to environmental disturbances interacting with vehicle dynamics might lead to sub-optimal plans. In order to incorporate motion uncertainty into the trajectory planning problem, MDP based framework is often used. State transition probability encodes the vehicle dynamics and environmental disturbance information into MDP formulation. In order to do physics-aware trajectory planning, one way to incorporate the physics information into the problem formulation is to use Maneuver Automatons and employ simulations to estimate state transition probabilities. Major challenge experienced in incorporating simulation based state transition map estimation is slower simulation speed due to the fluid-rigid body interaction computation requirements, which can be alleviated using model simplification techniques.

Chapter 3
Contact Preserving Model Simplification for Rigid Body Dynamics Simulations

Accuracy of contact locations among bodies during rigid body dynamics simulation influences the accuracy of the simulation directly. Often collision detection algorithms are used for determining the contact points between moving bodies. Many mechanical parts have a large number of features and hence collision detection with the detailed part models slows down the rigid body dynamics simulations. Hierarchical decomposition based techniques are used for accelerating collision queries. These techniques require polygonal models as input generated from the part models. A preprocessing step can be used to simplify the original part geometry by removing irrelevant detail. We refer to such preprocessing as model simplification in this chapter[2]. Model simplification techniques developed for efficient graphical rendering may change the part geometry in such a manner that the contact points between parts may change as a result of the simplification. Hence, such simplifications may alter the results of simulation. In many simulation scenarios, all the parts participating in the simulation are known in advance. In such cases, the simulation context (*i.e.*, a priori knowledge of parts) can be exploited to simplify the part geometries such that the contact points among parts do not change. For example, parts with significant concavities may have regions on their boundaries that will be inaccessible to other

[2] The contents of this section was published in ASME 2009 International Design Engineering Technical Conferences(IDETC) & Computers and Information in Engineering Conference (CIE)[TG09].

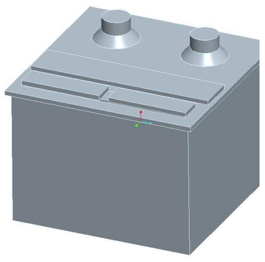
parts in the simulation and hence contact points cannot lie on such inaccessible regions. Removing such regions from the parts can simplify the model and hence speed up the simulation for interactive applications. In this chapter, we present an algorithm to simplify pair of interacting rigid body part models. We present a case study involving interactive simulation of unmanned boats to illustrate the benefits of the proposed approach.

3.1 Introduction

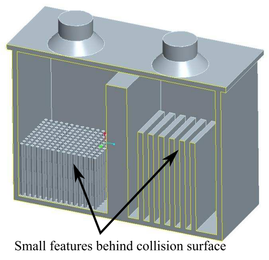
Rigid body dynamics simulations are nowadays used in a wide variety of interactive virtual environment based applications such as computer games, ground vehicle simulations, surgical simulation, and assembly process simulation. Many of the applications also need simulation of compliant parts in addition to rigid body simulations especially in health-care and manufacturing. Such compliance is often modeled using rigid bodies connected by springs (*i.e.*, pseudo-rigid bodies). Pseudo-rigid bodies are also simulated using the rigid body simulations. Thus, the rigid body dynamics simulation can be applied to a 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 the contact points among colliding rigid bodies at each simulation time step. Contact points play an important role in the computation of reaction forces due to the collision among rigid bodies. The reaction forces due to collision are then integrated to update the velocities and positions of the rigid bodies. Often collision detection algorithms are used for computing the contact points. Since the mechanical parts have large number of features, the collision detection using the detailed part models slows down the rigid body dynamics simulation. In order to reduce the computation time for collision detection, part model geometry needs to be simplified. Geometric model simplification involves elimination of features from the geometry that are redundant from the point of view of application. Simplification of geometric models without affecting the potential contact points can improve the performance of rigid body dynamics simulation without adversely influencing the accuracy of the simulation.

To simplify a geometric model, many techniques involving vertex, edge and facet decimation have been reported [TBG09, CMS98, LLP02, SBB97, She01, FCF+08]. Decimation based techniques have proved to be very useful for the applications like graphics rendering, finite element analysis model preparation, fast transmission of models over network, etc. 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 is 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. Hierarchical bounding volume decomposition is a highly successful model simplification technique that is used for accelerating collision queries to determine contact points accurately. We performed a test on a collision detection engine RAPID [GLM96], which is based on oriented bounding box (OBB) decomposition scheme, as shown in the Figure 3.1. The figure shows variation of collision computation time T versus the distance D between part model A and B. In the figure, model A is successively brought close to another part model B having the same geometry and for each relative separation distance D the collision computation time T is determined and plotted. A point to be noted about the the geometry of the parts A and B shown in Figure 3.1(b) is that there are small features just behind the surface of collision as shown in cut-through view in Figure 3.1(b). Figure 3.1(c) shows that the collision detection time is about $0.001s$ when the parts are far apart ($D \geq 7mm$). However, in close proximity ($D < 7mm$) the collision detection time increases rapidly. Henceforth, we'll refer to the case of part separation distance when the collision detection takes largest amount of time as the *worst* case. Figure 3.1(c) shows that the worst case collision computation time for the detailed geometry is about $0.038s$. The reason behind this dramatic increase in collision computation time is shown in Figure 3.1d, which shows that at farther distances the number of collision tests is smaller whereas in close proximity the number of collision tests increases. Some details of the part model are inaccessible in all possible relative pose of the parts and thus unnecessary from the point of view of rigid body collision. The inaccessible features can thus be removed without altering the potential contact points to simplify the part model. The worst case collision computation time obtained using such a simplified part model is $0.002s$ as shown in Figure 3.1(c). The drastic reduction in collision computation time by removing the inaccessible features can be explained as follows. Since the inaccessible details are spatially near to the point of collision, the hierarchical decomposition cannot take care of pruning the unnecessary facets during the lowest level collision query when the parts are in the close proximity. In other words, during the lowest level collision computations in the hierarchy, the inaccessible facets which are spatially very near to the point of collisions need to be tested for collision and increase the collision computation time. This causes increase in the collision computation time when the parts come very close to each other (as D reduces to zero) as shown in Figure 3.1(c). In many parts similar to the one shown in Figure 3.1, there are many details which are inaccessible but spatially very close to the point of collision. The inaccessible facets if removed off-line (*i.e.*, before the simulation), can reduce the computation time significantly without affecting the accuracy of the results.

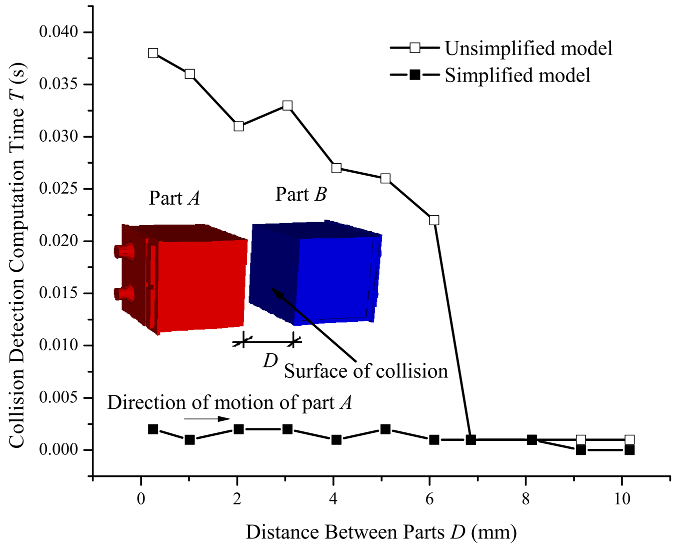
The potential contact points depend upon the collision context (*i.e.*, which parts are colliding). In many problems the collision context is known in advance or can be easily determined as the parts are known beforehand. This opens up a possibility of storing and retrieving multiple representations of parts based on the collision contexts, where each representation can be simplified for the given collision context. This scheme is promising as the memory is relatively inexpensive compared to the real-time computation of contact points for fully featured part models. We plan to utilize the collision context to generate physics-preserving simplified models. Our idea is 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 possible contact points are preserved using part accessibility considerations. In the Figure 3.2, the simplification process of part model A with respect to part model B is shown. Firstly, the inaccessible facets of part A with respect to part B are determined and then removed from A to generate simplified model A_s. We assume that the input models A and B are completely closed watertight shells, however, it should be noted that the resulting simplified model A_s is not necessarily a completely closed watertight



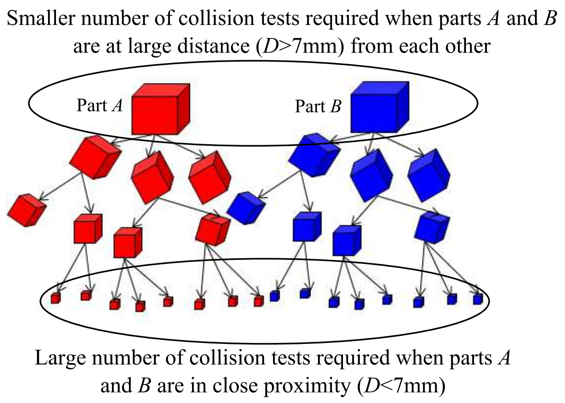
(a) *Part model.*



(b) *Cut-through view.*



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


Smaller number of collision tests required when parts A and B are at large distance (D>7mm) from each other
Part A
Part B
Large number of collision tests required when parts A and B are in close proximity (D<7mm)

(d) *Number of collision tests lesser when parts are far apart and more in close proximity.*

Figure 3.1: *Collision computation time drastically increases when parts are in close proximity (Test case for hierarchical decomposition performed on RAPID collision detection engine).*

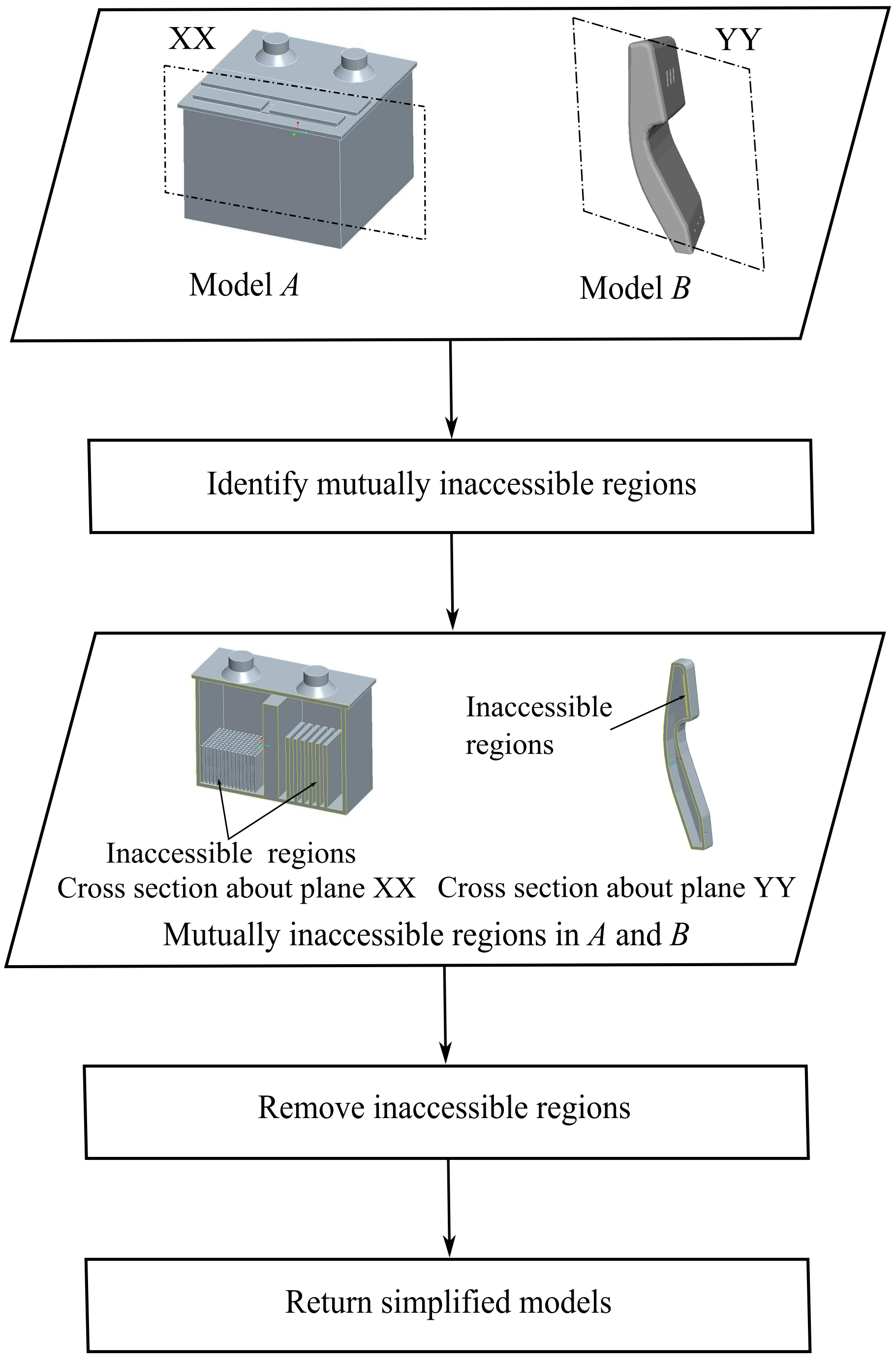

XX
YY
Model *A*
Model *B*
Identify mutually inaccessible regions
Inaccessible regions
Inaccessible regions
Cross section about plane XX
Cross section about plane YY
Mutually inaccessible regions in *A* and *B*
Remove inaccessible regions
Return simplified models

Figure 3.2: *Context dependent contact preserving model simplification process.*

shell. Such models are acceptable for the simulation purposes; as such simulations need only polygon soup data to perform collision queries. In Section 3.2, we present the problem statement and overview of the model simplification approach to solve the problem. Section 3.3 to 3.5 discusses various steps in the model simplification algorithm. In Section 3.6 we discuss the implementation details and the results and finally conclusions are presented in Section 3.7.

3.2 Problem Statement and Solution Approach

3.2.1 Problem Statement

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

Definition 1 Let A, B be polyhedral representation of a pair of parts. We define a *non-penetrative touch transforms* denoted by T, such that,

(i.) T is a rigid body transformation

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

where,

$$\text{touch}(X,Y) = \begin{cases} \text{true}, \text{int}(X) \cap \text{int}(Y) = \phi \text{ and } \text{bnd}(X) \cap \text{bnd}(Y) \neq \phi \\ \text{false}, \text{ otherwise} \end{cases} \tag{3.1}$$

and,

$$\text{penetration}(X,Y) = \begin{cases} \text{true}, \text{int}(X) \cap \text{int}(Y) \neq \phi \\ \text{false}, \text{ otherwise} \end{cases} \tag{3.2}$$

int(.) is the interior of the point set, bnd(.) is the boundary of the point set and ϕ is the null set. Henceforth we will refer to the polyhedral representation of a part as the *model* of the part. 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 is computed in the following sections.

Definition 2 For a given pair of models A and B, the set of *inaccessible facets* of A with respect to B is denoted by $A_{in,B}$ and defined as a set of facets for which touch$(A_{in,B}, TB)$ = false, for all possible non-penetrative touch transforms T.

In the Figure 3.2, the inaccessible facets of the model A with respect to model B are highlighted.

Problem Statement: Given a pair of part 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 $B_s = B - B_{in,A}$.

3.2.2 Overview of the Approach

In order to make contact with any facet f on the concave portion of part A, part B will need to access facet f via some passage or opening on part A (please refer to Figure 3.2). These passages and openings impose restrictions on the size of B that can reach f (*e.g.*, if B is very big compared to the available passage and opening, then it cannot reach f). Section 3.3 presents a formal approach to automatically identify and represent the concave regions, openings, and passages.

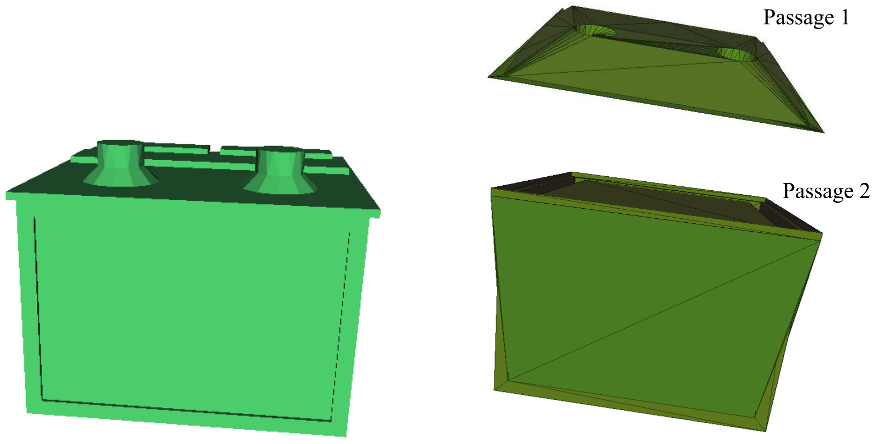
We are interested in knowing how deep B can enter into the passage of A. Usually parts with large cross-sections with respect to the passage can only enter the passage up to a certain distance before the cross-section becomes the bottleneck. Parts have different cross-sections in different directions. 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 call this measure, part section signature. Section 3.4 describes algorithms for computing the part section signature.

Finally, given a passage on A and part section signature of B, we analyze if B is small enough to enter the passage of A and reach f or not. Facets on the concave portion of A that cannot be reached by B due to the large size of B with respect to available passages and openings on A can be removed during the contact determination step. Section 3.5 describes the algorithm to determine and remove the list of inaccessible regions of the part models to obtain simplified models.

3.3 Determining Passages

We first define the notion of the passage set and passage, 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 3-cell complexes $p_{A,i}$, such that, $p_{A,i}$ belongs to the interior of the convex hull of A



(a) *3D model for part A.* **(b)** *Passages of part A.*

Figure 3.3: *Part model and passage set (passages of A are obtained by subtracting the A from from the convex hull of A).*

but does not belong to the interior of A.

$$P(A) = A_c - A \tag{3.3}$$

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.

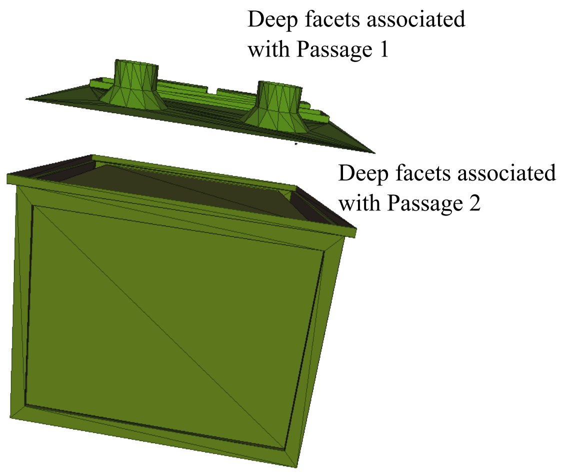
The passage set of the model shown in Figure 3.3 are illustrated in Figure 3.3(b).

Definition 4 A *deep facet* F_D of a model is defined as a facet that completely lies in the interior of the convex hull of the model.

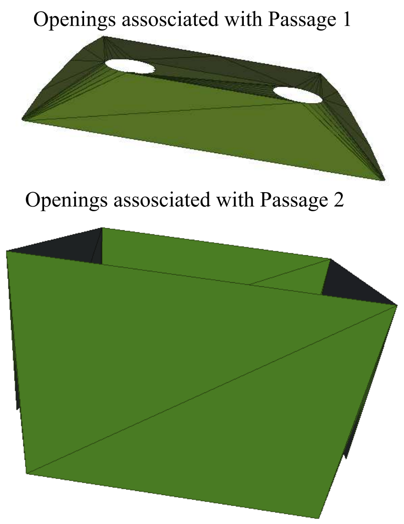
The deep facets of the model shown in Figure 3.3(a) are illustrated in Figure 3.4(a).

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

(i.) Ω is a connected set, and


Deep facets associated with Passage 1
Deep facets associated with Passage 2

(a) *Deep facets of part A corresponding to each passage shown in Figure 3.3.*


Openings assosciated with Passage 1
Openings assosciated with Passage 2

(b) *Openings of part A corresponding to each passage shown in Figure 3.3.*

Figure 3.4: *Deep facets and openings of part A.*

(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. The openings are actually computed by determining the triangulation of the virtual face which lies on the convex hull of the model but not on the part and procedure for the same is explained in this section in step 2.

The openings of model A is shown in Figure 3.4(b).

Definition 6 An opening Ω and a deep facet F_D belonging to a passage $p_{A,i}$ are said to be connected if there exists at least one pair of points $p_m \in \Omega$ and $p_n \in F_D$, such that the line segment L connecting p_m and p_n satisfies following,

(i.) All the 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 $p_{A,i}$.

The shortest distance between a deep facet and connected opening, *i.e.*, the length of the segment L is called as the *opening distance*.

The largest sphere that can be inscribed inside the volume of $p_{A,i}$ such that it contains at least one point lying on segment L, is called as *opening sphere* and the radius of the opening sphere is called as *opening radius*.

Some of the 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 (a very large number in computer program). Opening sphere is determined for such orphan facet 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.

The following steps are used to determine the passage set, deep facets, and openings of given model and represent the model as a graph.

Algorithm 3.1 - Connectivity Graph

Input Polyhedral representation of part A.

Output Connectivity graph representation of part 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 part shown in Figure 3.3(a) is shown in the Figure 3.3(b). We express A and A_c as Nef-polyhedron as implemented in CGAL [cga, HK05].

(ii.) Find the 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 the bounding facets of A but do lie on the bounding facets of A_c as opening facets. Group all the 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 [cga]. The openings of part shown in Figure 3.3(a) is depicted in Figure 3.4(b).

(iii.) Find the deep facets of A: Iterate through the bounding facets of model A and find all the facets of A that are coincident with the bounding facets (that are not openings) of the passage $p_{A,i}$, and mark them as the deep facets. The deep facets of part shown in Figure 3.3(a) are shown in Figure 3.4.

(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 the previous steps. Determine the connected opening nodes for each deep node and add to the connectivity graph. The nodes represent the deep facets or the openings while the arc shows the connectivity. Iterate through the nodes representing the deep facets and determine the bounding sphere for each connected opening and store it as the opening sphere of the connected deep facets. In order to simplify the computations we used the fact that the bounding sphere of the 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 opening and store as opening distance for the deep facet. If some deep facet is not connected to any opening then use the 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, we set the opening distance to a large value (we used 10^{10}).

(v.) Return connectivity graph obtained in step (iv.).

3.4 Construction of Part Section Signature

A deep facet on part A is said to be accessible by another part B, if there exists at least one relative orientation of A and B such that B *touches* the deep facet on A and does not *penetrate* into A. In order to answer the question that whether a given deep facet on A is *accessible* by the given part B, we need to represent part B as a function of *minimum opening distance* of the 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 distances of deep facets of A returning lower bound on the size of B as part section signature or PSS of B.

Definition 7 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 the following conditions:

(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 8 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 the following conditions:

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

Definition 10 For a given model M, direction vector $\vec{d}$ and a distance D, $(0 \leq D \leq R)$, we define a plane P_S as the *slicing plane*, if it satisfies following conditions:

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

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

Definition 11 For a given model M, a direction vector $\vec{d}$ and a depth D, $(0 \leq D)$, 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 the 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.

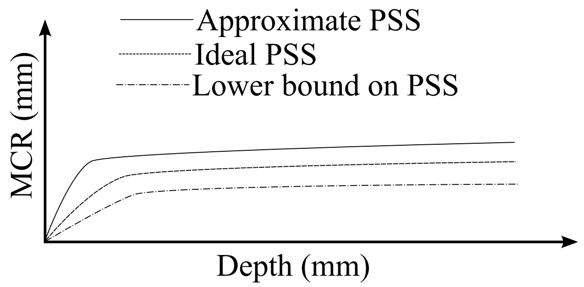
Definition 12 For a given model M, we define *part section signature* (PSS) of the part as,

$$\text{PSS}(M, D) = \min_{\vec{d}} \text{MCR}(M, \vec{d}, D) \tag{3.4}$$

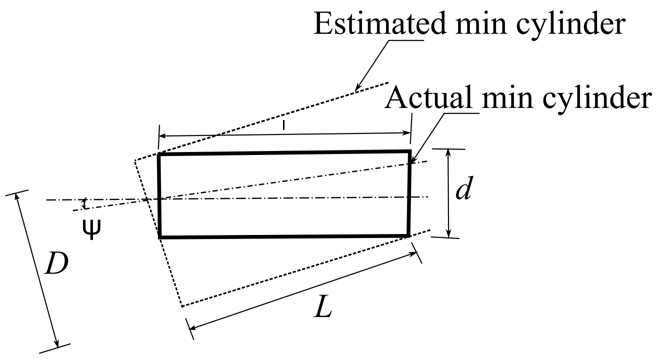
where, $\vec{d}$ is the direction of approach and D is distance between the first plane and the slicing plane for direction $\vec{d}$.

In other words, PSS is a 2D representation of part model M that gives the relationship between a given depth D of the 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 can be noted, that PSS is a monotonically non-decreasing function.

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



(a) *Comparison of approximate and theoretical PSS.*



(b) *Error estimation.*

Figure 3.5: *PSS error estimation.*

(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 the 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 the part can enter an opening to a specified depth. If it is found that the part cannot enter the opening to the given depth, the *connected* deep facets are removed to simplify the model, details of which are given in the next section. We are interested in conservative simplification (*i.e.*, we do not want to change contact points due to approximations in PSS). Hence, we computed an estimated lower bound on the theoretical PSS and used it as the conservative estimate of the theoretical PSS. The approximate, theoretical and the estimated lower bound on the PSS is shown schematically in the Figure 3.5(a). The estimated lower bound on PSS of a part guarantees that the part cannot enter a given hole if the theoretical PSS of the part does not do so. This approach ensures that when we delete a facet, the facet is truly inaccessible, however, sometimes a facet may be inaccessible but our approach will not be able to eliminate them 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 the set of directions we determine the MCR at various depths and then find out the approximate PSS. To estimate how much the actual PSS can be lesser than the computed approximate PSS, consider Figure 3.5(b). The ideal cylinder (with diameter d) enveloping the part to a depth of l is shown by the rectangle drawn in solid line. The estimated minimum cylinder (with diameter D) enveloping the part is shown by broken line, 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 $tan(\psi) \approx sin(\psi) \approx \psi$,

$$\Rightarrow \tfrac{d}{D} \approx 1 - \tfrac{l}{D}\psi,$$

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

thus,

$$d \geq D\{1 - \frac{l}{D}\alpha\} \tag{3.5}$$

We use equation 3.5 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.) Smaller the value of α, e_f is closer to unity. This means that estimated cylinder diameter D is close to actual cylinder diameter d. However, computation time of PSS increases at inverse square rate of decreasing α and thus a choice of tradeoff between better estimation and computation time can be made by selecting appropriate α.

The following algorithm is used to compute the lower bound on the PSS of part B.

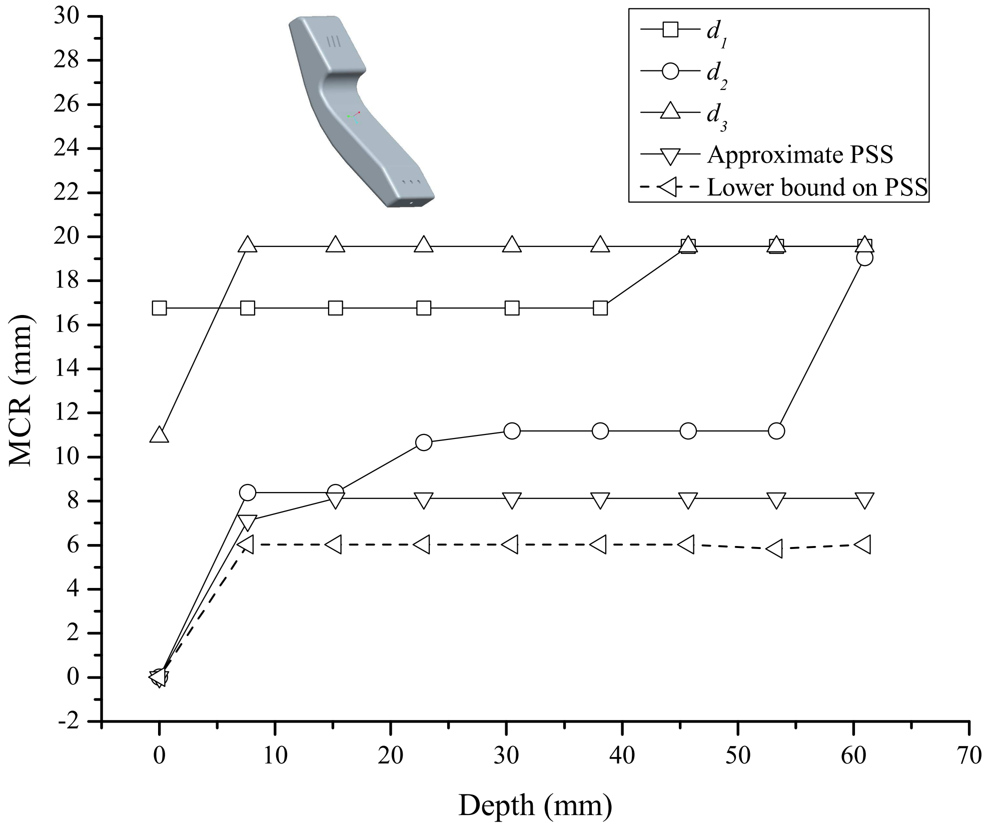


Figure 3.6: *Part section signature.*

Algorithm 3.2 - Determine Part Section Signature

Input

(a.) Polyhedral representation of part B, and

(b.) angular sampling increment α.

Output PSS of part B

Steps

(i.) Find the list of direction vectors: Determine the 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.

(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, using the list of depths of deep facets on the other part (*i.e.*, model A) generate the slicing planes normal to the direction vector and add to the slicing plane set. If the depth for a direction is larger than the *range* in that direction, the depth is set as equal to range.

(iii.) Compute the cross-sections along each direction vector: Intersect the model with each plane in the slicing plane set to obtain the point sets lying in the negative half spaces of each of the slicing planes. After this, determine the MCR of each of the point sets. In order to eliminate redundant computations, only compute the cross-sections for the range of depths of deep facets obtained in the previous section. For each candidate direction store the MCR variation along the respective directions. Figure 3.6 shows variation of MCR along three directions ($\vec{d_1}$, $\vec{d_2}$, and $\vec{d_3}$) for the part model shown in the figure.

(iv.) Compute the 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. For the part shown in Figure 3.6, variation of MCR along three directions of approach and the approximate PSS of the part are shown in Figure 3.6. We have shown only three representative directions $\vec{d_1}, \vec{d_2}$ and $\vec{d_3}$, in Figure 3.6, for the sake of brevity.

(v.) Apply conservative bound on the approximate PSS: Multiply factor e_f given in Equation 3.5 to approximate PSS and determine the lower bound on the PSS. The estimated lower bound on the PSS is shown by broken lines in the Figure 3.6.

3.5 Model Simplification

In this step, each deep facet of the connectivity graph of passages of model A is traversed to find out which of them are inaccessible through their corresponding connected openings by the PSS of the model B. The inaccessible deep facets of A by B and are subsequently deleted to generate the simplified model A_s. The steps followed are listed below.

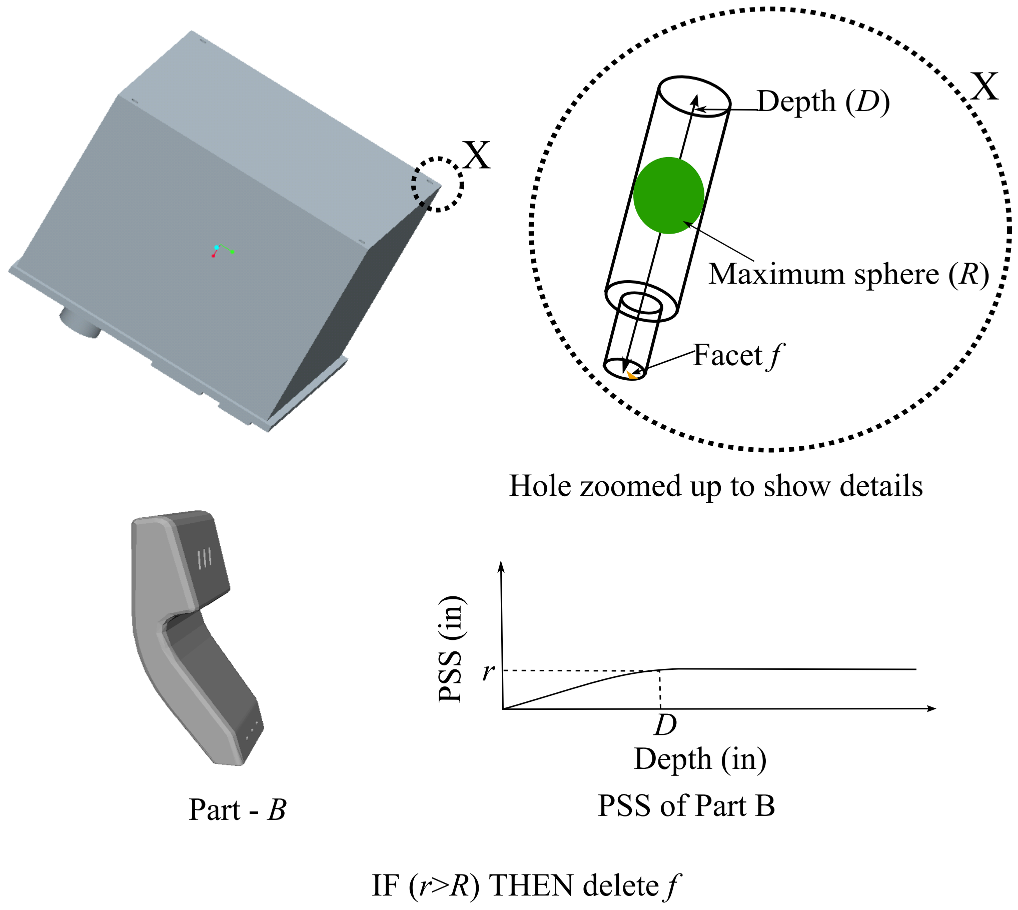


Figure 3.7: *Model simplification procedure (inaccessible facets are detected and then removed).*

Algorithm 3.3 - Model Simplification

Input

(a.) Connectivity graph representation of parts A and B, and

(b.) PSS of parts A and B.

Output Simplified models A_s and B_s.

Steps

(i.) Find the inaccessible facets of A with respect to B: Iterate through the openings corresponding to every deep facet of A and determine the smallest opening distance and the largest opening sphere. The smallest opening distance D is used to determine the PSS (r) of B at depth equal to D and the r is compared with the largest opening sphere radius (R). If radius (R) of the largest opening sphere of a deep facet is smaller than the PSS (r) of B at depth D for the smallest opening distance, the deep facet is considered inaccessible (see Theorem 1).

The procedure for determining accessibility of a facet f of A with respect to B is shown in Figure 3.7. The PSS of the part B at depth D is shown in the Figure 3.7 as r and the minimum distance from the highlighted facet f to corresponding opening is shown as D. The largest sphere inscribed in the passage of model A corresponding to the facet f has the radius R. To find whether facet f in A is accessible by the part B or not, R is compared with r. If r is larger than the R, then it can be concluded that B cannot touch facet f of A and hence f can be labeled as inaccessible by B.

(ii.) Remove the 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 in this step.

(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 radii is r_{max}.*

If $PSS(B, lp_{min}) > r_{max}$, where B is a part model, then B can never touch F_D.

Proof. We can prove this by contradiction.

Let us suppose that the part B can touch F_D for a non-penetrative touch transform T through an opening Ω_j with of opening distance lp_j.

Let the opening radius corresponding to opening Ω_j is r_j.

Let, transform T orient the part 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,

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

$$\Rightarrow MCR(B, \vec{d}, lp_{min}) \leq r_{max} \tag{3.6}$$

Now, from the given conditions in lemma,

$PSS(B, lp_{min}) > r_{max}$

and Equation 3.4,

$PSS(B, lp_{min}) \leq MCR(B, \vec{d}, lp_{min})$, for any arbitrary direction $\vec{d}$,

we have,

$$MCR(B, \vec{d}, lp_{min}) > r_{max} \tag{3.7}$$

Inequalities 3.6 and 3.7 contradict each other and hence under the given conditions, part B can never touch the deep facet F_D.

Hence the proof follows. □

3.6 Implementation, Testing and Results

We developed a prototype software implementation of the contact preserving model simplification algorithm discussed in this chapter. The programming platform was chosen as VC++ (version 8) using CGAL 3.6 on Windows Vista operating system [cga]. Our implementation takes STL files as input and after simplification, generates output in the same format. As a platform to test the simplified models obtained by using our approach, we used a framework that we have been developing for simulating surveillance operation for unmanned surface vehicles (USSV) [SSTG09]. The virtual environment simulates a maritime mission where a remote controlled boat tries to protect a valuable target from an intruding boat.

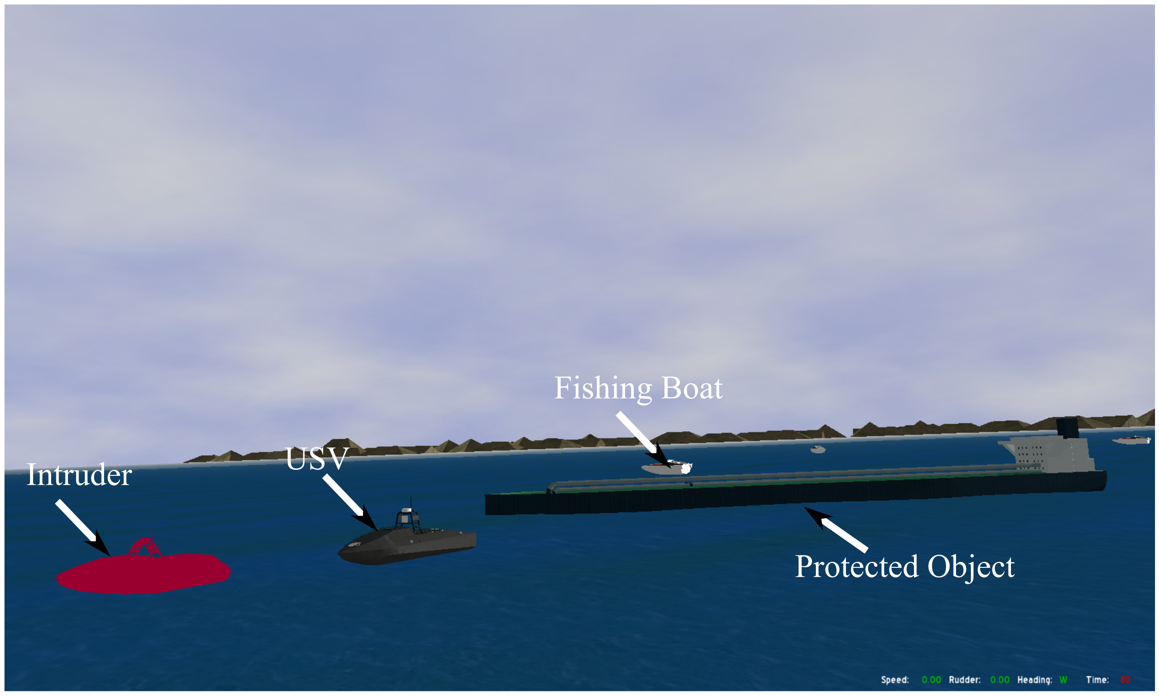


Figure 3.8: *Screen-shot of the virtual environment.*

Figure 3.9 shows the schematic view of the mission where, the intruder boat attempts to reach the protected object by crossing the buffer zone and the danger zone. The USSV's job is to block the intruder. The level of aggression with which, USSV blocks the intruder depends upon the position of the intruder (*i.e.*, whether the intruder lies in buffer zone or the danger zone). The USSV should block the intruder and at the same time should avoid collision with the other dynamic obstacles, which represent the harmless artifacts such as fishing boats. An important maneuver in this operation is the active blocking of the intruder boat by the remotely controlled USSV by steering towards to the front of the intruder boat in the danger zone. This application requires interactive rigid body dynamics simulation to train operators driving the boats using tele-operation. A meta-model is used to determine

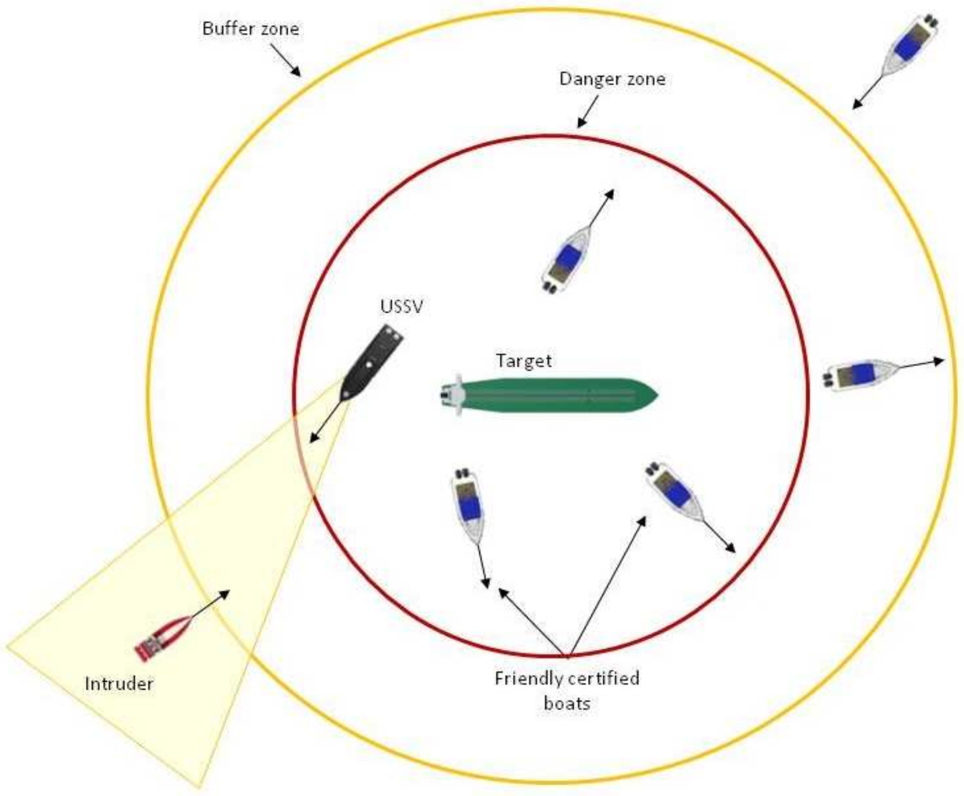
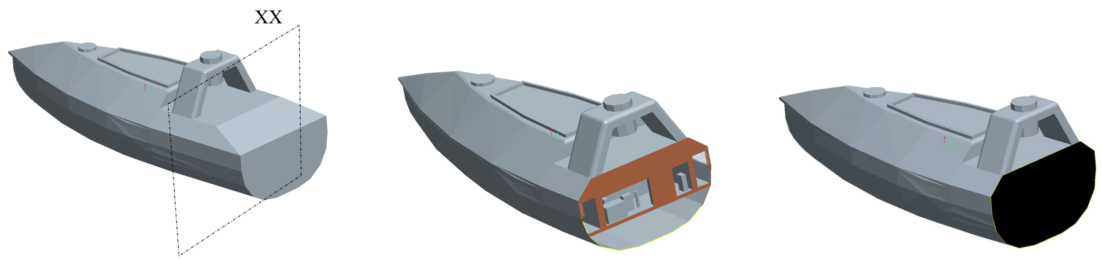


Figure 3.9: *Representative simplification results.*

the forces (such as ocean wave, gravity, buoyancy, etc.) acting on the boats due to the water. A screen-shot of the virtual environment is shown in Figure 3.8. We chose RAPID collision detection engine for collision queries [GLM96]. In our test, we imported a boat model (with 60030 facets) to represent both the USSV and the intruder and used this detailed model for the purpose of visualization as well as the collision detection. The boats were slowly moved towards each other and the time taken for collision query in each time step was recorded. The maximum collision detection time obtained with the detailed model (with 60030 facets) used as both the visualization and collision model was found to be 0.240 s. Using our model simplification approach, the simplified boat model obtained had 1120 facets (as shown in Figure 3.10). The features such as ribs, inner parts such as motors, batteries,



(a) *USSV model with* 60030 *facets.*

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

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

Figure 3.10: *Boat model used in the virtual environment.*

etc. are removed automatically using the model simplification approach as shown in Figure 3.10(c). It should be noted that only the ribs lying just behind the collision surface contribute significantly to collision detection time. Other features that are very far from the point of collision have negligible effect on the collision detection time.

Using the simplified model (with 1120 facets) as collision model and the detailed model (with 60030 facets) as visualization model we repeated the simulation. The maximum collision detection time obtained with the simplified model as the collision model was found to be 0.003 s. The contact points obtained in both the cases (with unsimplified and simplified collision models) were *exactly* same. The variation of collision detection time with respect to distance between the boats is shown in Figure 3.11. The speedup in the worst case collision detection time by using simplified models was thus by a factor of 80.0.

We also tested our implementation on several different pairs of parts that have features close to the inside of the wall. The parts are shown in Figure 3.12.

Representative results of model simplification are shown in Table 3.1. The

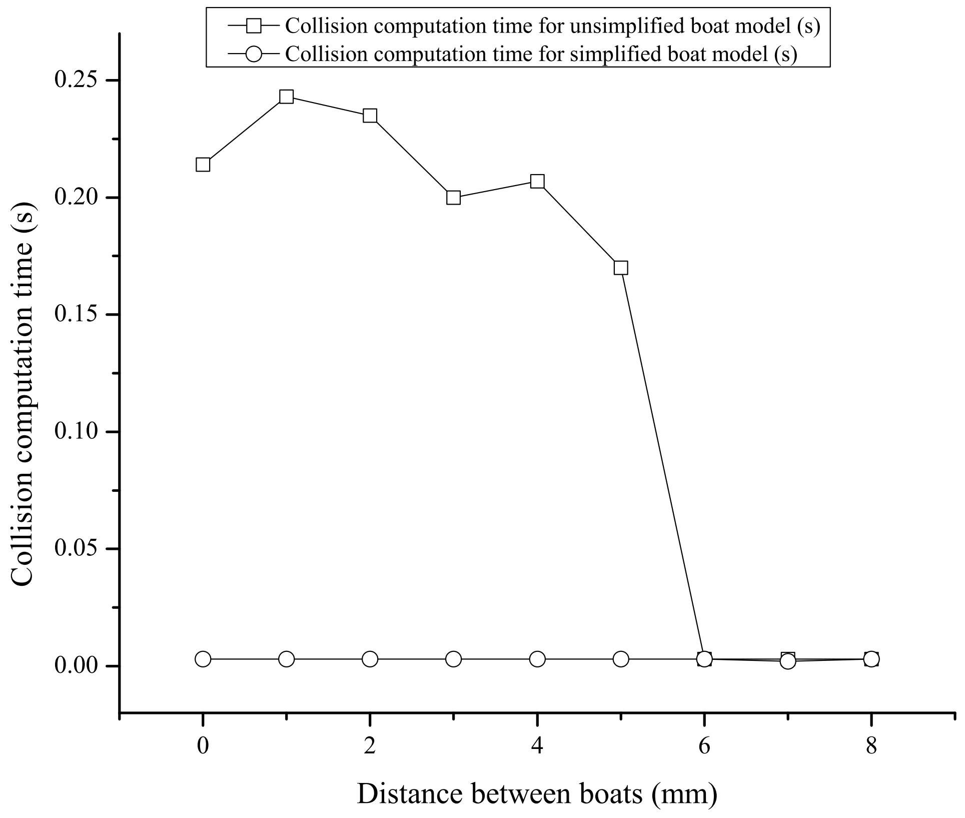


Figure 3.11: *Variation of collision detection time with the proximity of USSVs. When the USSVs are far apart (D¿6mm) collision computation time is very small because of the efficient hierarchical data-structure used by the existing collision detection approaches but in close proximity ($D < 6mm$) model simplification technique developed in this paper in conjunction with hierarchical data-structure reduces collision computation time by a factor of* 80.0.

Table 3.1: *Representative simplification results*

Model A	Model B	Unsimplified facet count		Simplified facet count		Facets simplified by factor of		Worst case collision detection time (s)		
		Model A	Model B	Model A	Model B	Model A	Model B	Unsimp. pair	Simp. pair	Speedup
Part 1	Part 2	34692	8448	667	228	52.00	37.10	0.080	0.001	80.00
Part 1	Part 3	34692	13156	667	2687	52.00	4.90	0.209	0.001	209.00
Part 3	Part 2	13156	8448	504	228	26.10	37.10	0.012	0.001	12.00
Part 5	Part 4	30768	12836	1356	752	22.70	17.10	0.005	0.004	1.20
Part 5	Part 3	30768	13156	1356	2687	22.70	4.90	0.048	0.005	9.60

figure shows the reduction in the facet count of the 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 4.9 to 52.0 depending

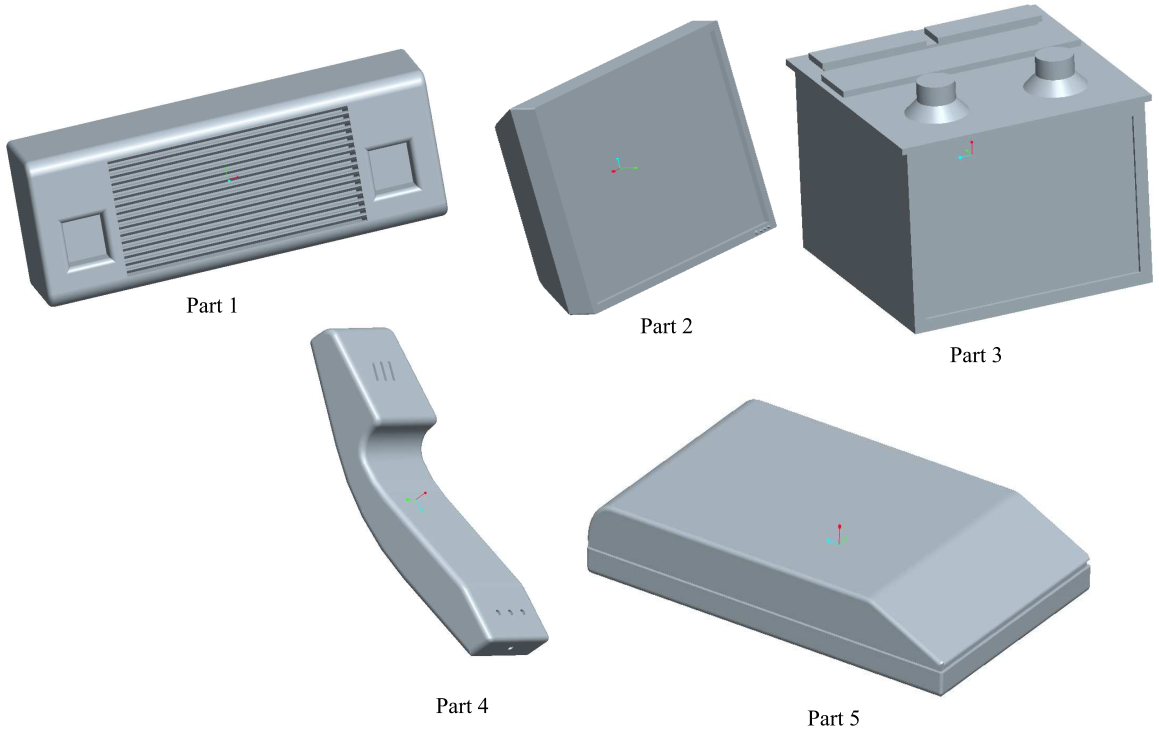


Figure 3.12: *Example part models.*

on the part complexity and the collision context. We repeated the test described in Figure 3.1 for each of the part pairs in Table 3.1. The reduction in worst case collision detection time by using the simplified models is shown in Table 3.1. The collision detection time reduced by factors in the range of 1.25 to 209 by utilizing simplified models over the unsimplified models. We ran our model simplification implementation on a computer with Quad-core processor and 8 Gigabytes of RAM and it took 456.2 s to simplify the five pairs of example models simultaneously. So, on an average, for the parts of the complexity level in the figures, one model took about 45.6s for the simplification.

3.7 Summary

We introduced the concept of context dependent contact preserving model simplification by using part accessibility considerations for accelerating the collision queries in the rigid body dynamics simulation. The accuracy of rigid body simulation depends upon the accuracy of contact points generated by the underlying collision detection engine. Our off-line approach gets rid of the facets not playing any role in collision while retaining all the facets which might possibly affect the results of collision query. Our main contributions are as follows:

(i.) We developed a conservative 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 developed and implemented a new off-line algorithm for contact preserving model simplification for rigid body dynamics simulation.

(iv.) We demonstrated reduction in the collision detection time obtained by using simplified models using our approach in the USSV simulation environment and several example part pairs.

Since the approach helps to reduce collision detection time without altering the potential contact points, the possible applications besides the rigid body simulation includes robot motion planning, assembly simulation, etc.

One limitation of this work is the conservative approximation of approaching parts with bounding cylinders that sets a lower bound on the part cross-section, which can simplify the other parts relative to the approaching part but might leave some facets which should be removed. This way, however, we guarantee that the part is never over-simplified. If applied in a purely combinatorial manner, the proposed approach will require $n - 1$ simplified models generated for every part in a scene consisting of n parts. In order to tackle with this problem we have developed a greedy approach to combine simplification opportunities from all relevant part pair interactions and generate a limited number of simplified models to meet the memory constraints and exploit the simplification opportunities.

Chapter 4
A Computational Framework for Real-time Unmanned Sea Surface Vehicle Motion Simulation

VE for unmanned sea surface vehicles (USSV) requires a six degree of freedom dynamics simulation in the time domain. In order to be interactive, the VE requires real-time performance of the underlying dynamics simulator. In general, the dynamics simulation of USSVs involves the following four main operations: (1) computation of dynamic pressure head due to fluid flow around the hull under the ocean wave, (2) computation of wet surface, (3) computing the surface integral of the dynamic pressure head over the wet surface, and (4) solving the rigid body dynamics equation. The first three operations depend upon the boat geometry complexity and need to be performed at each time step, making the simulation run very slow. In this chapter[3], we address the problem of physics-preserving model simplification for real-time potential flow based simulator for a USSV in the time domain, with an arbitrary hull geometry. The chapter reports model simplification algorithms based on clustering, temporal coherence and hardware acceleration using parallel computing on multiple cores to obtain real-time simulation performance for the developed VE. The average computation time was reduced by a factor of about 28.50 introducing an average error of about 5.8% with respect to the baseline computations, using the simplification techniques described in this chapter.

[3] The contents of this chapter was published in ASME 2010 International Design Engineering Technical Conferences(IDETC) & Computers and Information in Engineering Conference (CIE) in [TG10].

4.1 Introduction

Unmanned vehicles have emerged as an important tool in search, rescue, recovery, and surveillance operations. VE play a significant role in the effective design and operation of unmanned vehicles. First, VEs are useful for training operators that tele-operate unmanned vehicles. Virtual environment-based training significantly reduces the training cost [GAB^{+}08] and enables training of a large number of operators without requiring significant time on the physical platforms. Second, programming by demonstration has emerged as an important paradigm for acquiring autonomous behavior [ACVB09]. Virtual environments are expected to play a very important role in the deployment of programming by demonstration paradigm. Third, VEs can be a very valuable tool for hardware and software design of unmanned vehicles. For example, new designs can be first prototyped in the virtual environment.

In the chapter we are focusing on the development of a VE for simulating USSVs. This environment will be used for simulating the motion of USSVs under different sea states to facilitate operator training, design, planning, and control of the USSVs [SSTG09]. Figure 4.1 shows a snap-shot of the VE developed by [SSTG09, GAT^{+}10]. The VE simulates a typical maritime mission where the USSV's goal is to protect the oil tanker from the approaching intruder with malicious intent. For example, in the Figure 4.1, the operator can drive the USSV and demonstrate the strategies to block the intruder from reaching the protected object. The efficacy of the planning and control strategy obtained from any of the algorithms significantly depends upon the accuracy and computational speed with which the motion of the boats moving in the ocean can be predicted by the simulation. The problem of motion simulation of boats under the influence of the ocean waves is known as sea-keeping. Sea-keeping is inherently a computationally intensive operation, as it is an instance of the rigid-fluid two-way coupling problem.

One way to solve the sea-keeping problem is to solve it as a two-way coupling problem. The two-way coupling can be understood as the simultaneous effect of fluid flow on the motion of a floating USSV and the effect of the moving USSV on the surrounding fluid flow. Computational fluid dynamics (CFD) based approaches are very accurate in solving such problems. Unfortunately, the speed of computation using CFD approaches is extremely slow and there is hardly any hope to perform CFD simulation in real-time using the current computing technology.

Another approach used for sea-keeping simulation is the strip-theory based computations, wherein the boat geometry is idealized as the nearest regular geometry such as ellipsoid, cylinder, etc. The approximated formulas for computing the hydrostatic and hydrodynamic forces using the idealized geometries are then used. The strip theory based approaches are very fast but introduce significant errors. Moreover the technique is insensitive to variations in hull geometries. In other words, various hull geometries can be idealized into the same regular geometry leading to same force acting on them.

Potential flow theory based approaches imposes assumptions of irrotational, inviscid, and incompressible flow. Due to the assumptions, the Laplacian of the velocity potential becomes zero. The kinematic boundary condition involving the movement of fluid on the free surface and the dynamic boundary condition involving the pressure acting on the free surface is used to solve the partial differential equation. The solution yields velocity potential, which in conjunction with the Bernoullis law is used to compute the pressure due to the fluid flow. The pressure is then used to compute the force acting on the floating boat. The effects of boat on fluid flow are modeled as the added mass. The turbulence and viscosity related effects are modeled by the damping matrix. This leads to a much faster although a less accurate approach compared to CFD based approaches. The potential flow based approach is much more accurate compared to the classical strip theory based approaches. In most of the applications where a moderate accuracy is desired at a higher computational speed, the potential flow theory based approaches prove to be the acceptable choice for the sea-keeping simulation.

Although, potential flow based approach is faster than the CFD based approach, it is far from real-time. The reason for slow computation of USSV motion simulation can be explained as follows. The motion simulation using potential flow theory involves the following four main operations: (1) computation of the dynamic pressure head due to the fluid flow around the hull under the influence of the ocean wave, (2) computation of the instantaneous wet surface, (3) computation of the surface integral of the dynamic pressure head over the wet surface, and (4) solving the rigid body dynamics equation. First three operations depend upon the boat geometry complexity and need to be performed at each time step, making the simulation run very slow. To obtain the real-time performance, in this chapter we describe simplification algorithms based on clustering, parallelization, and temporal coherence.

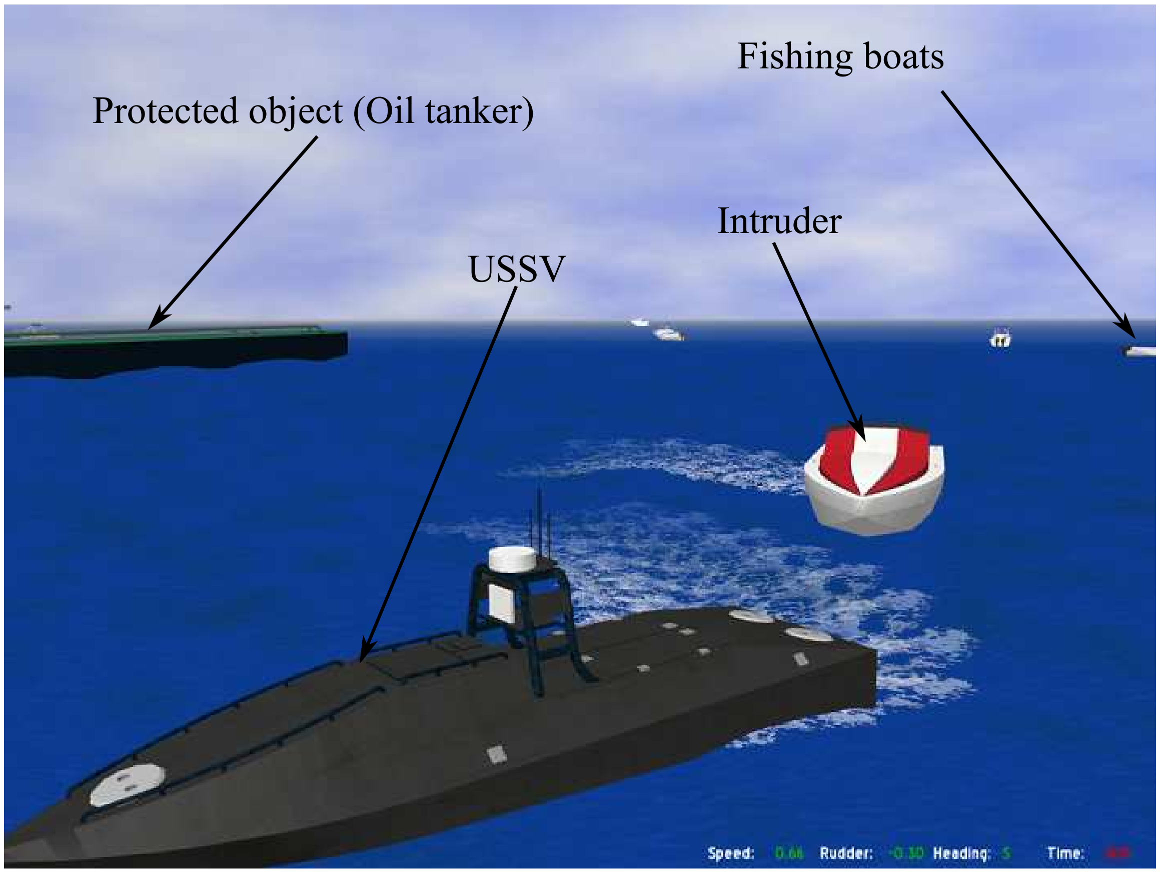


Figure 4.1: *USSV Virtual Environment developed by [SSTG09, GAT+10].*

These algorithms have been incorporated into the USSV virtual simulation system as shown in Figure 4.1.

The chapter is organized as follows. In Section 5.3 we present the governing equations of the fluid flow and rigid body dynamics model. Section 5.2 presents the problem of model simplification in the context of the seakeeping simulation. Sections 4.4 to 4.7 present our algorithms to reduce the computation time for the seakeeping simulation.

4.2 USSV Dynamics Simulation for Single Wave Component

In this section we present the governing equations of the model we implemented followed by a brief description of the developed simulator and time profile for computationally intensive steps in the simulation. The governing equations of fluid flow and boat motion discussed in this section are adapted from Newman [New77], Fossen [Fos94], and Krishnamurthy *et al.* [KKF05].

4.2.1 Governing Equation

For determining the dynamic pressure due to wave boat interaction we used the well known potential flow theory as given in Newman's textbook [New77]. We assume the flow to be irrotational ($\nabla \times \vec{V} = \vec{0}$), where, $\vec{V}$ is the fluid velocity. This means that there exists a velocity potential ϕ, such that,

$$\nabla\phi = \vec{V} \tag{4.1}$$

Now, let us assume that the fluid is incompressible ($\nabla.\vec{V}$), which gives us Laplace's equation.

$$\nabla^2\phi = 0 \tag{4.2}$$

The boundary conditions are:

(i.) Kinematic boundary condition: A fluid particle on the free surface of fluid will remain on the free surface. Let the free surface of the fluid be defined by the function $z = \zeta(x, y, t)$. Let us define a function $F(x, y, z, t) = z - \zeta = 0$, which is identically zero. From the condition stated in the kinematic boundary condition, the substantial derivative $\frac{D}{Dt}$ of F will vanish as the particles of fluid on free surface will remain on the free surface always. This can be expressed as follows.

$$\frac{D}{Dt}(z-\zeta)=0 \tag{4.3}$$

(ii.) Dynamic boundary condition: Pressure experienced by a particle on the free surface is same as the atmospheric pressure (p_0). We know, from the conservation of momentum, that a fluid particle moving with velocity V experiences a pressure p_{dyn}, given as follows:

$$\begin{aligned}\rho\frac{DV}{Dt}+\nabla p_{dyn}&=0,\\ \Rightarrow \rho\left[\frac{\partial V}{\partial t}+V.\nabla V\right]+\nabla p_{dyn}&=0,\\ \Rightarrow +\nabla p_{dyn}=-\rho\left[\frac{\partial V}{\partial t}+V.\nabla V\right]&,\\ \Rightarrow p_{dyn}=-\rho\left[\frac{\partial\phi}{\partial t}+\frac{1}{2}\nabla\phi.\nabla\phi\right]&\end{aligned}$$

where, ρ is the density of water in kg/m^3.

$p_{hstat}=p_0-\rho gz$, the negative sign appears because the Z - axis points upwards.

The total pressure is thus given as sum of dynamic and hydrostatic pressure and the dynamic boundary condition is expressed as follows:

$$\left[-\rho(\frac{\partial\phi}{\partial t}+0.5\nabla\phi.\nabla\phi)+p_0-\rho gz\right]_{z=\zeta}=p_0 \tag{4.4}$$

Equation 4.4 can be rewritten by canceling the term p_0 as follows:

$$\zeta=-\frac{1}{g}\left[\frac{\partial\phi}{\partial z}+\frac{1}{2}\nabla\phi.\nabla\phi\right]_{z=\zeta} \tag{4.5}$$

Substituting ζ in Equation 4.5 in Equation 4.3 gives us the combined equation representing both kinematic and boundary conditions as follows [New77]:

$$\left\{ \frac{D}{Dt} \left(z + \frac{1}{g} \left[\frac{\partial \phi}{t} + \frac{1}{2} \nabla \phi . \nabla \phi \right] \right) \right\}_{z=\zeta} = 0 \tag{4.6}$$

The boundary condition given in Equation 4.6 is nonlinear, as the free surface height function is not known in advance. In order to simplify, Newman proposed to linearize the velocity potential about the nominal height $z = 0$ as follows [New77]:

$$\phi(x, y, \zeta, t) = \phi(x, y, 0, t) + \zeta \left(\frac{\partial \phi}{\partial z} \right)_{z=0} + ... \tag{4.7}$$

Now, using the linearized values of ϕ of various order, sequence of boundary conditions, representing Equation 4.6 can be written, which are valid on known plane $z = 0$. We reproduce the boundary conditions for first and second order approximation of ϕ (after ignoring higher order terms) from Newman's book in Equation 4.8 and Equation 4.9 [New77].

$$\frac{\partial^2 \phi}{\partial t^2} + g \frac{\partial \phi}{\partial z} = 0 \tag{4.8}$$

The above boundary conditions are reduced to Equation 4.9 as given by Newman [New77].

$$g \frac{\partial \phi}{\partial z} + \frac{\partial^2 \phi}{\partial t^2} + 2 \nabla \phi . \nabla (\frac{\partial \phi}{\partial t}) - \frac{1}{g} \frac{\partial \phi}{\partial t} \frac{\partial}{\partial z} \left(\frac{\partial^2 \phi}{\partial t^2} + g \frac{\partial \phi}{\partial z} \right) = 0 \tag{4.9}$$

The solution of Equation 4.2 satisfying the boundary condition (Equation 4.9) yields the expression for the velocity field ϕ and an ocean wave free surface ζ for the ocean bed with infinite depth as shown below [New77].

$$\phi = \frac{gA}{\omega} \exp(kz) \sin(kx \cos\theta_w + ky \sin\theta_w - \omega t) \tag{4.10}$$

where,

g: acceleration due to gravity in m/s^2,

θ_w: wave direction in $radian$,

ω: wave frequency in $radian/s$, and

A: wave amplitude in m.

$$\begin{aligned} \zeta(x,y,t) = A \cos(kx \cos\theta_w + ky \sin\theta_w - \omega t) + \\ 0.5A^2 k \cos(2kx \cos\theta_w + 2ky \sin\theta_w - 2\omega t) \end{aligned} \tag{4.11}$$

The wave number k is related to the wave frequency ω for the infinite depth by the dispersion relationship given in Equation 4.12.

$$k = \frac{\omega^2}{g} \tag{4.12}$$

The velocity potential ϕ and the wave elevation ζ thus obtained is the starting point for determining the dynamic pressure head due to the wave boat interaction. Using Bernoulli's equation, the dynamic pressure Φ can be expressed in terms of the velocity potential ϕ as follows:

$$\Phi(x,y,z,t) = -\rho \left[\frac{\partial\phi}{\partial t} + \frac{1}{2} \nabla\phi . \nabla\phi \right] \tag{4.13}$$

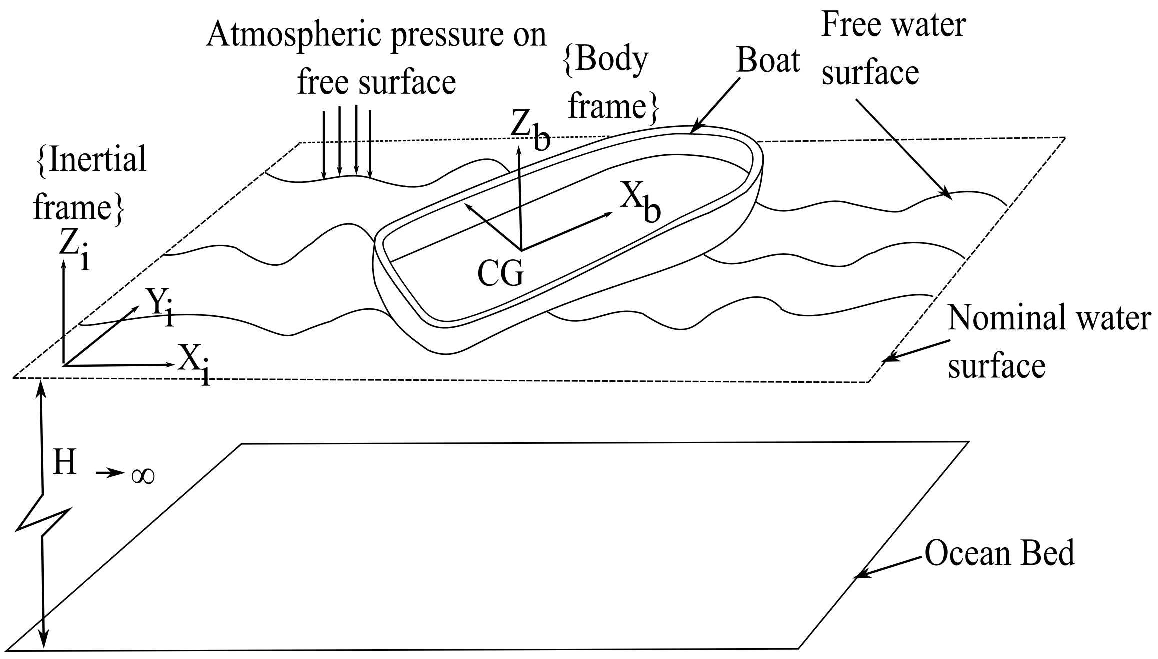


Figure 4.2: *Description of coordinate systems used in the presented model: Inertial and body coordinate systems are shown.*

The dynamic pressure head Φ can be integrated over the instantaneous wet surface of the boat S_B to obtain the force due to the wave boat interaction.

$$F_W = \begin{bmatrix} \rho \oint_{S_B} \left[\frac{\partial \phi}{\partial t} + 0.5 \nabla\phi.\nabla\phi\right] d\vec{S} \\ \rho \oint_{S_B} \left[\frac{\partial \phi}{\partial t} + 0.5 \nabla\phi.\nabla\phi\right] \left(\vec{r} \times d\vec{S}\right) \end{bmatrix} \tag{4.14}$$

where,

S_B: instantaneous wet surface of the USSV.

We implemented the 6-DOF dynamics model for USSVs given by Fossen [Fos94]. In this model, the USSV is assumed to be a rigid body. The origin of the inertial frame of reference is set at the nominal water level with Z axis vertical and pointing upwards. The body coordinate system is attached to the boat's center of gravity. The coordinate systems used are shown in Figure 4.2.

The governing dynamics equation is given in Equation 4.15.

$$\begin{aligned} &M_H\dot{v} + C_H(v)v + D_H(v)v + g(p) = F_E + F_P \\ &\dot{p} = J_p(v) \end{aligned} \tag{4.15}$$

where,

$p = [x, y, z, \theta_x, \theta_y, \theta_z]^T$ is pose vector expressed in the inertial frame, (x, y, z) is the Cartesian coordinate in m and θ's are Euler angles about subscript axes in $radians$,

$v = [v_x, v_y, v_z, \alpha_x, \alpha_y, \alpha_z]^T$ is velocity vector expressed in the body frame relative to the inertial frame, ($v_t = [v_x, v_y, v_z]^T$) is linear velocity in m/s and $v_r = [\alpha_x, \alpha_y, \alpha_z]^T$'s is angular velocity in $radian/s$,

$R = \begin{bmatrix} c_yc_z & s_xs_yc_z - c_xs_z & c_xs_yc_z + s_xs_z \\ c_ys_z & s_xs_ys_z + c_xc_z & c_xs_ys_z - s_xc_z \\ -s_y & s_xc_y & c_xc_y \end{bmatrix}$ is rotation matrix rotating a vector expressed in the body frame to the inertial frame, c_x means $\cos\theta_x$,

$J = \begin{bmatrix} R & 0_{3\times3} \\ 0_{3\times3} & J_r \end{bmatrix}$ is Jacobian matrix,

$J_r = \begin{bmatrix} 1 & s_xt_y & c_xt_y \\ 0 & c_x & -s_x \\ 0 & \frac{s_x}{c_y} & \frac{c_x}{c_y} \end{bmatrix}$,

$\vec{x}\times \equiv S(\vec{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$ is matrix dual (for cross product) of vector $\vec{x} = [x_1, x_2, x_3]^T$,

$p_{G,B}$ is vector representing the position of center of gravity in the body frame of reference,

m is mass of the USSV in kg,

I_b is 3×3 matrix representing the inertia tensor of the USSV in kgm^2,

$M_{RB} = \begin{bmatrix} mI_{3\times3} & -mS(p_{G,B}) \\ mS(p_{G,B}) & I_b \end{bmatrix}$ is matrix representing inertia tensor of the USSV,

M_A is (6×6) diagonal matrix representing the added mass of the USSV,

$$\begin{aligned} M_{A,11} &= 0.1m \\ M_{A,22} &= 4.75\rho a^2 \\ M_{A,33} &= 4.75\rho a^2 \\ M_{A,44} &= 4.75\rho a^2 \\ M_{A,55} &= 0.396\rho a^2 L_x^2 + 0.0833\frac{0.1m}{L_X}L_z^3 \\ M_{A,66} &= 0.0833\frac{0.1m}{L_X}L_y^3 + 0.396\rho a^2 L_x^3 \end{aligned}$$

,

where L_x, L_y, and L_z are length, width and height of the bounding box of the hull respectively,

$M_H = M_{RB} + M_S$ is $(6\ times 6)$ matrix representing the the total inertia,

$C_{RB} = \begin{bmatrix} mS(v_r)_{3\times3} & -mS(v_r)S(p_{G,B}) \\ mS(v_r)S(p_{G,B}) & -S(I_b v_r) \end{bmatrix}$ is Coriolis and centripetal matrix,

$C_A = \begin{bmatrix} 0_{3\times3} & -S(M_{A,11}v_t + M_{A,12}v_r) \\ -S(M_{A,11}v_t + M_{A,12}v_r) & -S(M_{A,21}v_t + M_{A,22}v_r) \end{bmatrix}$ is matrix representing the effect of added mass,

$M_{A,ij}$ represents (i, j) sub-matrix of M_A of size 3×3,

$C_H = C_{RB} + C_A$ is 6×6 matrix representing the total effect of Coriolis and added mass term,

D_H is 6×6 damping matrix,

$g(p)$ is 6×1 vector representing the restoring force expressed in the body frame in N,

F_E is 6×1 vector representing the environment force vector expressed in the body frame in N,

F_W is 6×1 vector representing the ocean wave and the boat interaction force expressed in the body frame in N, and

F_P is 6×1 actuation force vector expressed in the body frame in N.

In the right hand side of Equation 4.15, term F_E means environmental force. The environmental force consists of effects due to ocean current, wind, and ocean wave. We ignore the effects of ocean current and wind and only consider the force due to ocean wave F_W in the implementation. However, given a suitable model of the force due to ocean current and wind, it can be easily incorporated into the simulation framework. We thus replace F_E by F_W in Equation 4.15. The force due to ocean wave is computed by using Equation 4.14, which is obtained by the potential flow theory in the implementation.

4.2.2 Implementation of Simulator

We implemented the model given in Section 4.2.1 to simulate the motion of a USSV under given initial state and wave conditions. We assume that the USSV geometry is provided in polygonal form. We chose .STL as the input file format for the USSV model. To compute the wave force we discretized Equation 4.14 as follows:

$$F_W = \begin{bmatrix} \rho \sum_{j=1}^{n} \Phi_j d\vec{S_j} \\ \rho \sum_{j=1}^{n} \Phi_j \left(\vec{r_j} \times d\vec{S_j} \right) \end{bmatrix} \tag{4.16}$$

where,

$$\Phi_j = \left[\frac{\partial \phi}{\partial t} + 0.5 \nabla \phi . \nabla \phi \right]_{p_j} \tag{4.17}$$

and n is the number of instantaneous wet facets. So, the steps of simulation are enumerated below.

(i.) Determine the instantaneous wet surface (S_B) of the USSV by finding out the facets lying below and on the wave surface given by Equation 4.11.

(ii.) Use Equation 4.10, 4.13, 4.16, and S_B determined in the previous step to compute the wave force F_W.

(iii.) Determine the displaced volume of water by the surface S_B and compute the buoyancy force $g(p)$.

(iv.) Determine the inertia matrix (M_H), Coriolis matrix ($C(v)$) and damping matrix ($D(v)$). We estimated the added mass matrix M_A using strip theory as explained by Fossen [Fos94]. The added mass can, in practice, be computed by performing system identification for a given USSV model under various sea-states. For the simulation purpose, we utilized strip theory to estimate the added mass matrix M_A. It should be noted that strip theory is only used here to estimate the added mass, and not the forces due to ocean wave.

Table 4.1: *Typical time profile for the simulation process for* 1500 *time steps of size* 0.07 *s.*

Computation Item	Computation Time (s)	Percentage
Force due to ocean Wave	740.70	62.5
List of wet facets	444.65	37.49
Differential Equation solution	0.67	0.06

(v.) Solve Equation 4.15 numerically using Runge-Kutta fourth order (RK4) integration technique by substituting $F_E = F_W$.

4.2.3 Time Profile of Simulator

The simulator explained in Section 4.2.2 has three main operations being repeated in each time step as listed below:

(i.) Computation of the list of wet facets.

(ii.) Computation of the surface integral for determining the wave force and the buoyancy force.

(iii.) Solving the differential equation given in Equation 4.15.

We ran the simulation for five different wave directions ($\theta_w = 0, 0.63, 1.25, 1.88, 2.51$ radians). The boat model for the simulation had 960 triangular facets. We computed the average time taken for the boat model for 1500 simulation time steps (with each time step of length 0.07 sec) for each operation over all five wave directions, and the result is shown in Table 4.1. We performed the simulation on a computer with Intel(R) Core(TM)2 Quad 2.83 GHz CPU and 8GB RAM. All the computations reported in this chapter are performed on the same computer and for the same boat model, so that the comparisons between computation times are fair.

It is evident from the Table 4.1 that most of the time (about 99.94%) is spent in computing the force due to ocean waves and wet surface determination. Also, Table 4.1 suggests that the time taken for computing one time step of length 0.07 s is about $\frac{1186.02}{1500} \approx 0.8$ s, which is far from the real-time performance.

4.3 Problem Statement and Solution Approach

As discussed in the Section 4.2.3, the computation time for the USSV simulation is too high for attaining real-time refresh rate, which motivates us to find ways to simplify the computations to reduce the computation time. In this section we shall present the problem statement formally and give an overview of approach to solve the problem.

4.3.1 Problem Statement

Given a polygonal representation of the USSV hull geometry and its dynamics properties like mass, moment of inertia tensor, damping matrix, etc. and fluid and ocean wave characteristics such as density, wave direction, amplitude, frequency, etc., perform model simplification to do dynamics simulation of the USSV in real-time.

4.3.2 Approach

Based on the observations in the Section 4.2.3, the computations can be made faster by:

(i.) Determining the list of facets in the USSV model, which will never come in contact with the water under normal operating conditions (without USSV sinking completely or getting turned over) and removing them to get the wettable region of the boat model.

(ii.) Performing clustering of the USSV model facets to club all the facets with similar dynamic pressure value, in order to reduce the time taken to compute the wet facets and force due to ocean wave and hull interaction.

(iii.) Exploiting the parallel nature of the computation and reducing the computation time by utilizing multi-core computers.

(iv.) Estimating the force due to ocean wave acting on the USSV in a time step using force in the previous time step instead of explicitly computing forces in each time step. We call this approach as temporal coherence, as the forces do not change significantly in short period of time and depend on the force in the previous time step.

In the following sections we discuss the above simplification strategies in detail.

4.4 Preprocessing of Boat Model to Determine Wettable Region

A USSV model can be very complex geometrically and can have many unnecessary details from the point of view of computing hydrostatic and hydrodynamic forces. If the features that are covered by the hull are retained during the force computations then the following problems might arise:

(i.) The facets of the features covered by the hull would be determined as the wet facets and would introduce error into the force computation. This is because those facets never come in contact with the water as they are occluded by the facets of the boat hull.

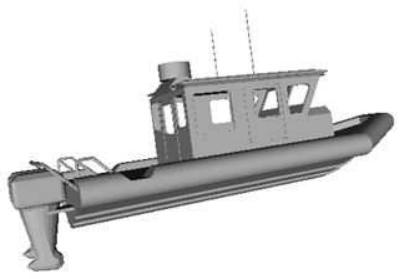
(ii.) Testing of such facets during the simulation for whether they touch water, would be an additional computational overhead during the simulation runtime leading to performance deterioration.

For computation of hydrodynamics and hydrostatic forces, we are only interested in facets that might come in contact with the water (without the USSV sinking completely or getting turned over). This is a simple problem to solve but a crucial step for preprocessing the USSV model. We assume that the USSV model is oriented in such a way that the axis of yaw is aligned to inertial Z axis. Also, we assume that the undercut features on the boat hull are negligible. The steps we follow to automate this process are enumerated below:

(i.) For all facets, if the normal makes an angle larger than 120^o, with the negative Z direction, remove them.

(ii.) For all the remaining facets, shoot ray from the center of gravity of the facets in the negative Z direction. If the rays intersect at least one facet other than itself, remove the facet.

By following the above two steps the wettable region of the boat model as shown in Figure 4.3 is obtained.


(a) *3D model for USSV.*

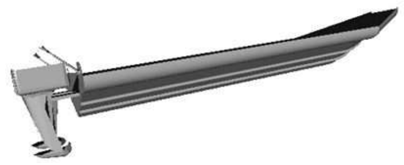

(b) *Simplified USSV model.*

Figure 4.3: *USSV model simplification by removing facets that never come in contact with water.*

4.5 Simplification Using Clustering

The preprocessed *wettable* model obtained by the technique described in the Section 4.4 is used for determining the hydrodynamic and hydrostatic forces by computing the surface integral of the dynamic pressure head acting over the USSV's instantaneous wet surface. The computational effort spent on the surface integral computation using Equation 4.16 can be enumerated as follows:

(i.) Computing the value of Φ at each facet center in the list of wet facets S_B in each time step using Equation 4.18.

$$\begin{split}\Phi(x,y,z) = \frac{gAk}{\omega}\exp(kz)(\cos\theta_w \cos f\dot{x} + \\ \sin\theta_w \cos f\dot{y} + \sin f\dot{z} - \frac{\omega}{k}\cos f) + 0.5(\frac{g^2k^2A^2}{\omega^2}\exp(2kz))\end{split} \tag{4.18}$$

where,

$$f = kx\cos\theta_w + ky\sin\theta_w - \omega t \tag{4.19}$$

Equation 4.18 is obtained by using Equation 4.10 and 4.17, in terms of wave amplitude, wave number, wave frequency and wave direction.

(ii.) Evaluating the sum in Equation 4.16.

A look at Equation 4.18 shows that we need to perform at least four trigonometric evaluations and one exponential function evaluation. Now, even if there are on an average 500 wet facets in each time step of the differential equation solver, we need to perform 2000 evaluations of trigonometric and exponential functions. Trigonometric and exponential function calls are computationally expensive and must be minimized. In order to minimize such function calls we can make use of the fact that the dynamic pressure values at the facets close to each other are similar. Such contiguous facets can be clustered to reduce the number of trigonometric and exponential function calls by introducing some small error. It should be noted that the clustering is performed before the simulation (*i.e.*, off-line) and thus the computational complexity of clustering does not affect the simulation speed.

In this section, we introduce the concept of clustering the hull facets to reduce the time taken to compute the force due to the ocean waves and to compute the wet facets. We also theoretically analyze the effect of clustering the facets on the computation time and error introduced and provides numerical validation of the theoretical results.

4.5.1 Clustering Algorithm

Let us represent the USSV's center of gravity (or the origin of the body frame of reference attached to the USSV) by $\vec{x_c}$, and position of j^{th} wet triangular facet by $\vec{x_j}$ in the inertial frame of reference. Also, let us represent, the position of the

j^{th} wet triangular facet by $\vec{r_j}$ with respect to $\vec{x_c}$. Now, from Equation 4.17 and using Taylor series expansion, Φ at a given wet triangular facet can be expressed as follows:

$$\Phi(\vec{x_j}) = \Phi(\vec{x_c} + \vec{r_j}) \approx \Phi(\vec{x_c}) + \vec{r_j}^T \nabla\Phi(\vec{x_c}) + 0.5\vec{r_j}^T \nabla^2\Phi(\vec{x_c})\vec{r_j} \tag{4.20}$$

We can ignore higher order terms because the r^{th} term contains $\nabla^{r-1}\Phi$, which is proportional to k^r, and for ocean waves usually $0.0 < k < 1.0$. Thus, the terms $r > 3$ are negligible. Equation 4.20 suggests that the value of Φ depends only on $\vec{r_j}$ for a given position $\vec{x_c}$ of the USSV. This implies that the value of Φ for facets which are sufficiently close to each other geometrically can be assumed to be the same. This observation establishes the basis of the decomposition of the facet set to form clusters.

Definition 13 A cluster κ of size $\gamma \in \mathbb{R}$ is defined as the set of triangular facets t_j such that the diagonal of the bounding box of κ is smaller than or equal to γ.

Definition 14 Cluster center of a cluster κ is defined as a point $P_c \in \mathbb{R}$, such that,

$$P_c = \frac{\sum_{i=1}^{M} a_i P_i}{\sum_{i=1}^{M} a_i} \tag{4.21}$$

where, M is the cardinality of κ and P_i is the centroid of the i^{th} facet belonging to κ.

The problem of clustering can be viewed as a set partitioning problem. The set of facets representing the hull of the USSV is partitioned into subsets (clusters). The partitioning is done in such a way that the average distance of all the facets belonging to a cluster from the cluster center is minimized. We utilized K-means algorithm to accomplish the clustering. The steps of clustering algorithm we used are listed as follows:

Algorithm 4.1 - K-means Clustering

Input

(a.) List of n triangular facets Θ,

(b.) cluster count C, and

(c.) subdivision threshold $\epsilon \in \mathbb{R}$.

Output List of C clusters Υ such that $C < n$

Steps

(i.) Determine the bounding box B of Θ.

(ii.) Recursively subdivide each triangle inside Θ using Loop subdivision until triangle area is less than $\epsilon Diagonal(B)$.

(iii.) Determine $q = \lceil \sqrt{\frac{width(B)}{length(B)} C} \rceil$ and $p = \lceil \frac{C}{q} \rceil$. It can be seen that $pq \geq C$. Divide the XY plane of B into $p \times q$ rectangular grids. If $pq > C$, merge $pq - C$ rectangles. Use the exactly C regions on the XY plane, thus obtained to partition the set of facets obtained in step (ii) into C subsets. For each region, project vertices of each facet on the XY plane. If at least two of the projected vertices lie inside the region add facet into the subset and mark the facet as visited. Repeat the process for each region and unvisited facets. This step returns C clusters of facets. This heuristic step is performed to generate an initial clustering.

(iv.) Determine the cluster center of each of the C clusters obtained in step (iii), using Equation 4.21.

(v.) For each cluster center, determine the facets nearest to the cluster center, than any other center. This will yield a new partition and hence new set of C clusters.

(vi.) Determine cluster centers for each of the C new clusters and determine the mean squared distance from the respective old centers.

(vii.) If the mean squared distance is less than a small value (we chose 10^{-5}), return the C clusters, else go to step (v).

We used the clustering approach (with $C = 45$, $\epsilon = 10^{-3}$) for simulation of the USSV model shown in Figure 4.3(b) with 960 facets, for 1500 time steps (representing time step of size 0.07 s) and for five different wave directions. The results are shown in Table 4.2. We used Equation 4.22 to compute the force error introduced due to the clustering simplification.

$$e_F = \frac{\|\langle F_W - g(p)\rangle_{\text{baseline}}\| - \|\langle F_W - g(p)\rangle_{\text{simplified}}\|}{\|\langle F_W - g(p)\rangle_{\text{baseline}}\|} \tag{4.22}$$

where, subscript baseline signifies the computation performed without using the clustering approximation approach, whereas subscript simplified signifies the computation performed using clustering approximation approach. Table 4.2 shows that

Table 4.2: *Results of simplification using clustering approach for (1500 time steps, $C = 45$, and $\epsilon = 10^{-3}$).*

θ_w (radians)	Baseline computation time (s)	Clustering computation time (s)	Average force error introduced due to clustering based computation over baseline (%)
0.000	1189.52	219.10	4.30
0.628	1178.90	218.70	3.30
1.256	1185.43	217.60	1.59
1.884	1220.49	216.40	1.71
2.512	1182.54	217.40	3.45
Average	**1191.40**	**216.80**	**2.88**

the average computation time reduced by a factor of $\frac{1191.40}{216.80} = 5.50$ with an average force error of 2.88% using the clustering approach.

4.5.2 Effect of Cluster Count on Error and Computation Time

Intuitively speaking, increasing the number of clusters should increase the time taken to compute and should reduce the error. In this section, we derive the dependence of computation time and error introduced on the chosen cluster count and present numerical validation of the derived relationship. Let us assume that the boat hull has n facets and we partition the facets into C clusters using the K-means algorithm discussed before. Let the j^{th} cluster contain N_j facets. Let us denote the total area vector of the facets contained by the j^{th} cluster as $\vec{s_j}$. We can express $\vec{s_j}$ as follows:

$$\vec{s_j} = \sum_{p=1}^{N_j} d\vec{S}_{p,j} \tag{4.23}$$

where, $d\vec{S}_{p,j}$ is the area of the p^{th} facet of the j^{th} cluster. Based on Table 4.1 and the discussion in Section 4.4, we know that the major amount of time taken in computations can be divided into three main operations:

(i.) computation of the dynamic pressure, (ii.) computation of the list of wet facets, and

(iii.) computation of the instantaneous volume of the wet region.

Let, the time taken for computing dynamic pressure at a given point be T_p, the time taken for doing a test on one cluster for wet facets be T_w, and the average time taken for the computation of total volume of the wet region be T_v. The time taken for computing the volume of the wet region (T_v) depends only on the number of wet facets which for a boat hull does not vary much and thus, we can assume it to be independent of the cluster count.

Firstly, let us estimate the variation of the average time for computation with the cluster count (C). In the case of C clusters, we compute the dynamic pressure C number of times in one time step, and thus, the total time taken for computing the dynamic pressure is CT_p. Similarly, for determining wet facets we perform C tests which takes CT_w time. Thus, the total time (T) taken for computing the force for one simulation time step is given in Equation 4.24.

$$T = T_v + C(T_w + T_p) \tag{4.24}$$

The simulation time, thus, varies as $O(C)$ based on Equation 4.24.

In order to compute the force using clustering, we only compute the dynamic pressures at the cluster centers and use them to compute the total force as follows:

$$\vec{F}_{cluster} = \sum_{j=1}^{C} \Phi(\vec{x_j})\vec{s_j} \tag{4.25}$$

where, $\vec{x_j}$ is the cluster center of j^{th} cluster.

The exact total force (computed by considering all facets instead of just the clusters) can be given as follows:

$$\begin{aligned}
\vec{F}_{exact} = \sum_{j=1}^{C}\sum_{k=1}^{N_j}\Phi(\vec{x}_j + \vec{r}_{k,j})d\vec{S}_{k,j} = \\
\sum_{j=1}^{C}\left[\Phi(\vec{x}_j)(\sum_{k=1}^{N_j} d\vec{S}_{k,j}) + \sum_{k=1}^{N_j}(\vec{r}_{k,j}.\nabla\Phi(\vec{x}_j))\, d\vec{S}_{k,j})\right] + H\vec{O}T = \\
\sum_{j=1}^{C}\Phi(\vec{x}_j)\vec{s}_j + \sum_{j=1}^{C}\sum_{k=1}^{N_j}(\vec{r}_{k,j}.\nabla\Phi(\vec{x}_j))d\vec{S}_{k,j} + H\vec{O}T = \\
\vec{F}_{cluster} + \sum_{j=1}^{C}\left[\sum_{k=1}^{N_j}(\vec{r}_{k,j}.\nabla\Phi(\vec{x}_j))d\vec{S}_{k,j}\right] + H\vec{O}T
\end{aligned} \tag{4.26}$$

where $\vec{r}_{k,j}$ is the position vector of centroid of the k^{th} facet of the j^{th} cluster, N_j is the number of facets in the j^{th} cluster, and HOT represent the higher order terms. Now, we can compute error in the force estimation $\vec{e}_{force}$ due to the use of the clustering approach as follows:

$$\begin{aligned}
\vec{e}_{force} = \vec{F}_{exact} - \vec{F}_{cluster} = \sum_{j=1}^{C}\left[\sum_{k=1}^{N_j}(\vec{r}_{k,j}.\nabla\Phi(\vec{x}_j))d\vec{S}_{k,j}\right] + H\vec{O}T \approx \\
\sum_{j=1}^{C}\left[\sum_{k=1}^{N_j}(\vec{r}_{k,j}.\nabla\Phi(\vec{x}_j))d\vec{S}_{k,j}\right] \text{ ignoring higher order term } H\vec{O}T
\end{aligned} \tag{4.27}$$

The magnitude of error in the force estimation can be expressed and approximated by the following equation.

$$\|\vec{e}_{force}\| = \|\sum_{j=1}^{C}\left[\sum_{k=1}^{N_j}(\vec{r}_{k,j}.\nabla\Phi(\vec{x}_j))d\vec{S}_{k,j}\right]\| \leq R\alpha S \tag{4.28}$$

where, α is the maximum value attained by the function $\|\nabla\Phi\|$ defined in the instantaneous region bounded by the hull geometry. We know that α must be a scalar and finite value as the hull region is a finite region. S is the absolute value of the surface area of the wet surface, given as follows:

$$S = \|\sum_{j=1}^{C}\sum_{k=1}^{N_j} d\vec{S}_{k,j}\| \tag{4.29}$$

R is an upper bound on the bounding radius of cluster. Since, the total surface area of the hull is S and the hull is divided into C almost equal area regions, the total surface area $A_{cluster}$ of facets in a cluster can be approximated as follows:

$$A_{cluster} \approx \frac{S}{C} \tag{4.30}$$

Since the boat geometry is smooth (without many undercuts), the surface area of the bounding sphere of a cluster is always more than the surface area of the facets contained in the cluster We can thus, estimate R as follows:

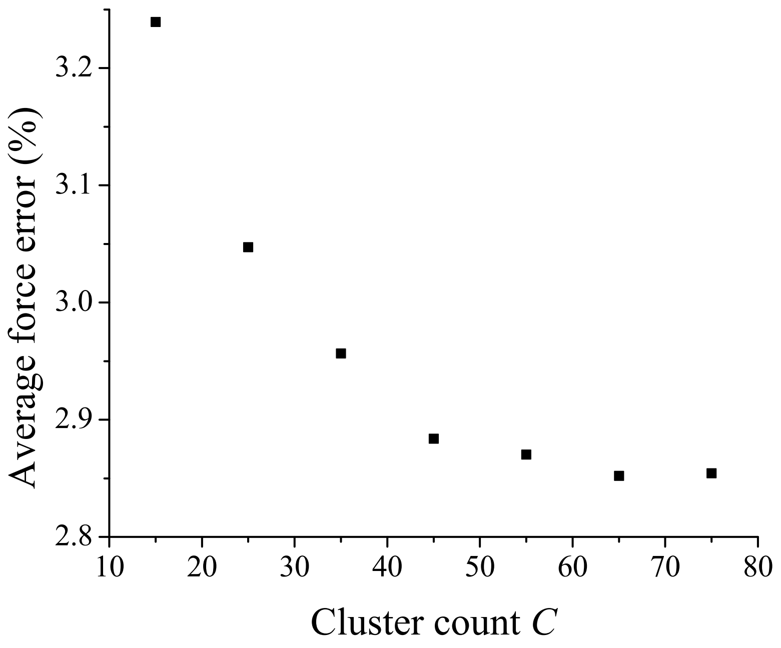
$$R < \sqrt{\frac{A_{cluster}}{4\pi}} < \sqrt{\frac{S}{4\pi C}} \tag{4.31}$$

Using, Inequality 4.31 and Equality 4.28, we get:

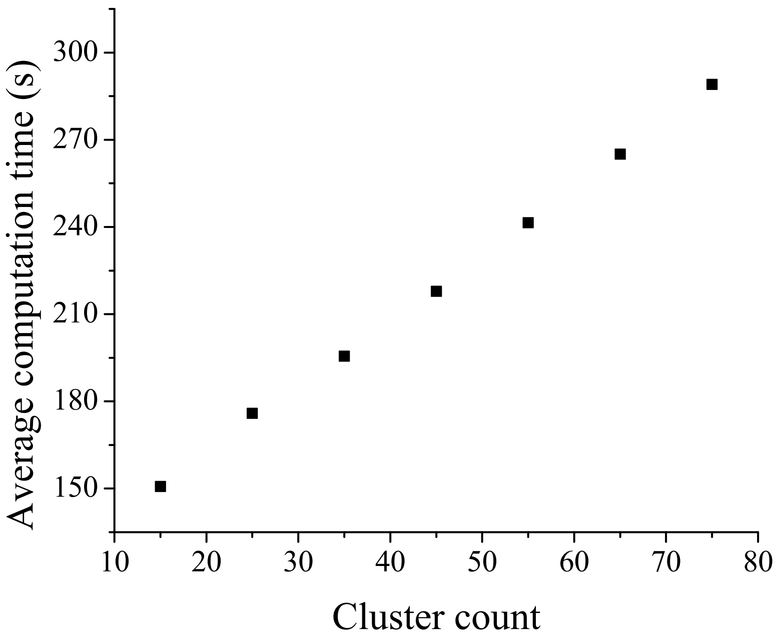
$$\|\vec{e}_{force}\| < \frac{\alpha S^{1.5}}{2\sqrt{\pi C}} \tag{4.32}$$

Thus the error in the force computation introduced due to the cluster approximation varies as $O(C^{-0.5})$.

We conducted numerical tests to validate the derived results. In these tests, the USSV is hit by waves from five different directions for 1500 time steps (with each time step of length 0.07s) and the computations are performed using cluster sizes ranging



(a) *Variation of average force error with cluster count C.*



(b) *Variation of average computation time with cluster count C.*

Figure 4.4: *Variation of average force error and computation time with cluster count C.*

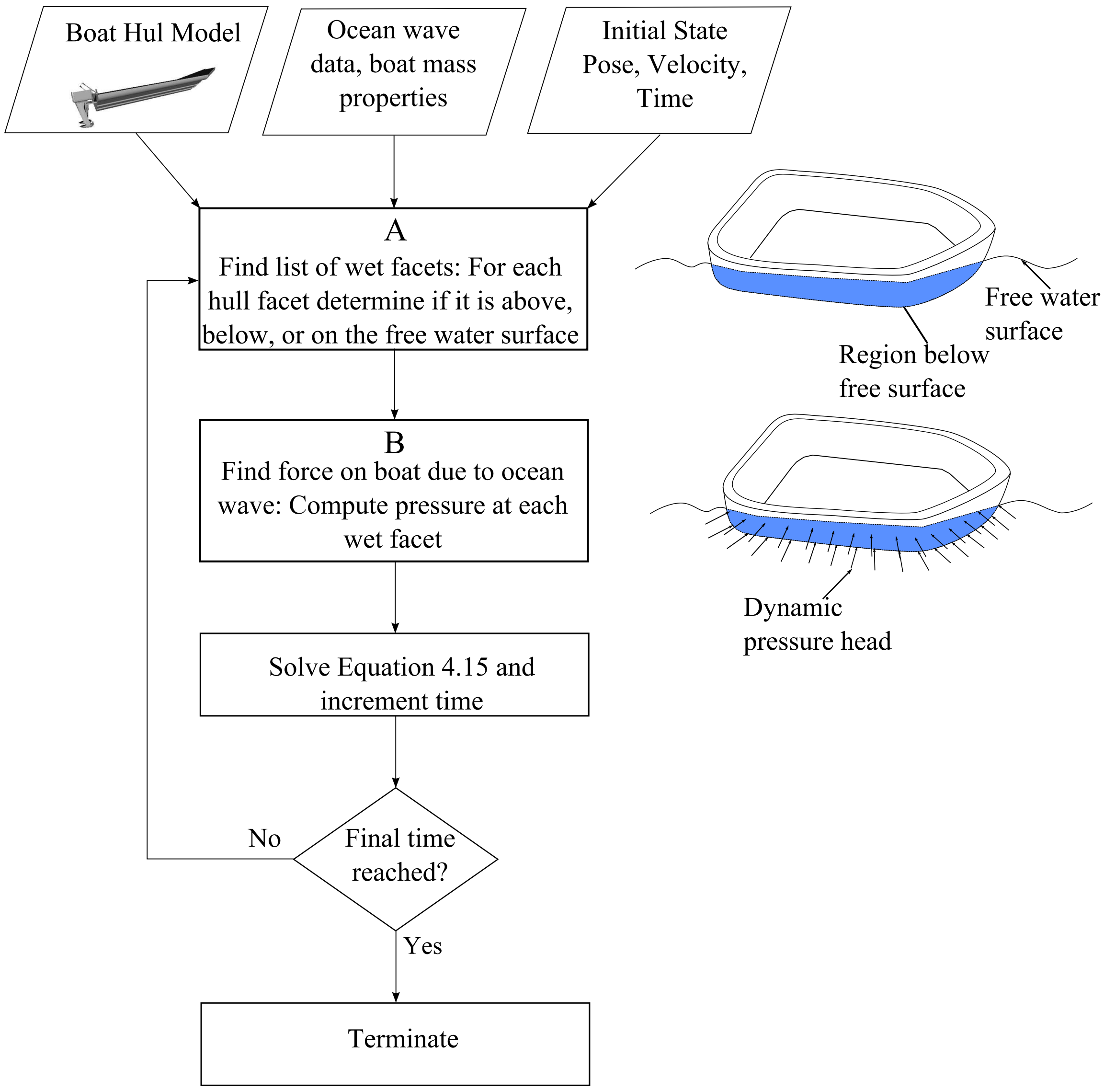
from 15 to 75 in the increments of 10. The variation of the average error and average time taken for 1500 time steps (averaged over 5 different wave directions) is plotted against the cluster sizes and presented in Figure 4.4. It can be observed in Figure 4.4

that the average computation time increases almost linearly while the error reduces non-linearly with the increasing cluster count agreeing well with Equations 4.24 and 4.32. It can be noticed in Figure 4.4 that the error percentage gets stalled at the value of about 2.85% and does not reduce significantly after $C = 65$. This is because the triangles defining the boat model are subdivided into smaller triangles before clustering. The clusters are then formed on the basis of geometric nearness of constituent triangles and not the similarity from the original face set. This causes a small error from the baseline computation and causes the stalling phenomenon in Figure 4.4(a). To summarize, cluster count C is a parameter that can be used as an approximation handle to control the time taken for computation and the error introduced due to the clustering approximation.

4.6 Parallelization

The problem of computing the wet facets and performing vector operations is amenable to parallelization as results of computation taking place at each cluster are independent of each other. After performing the clustering, computation time can further be reduced by using parallelization by changing the sequential code to parallel, without changing the error introduced. In Figure 4.5, block A is used for computing the wet facets and B for computing the forces. Computations performed on each facet are not dependent on the results from other facets. Thus, we can spawn a multiple number of threads based on available computing resources to balance the computation over each processor.

We performed the tests on Intel's Quad processors and employed four threads



Boat Hul Model
Ocean wave data, boat mass properties
Initial State Pose, Velocity, Time
A
Find list of wet facets: For each hull facet determine if it is above, below, or on the free water surface
Free water surface
Region below free surface
B
Find force on boat due to ocean wave: Compute pressure at each wet facet
Dynamic pressure head
Solve Equation 4.15 and increment time
No
Final time reached?
Yes
Terminate

Figure 4.5: *Simulation computation process flow.*

with dynamic scheduling. It should be noted that the computer we used for testing the parallel version of the code is the same as explained in the earlier section. We chose four threads with dynamic scheduling because the performance was best with four threads working in a numerical experiment we performed. We ran a simulation of 1500 time steps with consecutively 2 to 16 threads and the result is shown in Figure 4.6, which shows that the dynamic scheduling performs better than static

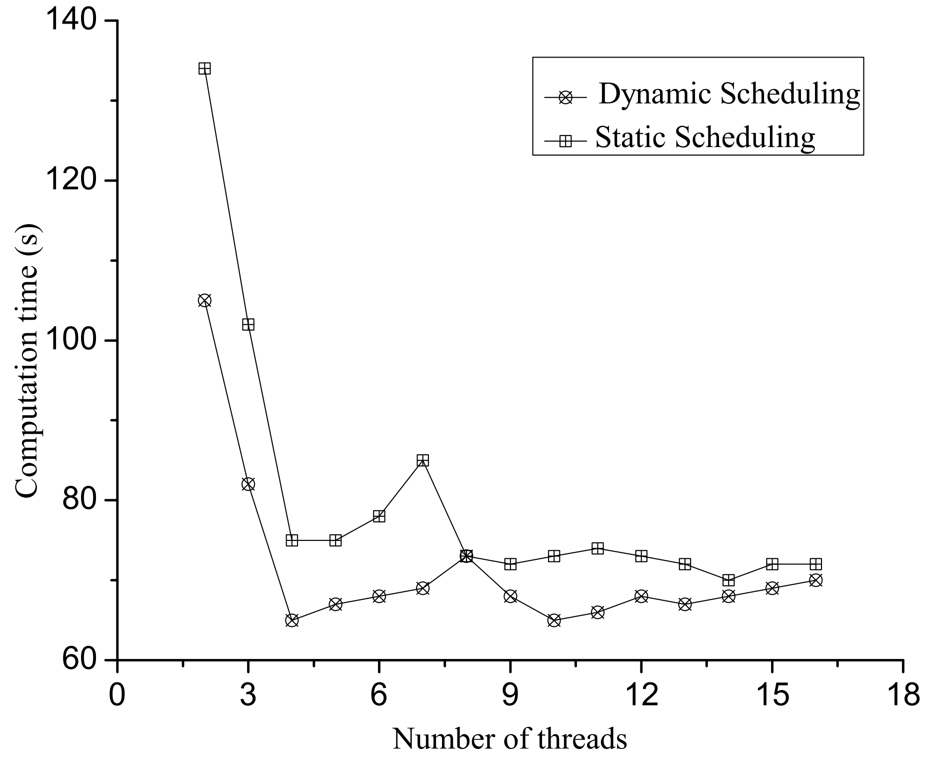


Figure 4.6: *Variation of computation time with number of threads.*

scheduling. In dynamic scheduling jobs are dynamically allocated to free threads whereas in static scheduling the jobs are preallocated to the threads and even if a thread finishes first, it waits for others to finish. The performance was best when four threads were chosen. We used OpenMP to perform parallelization [CJP08]. In the OpenMP based approach, preprocessor directives are placed before each looping statement and the OpenMP optimizes the assignment and scheduling of the tasks to the threads internally.

Table 4.3 shows the results for the same example from the Section 4.5. The average computation time was reduced by a factor of $\frac{1191.40}{67.00} = 17.78$ with an average force error of 2.88% using the parallelization and clustering based approach. It can be noted that the computation time reduced by a factor of $\frac{216.80}{67.00} = 3.24$ with

Table 4.3: *Improvement due to parallelization on top of clustering approach for (1500 time steps, $C = 45$, and $\epsilon = 10^{-3}$) with four threads on Quad processor.*

θ_w (radians)	Baseline computation time (s)	Clustering with parallelization computation time (s)	Average force error introduced due to clustering and parallelization based computation over baseline (%)
0.000	1189.52	68.90	4.30
0.628	1178.90	72.30	3.30
1.256	1185.43	68.50	1.59
1.884	1220.49	61.50	1.71
2.512	1182.54	63.80	3.45
Average	**1191.40**	**67.00**	**2.88**

respect to that of clustering and there is no change in the average force error after parallelization. The average time taken for one time step, by using parallelization along with clustering approach is about $\frac{67.00}{1500} = 0.045$ s, which can be used to simulate the USSV motion in real-time for time step of length 0.07 s.

4.7 Temporal Coherence

In the clustering based approach, we utilized the fact that the dynamic pressure values do not change significantly in the close spatial vicinity to simplify the computations. Another fact that can be used to further simplify the computations is that the force due to the ocean waves on the USSV does not change significantly in a short period of time. This simple but useful fact enables us to skip the computation of the list of the wet facets and ocean wave force in some time steps and reuse the list of the wet facets and force due to the ocean wave obtained in the previous time step. This leads to a significant reduction of the computation time with the introduction of some error. The basis of skipping the computation of the list of the wet facets and ocean wave force is the similarity of the instantaneous ocean wave height-field acting on the USSV in the consecutive time step. In other words, if the ocean wave in the proximity of the USSV does not change significantly in a given period of time, we can safely assume that the force acting on the USSV due to the ocean wave also does not change significantly and the force computed in the previous time step can be reused.

4.7.1 Temporal Coherence Algorithm

In this section, we present the algorithm implementing the idea of temporal coherence explained before. In order to explain the algorithm, we present the following two definitions first.

Definition 15 Let, B be the bounding box of the USSV and a rectangle R_B be the projection of B on the XY plane. Let Λ denote uniform grid of size $m \times n$ on R_B.

We define the instantaneous ocean wave height-field for a moving USSV as a vector $\vec{G} \in \mathbb{R}^{mn}$, such that the elements of $\vec{G}$ are the ordered elevations of the ocean wave at the mn grid points on Λ.

Definition 16 For a pair of ocean wave height-fields $\vec{G_1}$ and $\vec{G_2}$, we define the distance d_{hf} between $\vec{G_1}$ and $\vec{G_2}$ as the following second order norm.

$$d_{hf} = \|\vec{G_1} - \vec{G_2}\| \tag{4.33}$$

Algorithm 4.2 - Temporal Coherence

Input

(a.) USSV model,

(b.) initial conditions, and

(c.) tolerance τ.

Output Velocity and pose of the USSV at all time steps

Steps

(i.) In the first time step, *i.e.*, when the time is zero, determine the list of wet facets $S_{B,P}$ and the force due to the ocean wave $F_{W,P}$ on the USSV as described in the Section 4.2.2. Also determine the instantaneous ocean wave height-field $\vec{G}_i$. Solve Equation 4.15, by setting $F_W = F_{W,P}$ and $S_B = S_{B,P}$ and advance the time step.

(ii.) Determine the instantaneous height-field $\vec{G}$. Determine the distance d_{hf} between $\vec{G}_i$ and $\vec{G}$ using Equation 5.9. If $d_{hf} \geq \tau$ then compute the instantaneous list of wet facets and ocean wave force and store (by overwriting) in $S_{B,P}$ and $F_{W,P}$ respectively. Set $F_W = F_{W,P}$ and $S_B = S_{B,P}$ and solve Equation 4.15 and advance the time step.

(iii.) Repeat step (ii) until the simulation is terminated.

We applied the above algorithm on top of the clustering and parallel version of the code as described in the Sections 4.5 and 4.6. We chose $m = 3, n = 5$ and

Table 4.4: *Results of simplification using temporal coherence approach for* (1500 *time steps,* $C = 45$, $\epsilon = 10^{-3}$, *four threads on quad processor, and* $\tau = 0.10$).

θ_w (radians)	Baseline computation time (s)	Temporal coherence (on top of clustering and parallelization) computation time (s)	Average force error introduced due to clustering, parallelization, and temporal coherence based computation over baseline (%)
0.000	1189.52	43.46	9.30
0.628	1178.90	42.90	8.20
1.256	1185.43	40.20	3.40
1.884	1220.49	39.80	3.40
2.512	1182.54	42.70	4.20
Average	**1191.40**	**41.80**	**5.80**

$\tau = 0.10$ and performed a simulation of the USSV model shown in Figure 4.3(b) under the action of 5 different wave directions as described in the Sections 4.5. The average computation time for 1500 time steps (of length 0.07 s) for the test case is about 41.80 s.

The average computation time reduced by $\frac{1191.40}{41.80} = 28.50$ with an introduction of an average force error of 5.8% with respect to the baseline computation using the approaches based on clustering, parallelization, and temporal coherence together. The average time for computing one time step (of length 0.07 s) is about 0.028 s after performing the simplification based on the temporal coherence on top of the clustering and parallelization based simplification.

4.7.2 Effect of Wave Tolerance on Error and Computation Time

In this section, we derive the dependence of the computation time and error introduced on the chosen wave tolerance and present numerical validation of the derived relationship. Let us suppose that the component of ocean wave with highest amplitude (say A_m has the frequency of ω_m. The component of the ocean wave with the highest amplitude has the largest affect on the variation of the force on the

USSV. A wave component with amplitude A_m and the frequency ω_m takes $\frac{2\tau\pi}{A_m\omega_m}$ time for a variation of τ in water level at a given point. Also the wave has a time period given by $\frac{2\pi}{\omega_m}$. This implies that if we ignore the computation of the list of the wet facets (T_w) and ocean wave force (T_p) for the time when the variation of the wave is less than τ, the amount of computing time is given by Equation 4.34.

$$T = T_v + C(1 - \frac{\tau}{A_m})(T_w + T_p) \tag{4.34}$$

The terms C and T_v are explained in Equation 4.24.

In order to estimate the error in the force calculation due to the approximation introduced in this section, let us consider the variation of τ in the wave height at each point on the boundary of the USSV wet surface. If the perimeter of the curve obtained by the intersection of wave surface and the USSV surface is denoted by P, the maximum variation in force is given as $\Phi_m P\tau$, where Φ_m is the maximum possible dynamic pressure head. Thus the error in the force computation can be bounded by using Equation 4.35.

$$\|\vec{e}_{force}\| < \Phi_m P\tau \tag{4.35}$$

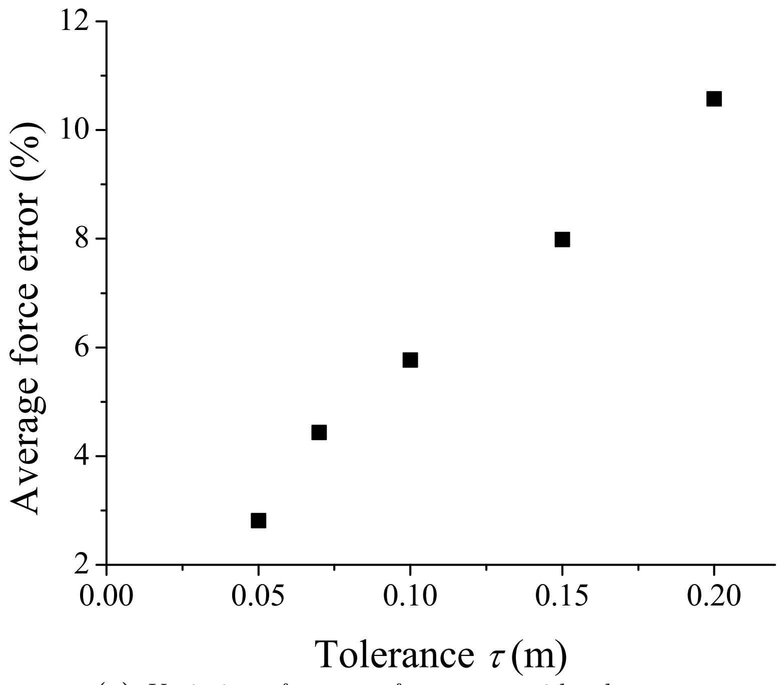
Equation 4.35 gives us the variation of the error in the force estimation due to the approximation introduced in this section as linear in terms of the tolerance.

In order to verify the above derived relationships, we performed numerical simulations for five different values of τ for the cluster size $C = 45$ and $\epsilon = 10^{-3}$ for five different wave directions. The variation of the average value of the force magnitude and computation time over five different wave directions with the tolerance τ is shown in Figure 4.7. The choice of tolerance dictates the reduction in the computation time and an increase in the average error in the force computation. This is because the tolerance τ is the magnitude of the difference between the wave surface experienced by the USSV at a time step and the previous time step which can be ignored to avoid the computation of the ocean wave force without introducing significant error.

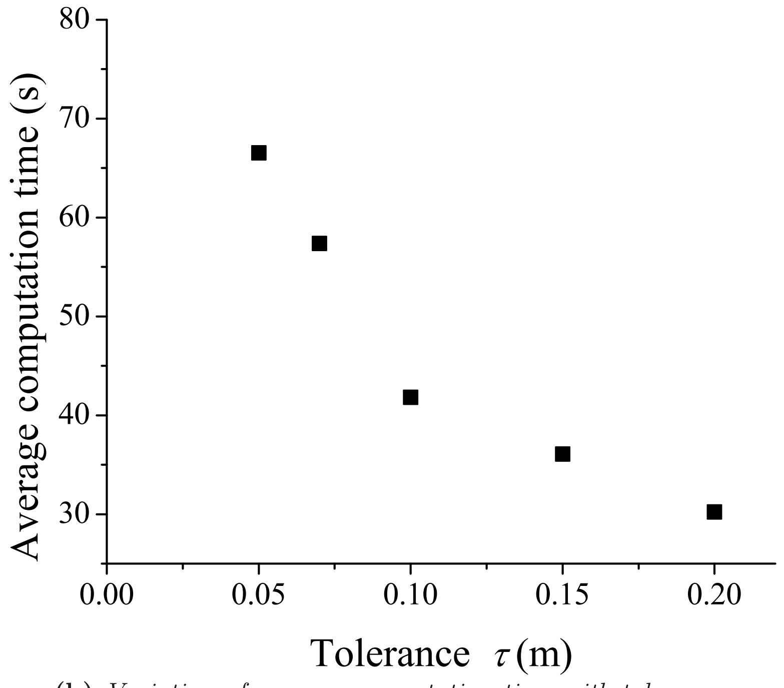
In summary, the tolerance can be used as an approximation parameter similarly as the cluster count explained in the Section 4.5 to control the desired gain in the computation time and the error introduced due to the temporal coherence based simplification.

4.8 Summary

We developed a virtual environment for six degrees of freedom simulation of unmanned sea surface vehicles. We obtained real-time performance of the simulator using a simplification approach based on the clustering, temporal coherence, and parallelization approach to perform physics-preserving model simplification for the potential flow based six degrees of freedom, time domain simulation of USSVs. The average computation time was reduced by a factor of about 28.50 introducing an average error of about 5.8% with respect to the baseline computations, using the simplification techniques described in this chapter. The approach can take care of any arbitrary hull geometry as long as it can be expressed as a polygonal model. We established theoretically that the time taken for computing the forces



(a) *Variation of average force error with tolerance.*



(b) *Variation of average computation time with tolerance.*

Figure 4.7: *Variation of average error and average computation time with tolerance. (1500 time steps, $C = 45$, and $\epsilon = 10^{-3}$).*

increases linearly with the increasing cluster count and reduces linearly with the increasing tolerance, whereas the error introduced reduces with the cluster count in inverse square root fashion and increases with the tolerance in a linear fashion. The numerical simulation results holds well with these theoretical results. Cluster count and tolerance can thus be used as a high level approximation parameter in the simplification scheme to control the computation time to achieve a desired level of accuracy.

The potential flow assumption about inviscid fluid flow holds well as the boundary layer phenomenon is only observed in the close vicinity of the hull surface. Nevertheless, the phenomenon related to wave breaking, wake and turbulence cannot be taken into account using the potential flow theory alone. We accounted for these terms by introducing the damping and the added mass matrix in the dynamics model. The damping and added mass matrices can be determined by performing parameter identification for a given USSV model under a given sea state by experiments and field trials. A more general modeling however is required to cover wide range of sea states. Also, potential flow theory is based on displacement physics. The hydroplaning phenomenon observed in the boats moving at very high speeds cannot be modeled using the potential flow theory and requires use of a fluid flow model based on planing physics. The simplification approach based on clustering and temporal coherence is however, independent of any fluid flow model used and can be employed in any kind of fluid flow to improve the speed of computation.

Chapter 5
Generation of State Transition Models Using Simulations for Unmanned Sea Surface Vehicle Trajectory Planning

Trajectory planning for unmanned sea surface vehicles (USSVs) in high sea-states is a challenging problem. Large and somewhat stochastic ocean forces can cause significant deviations in the motion of the USSV. Controllers are employed to reject disturbances and get back on the desired trajectory. However, the motion uncertainty can be still high and needs to be accounted for during the trajectory planning to circumvent collisions with the obstacles. We model the trajectory planning problem of the USSVs as Markov decision process (MDP). A crucial element in MDP is the state transition probability, which in the case of USSV missions need to be evaluated on board as the environment may change significantly during the mission. Accurate and fast estimation of state transition probabilities is indispensable for generating effective trajectory plans. A key component of our approach is the simulation based estimation of transition probabilities from one state to another when executing an action. In this chapter[1], we present algorithms to generate state transition model using Monte-Carlo simulation of USSV motion. Our simulations are based on potential flow based 6-DOF dynamics. We also present model simplification based algorithm based on temporal coherence and its implementation on GPU to accelerate simulation computation performance. The computing speedup

[1] The contents of this chapter was published in ASME 2011 International Design Engineering Technical Conferences(IDETC) & Computers and Information in Engineering Conference (CIE)[TG11].

obtained by GPU parallelization and temporal coherence based model simplification enables fast computation of transition probabilities which is subsequently used in a stochastic dynamic programming based approach to solve the MDP. Using this approach, we are able to generate dynamically feasible trajectories for USSVs that exhibit safe behaviors in high sea-states in the vicinity of static obstacles. We tested the algorithms in simulation environment.

5.1 Introduction

Recently, USSVs began to be increasingly utilized as a promising tool in search, rescue, recovery, and surveillance operations. In most of the applications, the unmanned vehicles need autonomous motion planning capability to efficiently move between way points without colliding with the surrounding obstacles. The USSVs operate in the ocean environment with the disturbances caused by the ocean waves, currents, wake of other ships, etc. The ocean waves interact with the USSVs imparting uncertainty to their motion. Due to the high motion uncertainty, an action taken by the USSV may not lead to the exact desired motion despite of using a feedback controller. In addition to the dynamics based constraints, the motion uncertainty makes the task of the trajectory planning challenging particularly in highly cluttered environments.

In order to explain the influence of the vehicle's dynamics and the nondeterministic effect of the ocean on the planned trajectories, consider Figure 5.1, which shows an example of a marine environment. The circles M, P, and Q denotes three consecutive mission way points. When USSV reaches to the way point P, it needs to

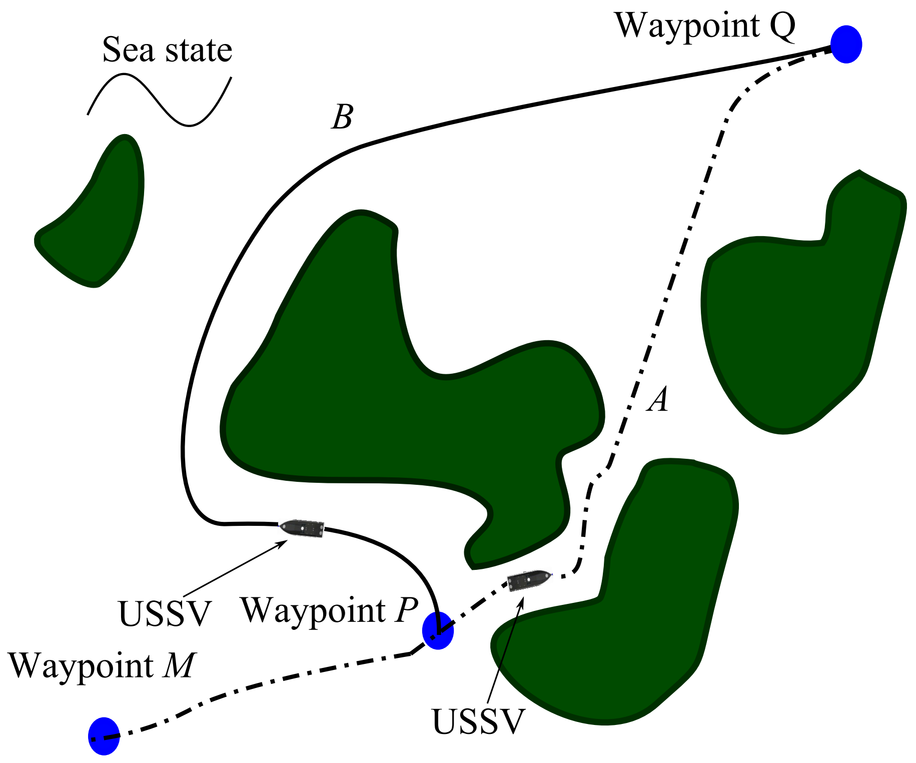


Figure 5.1: *Trajectories obtained with and without considering ocean wave disturbances (trajectory A is shorter but riskier and may lead to collision in the event of high sea-state whereas B is longer but more conservative to minimize the risk of collision).*

find the shortest possible trajectory between the way points P and Q with the minimum risk of collision with the static obstacles on the way. A rather naive approach to solve this problem would be to find the shortest possible collision free trajectory without explicitly considering the ocean disturbances (shown as trajectory A). This can be determined using gradient based optimization approach that minimizes the path length respecting the collision and dynamics constraints. With this approach, the high ocean disturbances may not allow the vehicle to closely follow the intended trajectory as the interaction of the USSV with the ocean waves might lead to a collision with an obstacle in the narrow region. On the other hand, the trajectory which is safe with respect to the ocean disturbances is shown in Figure 5.1 as trajectory B. Change in sea-state is usually determined using sensors and based upon that data during the mission execution suitable trajectory among A or B can be chosen.

In this chapter, we tackle the above outlined physics aware trajectory planning problem in highly uncertain ocean environments by combining the MDP [LaV06, Put94, TBF05] framework and a dynamically feasible motion primitive based state space representation [SMFH03, PKK09]. The MDP framework allows to naturally consider the motion uncertainties in the trajectory planning process [LaV06, TBF05]. In addition, we search for the trajectory in the space of dynamically feasible motion primitives that are discretized actions with a predefined trajectory length, in terms of MDP. The motion primitives were considered, for example, by Pivtoraiko *et al.* who developed lattice based representation for unmanned ground vehicles trajectory planning to obtain dynamically feasible trajectories in deterministic environments [PKK09]. The use of motion primitives in the MDP based framework, thus, allows generating trajectories that explicitly consider the constraints imposed by vehicle dynamics. This is unlike planning on a rectangular grid that might yield a dynamically infeasible trajectory.

The developed MDP based framework represents the uncertainty in the motion primitives by the use of state transition map. The state transition map maps each state of the vehicle and a given motion primitive to all possible resulting states with the respective probabilities of transitions. The state transition map can be obtained by either running field experiments or by using computer simulations. Field experiments, although, the most accurate method of determining state transition map are expensive and might be infeasible in some cases, when multiple sea-states and vehicle dynamics parameters need to be taken into account. Computer simulations are inexpensive and can be improved by incorporating experimental knowledge. The complexity of various USSV missions and the environments requires generating the state transition map on line, based on the information gathered by the sensors during the vehicle's operations. It may be infeasible to run all the possible simulations off line (before the mission) to generate the state transition map. This is because the sea-state is decided by several factors namely, amplitude, frequency, wave directions, and wave number of the wave components forming the waves. Each of these factors is continuous and has non-linear influence on the sea-state. In addition to this, the ocean interacts with the USSV in a non-linear fashion. The initial conditions for the simulations can thus become combinatorially prohibiting for an off line estimation of the state transition map for all possible sea-states. A potential flow theory based high fidelity 6 degree of freedom (DOF) dynamics simulator can generate fairly accurate state transition map and aid in generating both safe and low cost trajectories [KKF05, TG10, SS07]. The major problem with using such a high fidelity simulator is the computational performance of the simulation. This is mainly because of the fluid to rigid body interaction computation. One way to accelerate the simulation computations is by using computing parallelization. Performing parallel computations require computing clusters to be communicating over a network, which might be generally unreliable. Moreover, loading computing clusters on board might add to the weight of the payload and may not be desired. Another way for on board computing is GPU which is very powerful and lightweight.

In addition to improving the performance of state transition map computation by the use of GPU hardware, we employ model simplification techniques [TG10]. This allows even faster computation of the state transition map as may be required for some parts of the mission where computing gains available by using GPU may not be sufficient. By simplifying the physics of the simulated interactions, the simulations will get faster but less accurate. The accuracy of the computed state transition map depends on the available time for computation, and thus allows *anytime* trajectory planning capability. In Ref. [TG10], we developed a computational framework for real-time computation of 6-DOF motion of USSV under the influence of ocean waves. We were able to reduce the computation time by a factor of 28.5 with a mean square force computation error of 5.8% over the baseline computations. The reported model simplification techniques are modified to enable use of much accurate 6-DOF simulation model for the state transition map estimation instead of the less accurate 3-DOF simulation models.

The motion primitives and the dynamics simulations can be used to establish connectivity among the vehicle's states to develop a state transition map. Once the dynamically feasible state transition map of the USSV is generated, the vehicle's state space can be represented as a graph due to the connectivity information contained in the map. The trajectory planning problem can then be solved using the value iteration of the stochastic dynamic programming (SDP) [LaV06]. Posing a trajectory planning problem as MDP and using the value iteration to solve it, is by no means a new technique. Our main contribution in this chapter is threefold namely, (1) incorporation of the 6-DOF dynamics simulation in the formulation of the trajectory planning problem so that the computed trajectory plan satisfies the dynamics constraints as well as handles uncertainties, (2) use of GPU based parallelization schemes to make the simulation faster leading to on-line computation of state transition map, and (3) incorporation of model simplification techniques with GPU acceleration for computing state transition map even faster and thus allowing the *anytime* trajectory planning capability.

5.2 Problem Statement and Solution Approach

5.2.1 Problem Statement

Given,

(i.) a finite non-empty state space S,

(ii.) for each state $s \in S$ a finite non-empty action space $u(s)$,

(iii.) a dynamics motion model (based on potential flow theory) $\dot{x} = f(x, u, w)$ of the USSV where w is nondeterministic noise term,

(iv.) set of goal states S_G, and

(v.) obstacle map Ω such that,

$$\begin{aligned} \Omega(s) &= 1, \text{ if } s \text{ lies on obstacle} \\ &= 0, \text{ if } s \text{ is on free space,} \end{aligned}$$

Compute following:

(i.) State transition map over S: The state transition map should represent the motion uncertainty exhibited by the given motion model under each given action in u in the form of associated probability of transition for the corresponding state transition (probabilities of transition represented as $p(s_k|s_i, u_i)$, $\forall s_k,\ s_i \in S$, and $u_i \in u$). Perform GPU based computing acceleration and develop model simplification techniques for on-line estimation of the state transition map.

(ii.) Trajectory plan: Using the state transition map computed in step [i] determine trajectory plan to generate dynamically feasible trajectory in each planning cycle to reach the target locations in X_G from any given starting location of the USSV. The computed trajectory plan ensures that the generated trajectory is updated in every planning cycle to recover from the pose errors introduced due to the influence of the ocean environment. This kind of trajectory plan is also referred to as *feedback plan* in Chapter 8 of [LaV06]. We assume perfect state information is available at all times.

5.2.2 Approach

The approach is enumerated in the following steps.

(i.) Enhance the given USSV motion model to suit the requirements for GPU implementation. Implement the motion model on GPU and develop simplification algorithms to enable faster simulation.

(ii.) Model the trajectory planning problem as MDP by representing state-action space in a lattice data structure and compute the state transition map for the discretized action space.

(iii.) Apply value iteration of stochastic dynamic programming to determine the trajectory plan. The generated trajectory plan enables the USSV to find the optimal trajectory from each discretized state $x \in X$.

In the following sections we discuss the above steps in detail.

5.3 USSV Dynamics Simulation for Multiple Wave Components

In this section we present the governing equations of the implemented dynamics model which is used in the later sections dealing with the trajectory planning [TG10]. We extend the equations to handle the arbitrary number of wave components and to incorporate uncertainty into the system.

5.3.1 USSV Motion Equations Due to USSV and Ocean Wave Interaction

We implemented the 6-DOF dynamics model for the USSVs given by Fossen [Fos94]. In this model, the USSV is assumed to be a rigid body. The coordinate system used in the model is shown in Figure 4.2. The origin of the inertial frame of reference is set at the nominal water level with the Z-axis being vertical and pointing upwards. The body coordinate system for representing the hull geometry and the velocity directions of the USSV is attached to the USSV's center of gravity (CG).

We enhance the model presented in Chapter 4 for multiple wave components and nondeterministic noise in the ocean in this chapter. In the Equation 4.14, the term S_B is the instantaneous wet surface of USSV when hit by ocean wave represented by height field η composed of Q wave components given by the following equation.

$$\begin{gathered}\eta(x,y,t) = \sum_{j=1}^{Q} A_j \cos(k_j x \cos\theta_{w,j} + k_j y \sin\theta_{w,j} - \omega_j t + \psi_j) + \\ 0.5 A_j^2 k_j \cos(2k_j x \cos\theta_{w,j} + 2k_j y \sin\theta_{w,j} - 2\omega_j t + 2\psi_j)\end{gathered} \tag{5.1}$$

where,

$A_j, \omega_j, k_j, \theta_{w,j}$ are the amplitude, frequency, wave number, and wave direction respectively for j^{th} wave component, and

$\psi_j \in [0, 2\pi)$ uniformly random phase lag term.

The velocity field ϕ is computed using following equation.

$$\phi = \sum_{j=1}^{Q} \frac{gA_j}{\omega_j} \exp(k_j z) \sin(k_j x \cos\theta_{w,j} + ky \sin\theta_{w,j} - \omega_j t + \psi_j) \tag{5.2}$$

The term F_P on the right hand side of Equation 4.15 is the force due to the actuation (the thrust and the rudder angle). There are many actuator models available for the USSVs and can be plugged in to Equation 4.15 [Fos94, KKF05]. We used a model given in Equation 5.3.

$$F_P = [K_1 \parallel n_{prop} \parallel n_{prop},\ 0,\ 0,\ 0,\ 0,\ K_2 * K_1 \parallel n_{prop} \parallel n_{prop}\theta_{rud}]^T \tag{5.3}$$

where K_1 is a constant and we chose it to be 1000, n_{prop} is the propeller's rpm, K_2 is a constant and we chose it to be 10, and θ_{rud} is the rudder angle.

We can express the parametric form [LaV06] of the 6-DOF model as follows:

$$\dot{x} = f(x, u) \tag{5.4}$$

where $x = \left[p^T \; v^T\right]^T$ is the state of the USSV, $u = [v_f \; \Theta]^T$ is the commanded control action to go with forward velocity of v_f at a heading angle of Θ, and $f = \left[(M_H^{-1}(-C_H(v)v - D_H(v)v - g(p) + F_E + F_P)^T \; J_p(v)^T\right]^T$.

We specify the desired control action using the desired forward velocity and the heading angle. Any other kind of desired control actions such as the desired angular velocity can also be chosen based on the available actuation models. For the purpose of this chapter we assume that the USSVs are controllable using the control actions specified by v_f and Θ.

The thrust n_{prop} and the rudder angle θ_{rud} can be computed using the PID controller given in Equation 5.5.

$$n_{prop} = K_{p,n}e_n + K_{i,n}\int e_n dt + K_{d,n}\frac{de_n}{dt} \tag{5.5}$$

$$\theta_{rud} = K_{p,\theta}e_\theta + K_{i,\theta}\int e_\theta dt + K_{d,\theta}\frac{de_\theta}{dt} \tag{5.6}$$

where e_n is the difference between the desired forward velocity v_f and the actual instantaneous forward velocity, e_θ is the difference between the desired heading angle Θ and the actual heading angle, and K's are the respective PID gains.

We chose the PID controller because of its widespread use, however, in order to execute the commanded control actions any other controller such as the back-stepping, sliding mode, etc. can be used [AMM10].

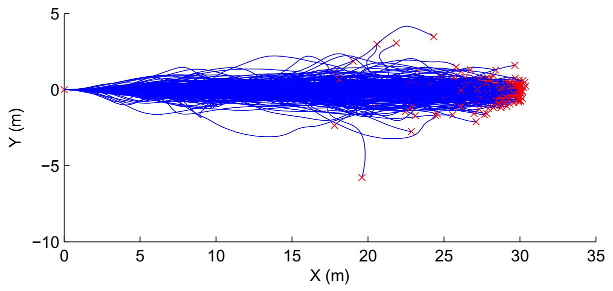
5.3.2 Uncertainty in the USSV Motion Model

In Section 5.3.1 the ocean wave (the ocean wave height and the velocity field) was initialized with given ocean wave parameters. An ocean wave, once initialized evolves deterministically over the time and can be predicted exactly using the solution of Laplace equation (see Equation 5.1) [New77]. However, the ocean waves initialized with the same parameters might look different due to the presence of the phase lag (between each wave component) parameters (ψ_j)'s which are uniform variates. This leads to the prediction of slightly different trajectories in each simulation run of the USSV operating under the ocean waves with exactly identical ocean wave parameters (initialized with uniform random phase lags) and an action. This effect is shown in Figure 5.2, in which the USSV is acted upon by a PID controller to move along a straight line for 256 different simulation runs (ocean wave component phase lags for each simulation run is randomly initialized). The variation in the trajectories of the USSV in each simulation run is due to the uncertainty introduced by random phase lags in the ocean wave components despite of the other ocean parameters and the PID control objective being exactly identical. Figure 5.2(b) shows the histogram of final positions reached by the USSV when commanded to reach at $(30, 0)$ with an orientation of 0^0 due to the disturbance caused by the ocean waves. Figure 5.2(c) shows the variation in the final orientation while the commanded action was 0^0. It should be noted that the variation in the ending pose is one of the main cause of randomness in the motion model which accumulates over the trajectory.

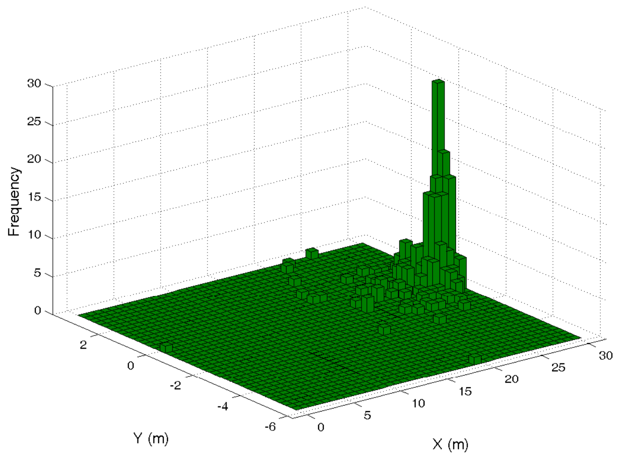
Formally, we can express the parametric form of the dynamics model (with uniformly random initial phase lag parameters) as follows:

$$\dot{x} = f(x, u, w) \tag{5.7}$$

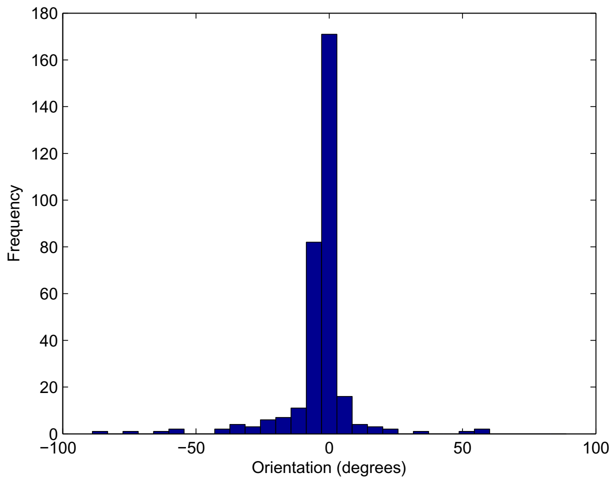
where w is the noise introduced due to the uniform random phase lags ψ_j.



(a) *Sample trajectories.*



(b) *Frequency distribution of ending USSV positions for sample trajectories.*



(c) *Frequency distribution of ending USSV orientations for sample trajectories.*

Figure 5.2: *Uncertainty in USSV motion model for action along X axis generated using sample size of* 256 *(computed for sea-state 4 with average ocean wave height of 1.8 m).*

5.3.3 Simulator Implementation on GPU

As described in Section 5.3.2 and shown in Figure 5.2, the USSV ends up in different poses for exactly identical action objective and initial states for different phase lag initializations. This means that for sufficiently large number of simulation runs with uniform phase lag initializations for a given set of initial conditions and action goal, the distribution of final states will represent the influence of the ocean on USSV motion. In this section, we describe Monte-Carlo simulation based approach to estimate the influence of nondeterministic effects of ocean on USSV motion for a given set of action goals. The Monte-Carlo sampling based approach requires running numerous dynamics simulations with uniformly random initial phase lags among wave components and hence real-time performance of the simulator may not be enough. We describe the modifications made in the simulation model [TG10, SS07, KKF05] for suitability of implementation on GPU.

We define state vector tuple $X = (x_1, x_2, .., x_M)$, where $x_k = [v_k^T x_k^T]^T$ is state vector of k^{th} simulation run and M is the number of Monte-Carlo simulation runs.

Let $U = (u_1, u_2, ..., u_M)$ be the action vector tuple with each element as a vector specifying action for corresponding simulation run. We denote action set for the entire sample using the tuple $\Upsilon = (U_1, U_2, ..., U_P)$ where U_j's are action vector tuples and P is the number of action goals.

Also, let $W = (w_1, w_2, ..., w_M)$ be the phase lag vector tuple where $w_k = [\psi_{1,k}, \psi_{2,k}, ..., \psi_{Q,k}]^T$ is the phase lag vector for k^{th} simulation run and $\psi_{q,k}$ is phase lag of q^{th} wave component of k^{th} simulation run.

Thus, the augmented dynamics equation can be written by generalizing Equation 5.7 for M Monte-Carlo simulation runs as follows:

$$\dot{X} = F(X, U, W) \tag{5.8}$$

where, F is the modified dynamics function representing simultaneous simulation runs. Let $F_{WT} = (F^T_{W,1}, F^T_{W,2}, ..., F^T_{W,M})$ be the ocean wave force tuple where $F_{W,k}$ is ocean wave force vector for k^{th} simulation run.

The computation steps of the simulation are enumerated below [TG10].

Algorithm 5.1 - GPU based Monte-Carlo Simulation of USSV dynamics

Input

(a.) Initial state vector tuple of USSV X_0,

(b.) number of Monte-Carlo runs M,

(c.) desired target state vector tuple X_t,

(d.) desired trajectory length l,

(e.) radius of acceptance r,

(f.) number of ocean wave components Q,

(g.) time step size Δt,

(h.) action set Υ,

(i.) polygonal geometry of USSV, and

(j.) sets of amplitudes A_p, frequencies ω_p, directions θ_p corresponding to each wave component where index p varies from 0 to $Q-1$,

Output Set of M trajectories

Steps

(i.) Initialize the state vector tuple of USSV $X = X_0$, phase lag tuple W, time $t = 0$, and trajectory length vector $L = [0, 0, ..., 0]^T$.

(ii.) Transform the USSV geometry to M states represented by X. Each transformation is performed by separate GPU thread. In this case, same instruction of transformation needs to operate on MN similar data, where N is number of polygonal facets representing the USSV geometry. We perform computations of this step on GPU.

(iii.) Determine the instantaneous wet surfaces ($S_{B,j}$) of the USSV by finding out the facets lying beneath and on the wave surface (computed by superimposing given Q ocean wave components using Equation 5.1) corresponding to j^{th} phase lag vector w_j and use Equation 4.14 to compute the wave force tuple F_{WT}. In this case computation of intersection of each polygonal facet with instantaneous ocean wave and force computation is performed by separate GPU threads. The number of independent operations required is again MN. We perform computations of this step on GPU.

(iv.) Determine the required control force vector tuple corresponding to the action set Υ using Equations 5.3, 5.5, and 5.6.

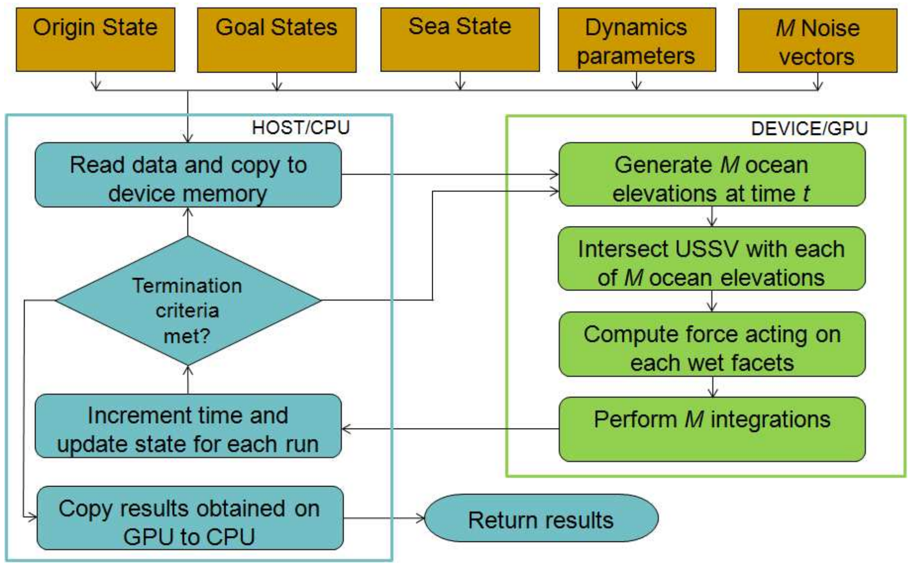
(v.) Determine the Coriolis matrix ($C_k(v)$) and the damping matrix ($D_k(v)$) corresponding to each Monte-Carlo simulation run. The number of independent operations required in this step is M. We perform computations of this step on GPU.

(vi.) Use Euler integration to solve Equation 5.8 by using the wave force tuple, Coriolis matrix, and damping matrix. Update time t to $t + \Delta t$. The number of operations needed in this step is M. We perform computations of this step on GPU.

(vii.) Find Euclidean distance ΔX between state tuple obtained from step (vi) and X and update trajectory length vector L with $L + \Delta X$. Compare each element of L with the desired trajectory length l. The Monte-Carlo runs for which trajectory length exceeds the set trajectory length l, do not update corresponding elements of X whereas for other runs update the elements in X with the solution found in step (vi). Since this is a step with logical branching we perform it on CPU.

(viii.) If trajectory lengths for all the runs exceeds l or all USSVs are within the radius of acceptance r from the respective target positions then return M trajectories else go to step (ii).

The computations performed on CPU and GPU are illutrated in Figure 5.3. We used NVIDIA's CUDA software development kit version 3.2 with Microsoft Visual Studio 2008 software development platform on Microsoft Windows 7 operating system for the implementation of Algorithm 1. The graphics hardware used



Origin State
Goal States
Sea State
Dynamics parameters
M Noise vectors
HOST/CPU
Read data and copy to device memory
Termination criteria met?
Increment time and update state for each run
Copy results obtained on GPU to CPU
Return results
DEVICE/GPU
Generate M ocean elevations at time t
Intersect USSV with each of M ocean elevations
Compute force acting on each wet facets
Perform M integrations

Figure 5.3: *Implementation of USSV Simulator on GPU.*

was NVIDIA GeForce GT 540M mounted on Dell XPS with Intel(R) Core(TM) i7-2620M CPU with 2.7GHz speed and 4GB RAM. We chose the number of CUDA threads per block to be 256 for each kernel function. For the CUDA kernel function needed to work on J data members, we chose the number of computing blocks to be $\frac{MN+T-1}{T}$. The number of triangular facets in the USSV model used in the simulations was 11158 and the bounding box dimensions of the model was $12 \times 4 \times 4$ m. We chose ocean wave composed of $Q = 20$ components with six components having amplitude of 0.2 m while rest fourteen with amplitude of 0.1 m. sixteen of the ocean wave component had frequency of 1 Hz while four of them had frequency of 2 Hz, and the direction θ_w's were evenly distributed in the range 0 to 2π radians. We chose simulation time step of size 0.05 s and ran the simulation for 200 time steps for 256 random phase lag initializations. Table 5.1 shows the comparison of the computational performance on GPU as compared to the CPU based computation. The OpenMP based multi-threading enabled 85% average CPU usage while running

Table 5.1: *Comparison of the computation gain due to GPU over the baseline computations performed on CPU*

M	Baseline computation time on CPU (s)	GPU Computation time (s)	Speedup
1	13.71	3.60	3.81
2	27.08	5.25	5.16
4	54.16	8.00	6.77
8	107.35	13.00	8.26
16	214.4	22.01	9.75
32	425.61	36.91	11.53
64	846.56	65.40	12.94
128	1691.18	123.80	13.66
256	3386.32	241.90	14.00

the baseline simulations. The GPU based approach resulted into speedup by factor ranging from 3.81 to 14.00 for the presented test case. Table 5.1 also shows that the speedup factor increases with increase in the number of Monte-Carlo runs because of the highly data parallel nature of the computations. All the results under the same conditions were identical to the CPU based baseline computations.

5.3.4 Model Simplification on GPU

Almost more than 99% of the computation time in the USSV simulation is spent in computing the forces acting on the USSV due to the ocean waves [TG10]. The ocean is represented as a spatio-temporally varying heightfield in the simulation model we are using in this chapter. One of the major factors that influences the forces due to the ocean waves is the variation in the ocean wave heightfield besides the fluid velocity around the USSV. The ocean wave heightfield does not change significantly with each simulation time step. For example, for a simulation time step of length 0.05 s, the possibility of ocean wave heightfield around the USSV changing significantly is very low. In such a situation one can utilize the force computed in the previous time step in the current time step of the simulation to save some computational effort. This is the underlying idea behind temporal coherence. In order to explain the temporal coherence idea in a more concrete way, we introduce and define *instantaneous ocean heightfield* as follows.

Definition 17 Let the ocean wave be specified by Q components consisting of amplitudes, frequencies, and directions. Let state vector tuple X denote the instantaneous states of the USSV for each Monte-Carlo simulation run, B_j be the bounding boxes of the USSV located at positions given by X and rectangles $R_{B,j}$ be the projections of B_j on the XY plane. Let Λ_j denote uniform grid of size $m \times n$ on $R_{B,j}$.

We define the instantaneous ocean wave height-field G as a $(Q \times mn)$ sized matrix G, such that the rows of G are the vectors made up of ordered elevations of the ocean wave at the mn grid points on Λ_j.

Definition 18 For a pair of ocean wave height-fields G_1 and G_2, we define the *heightfield distance vector* $\vec{h}_d$ between G_1 and G_2 as the following row-wise second order norm.

$$\vec{h}_d = \|G_{1,j} - G_{2,j}\| \tag{5.9}$$

where $G_{1,j}$ is the j^{th} row of G_1, and index j denotes Monte-Carlo run from 1 to M.

The force need not be computed in a simulation time step if the ocean wave heightfield distance around the USSV from the previous simulation time step is not significant. The temporal coherence test is performed as an additional operation in step (ii) of Algorithm 1 (described in Section 5.3.3). If it is found that the ocean wave heightfield distance corresponding to at least one Monte-Carlo run has changed significantly then step (iii) of the Algorithm 1 is performed else step (iii) is skipped and step (iv) is directly executed. By this, the execution of step (iii) in Algorithm 1 is avoided which introduces some simplification error but reduces computation time.

The steps for performing temporal coherence based model simplification on the GPU are described below.

Algorithm 5.2 - Temporal coherence based Model Simplification

Input

(a.) State vector tuple X,

(b.) number of Monte-Carlo runs M,

(c.) number of rows (m) and columns (n) of grid,

(d.) simulation time t and time step size Δt,

(e.) threshold τ for heightfield, and

(f.) threshold $d\tau$ for differential of heightfield.

Output Decision about whether to perform force computation in the next time step or reuse force computed in the earlier time step

Steps

(i.) If $t = 0$ return decision to perform force computation.

(ii.) If $t = \Delta t$ then initialize heightfield G_p and differential heightfield dG_p to a null matrix of size $M \times mn$ and store in global memory.

(ii.) Compute ocean wave heightfield G at time t and then compute differential heightfield $d_G = G - G_p$.

(iii.) Compute heightfield distance $\vec{h}_d$ vector between G and G_p.

(iv.) Compute differential heightfield distance $d\vec{h}_d$ between dG and dG_p.

(v.) If all the elements of $\vec{h}_d$ are less than τ and if all the elements of $d\vec{h}_d$ are less than $d\tau$ return the decision to reuse previous value of force else update $G_p = G$ and $dG_p = dG$ and return the decision to recompute force.

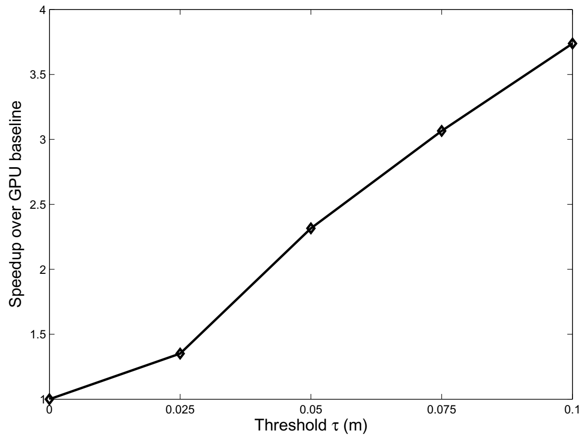
We chose $m = 2$, $n = 5$, $M = 256$, and $d\tau = 0.1$ and performed the simulations under the identical ocean and USSV parameter settings and varied τ from 0.00 to 0.10 in the increments of 0.025. The computation speedup factor over the GPU baseline computation time (when $\tau = 0.00$) varies from 1.04 to 3.61 depending on the set threshold τ as shown in Figure 5.4(a). The temporal coherence based simplification introduces errors in the computation of the final pose of the USSV in Monte-Carlo runs. Figure 5.5 shows the variation in the Euclidean distance of final positions of USSV from the nominal point and the difference of each final USSV orientations from the nominal orientation obtained by the Monte-Carlo simulation runs. The nominal pose is the commanded pose to which USSV should reach if there are no disturbances. In the test case, the nominal point is $(30, 0)$ and the nominal orientation is 0 radians. The variation in distance and orientation difference increases with the increasing threshold. Also, the number of outliers in each case increases with the increasing threshold. This is because, more the threshold lesser the number of times forces are computed and hence more will be the inaccuracy and lesser will be the computation time. We chose fixed randomization of the ocean wave for evaluating the plots in order to prevent the influence of randomization on the computing time and the variation of the pose errors.

It should be noted in Figure 5.4, that the computing gains increase slower in the range $0 < \tau < 0.025$ because for smaller threshold algorithm is unable to reuse force values computed in the previous steps and owing to the same reason the variation in the final pose and nominal pose is also comparatively less pronounced. At larger values of threshold $\tau > 0.075$, the variation in the distance and orientation increases rapidly. It can thus be concluded that τ can be varied in the window of $0.025 < \tau < 0.075$ for obtaining computational gain at the expense of acceptable errors.

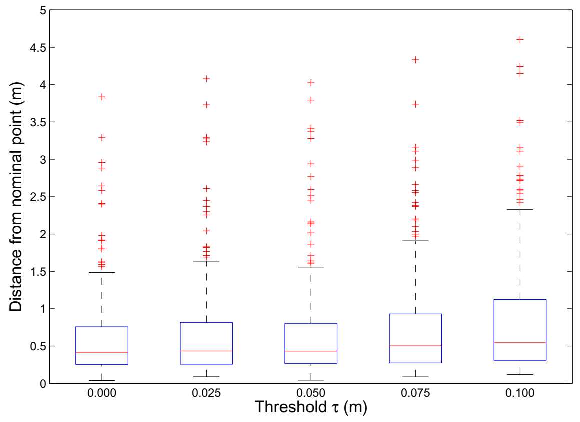
Figure 5.5 compares the computational gains obtained using temporal coherence on the GPU and CPU based simplification approaches (see Chapter 4) with the baseline computed on CPU (see Table 5.1. The main observations and respective analysis from the Figure 5.5 are explained below.

(i.) The computational speedup factor obtained using temporal coherence on GPU is in the range of 4.7 to 43.1 and increases with the number of Monte-Carlo runs. The increase in the speedup can be attributed to the fact that the main computational cost of GPU operations is the data transfer between GPU and CPU. In the USSV simulation, the data of state variables and the system matrices need to be transferred from GPU to CPU which is a constant time operation. When the number of runs is less, the appropriated compute time is larger whereas for large number of Monte-Carlo runs the cost of memory transfer reduces and hence the speedup factor increases.

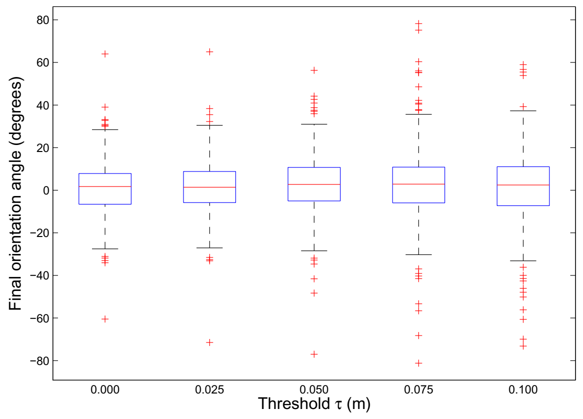
(ii.) The computational speedup factor due to temporal coherence over GPU baseline ranges from 1.2 to 3.1.



(a) *Computing speedup over GPU baseline vs threshold.*



(b) *Variation of distance from nominal point vs threshold.*



(c) *Variation of orientation from nominal point vs threshold.*

Figure 5.4: *Model simplification results of GPU based temporal coherence.*

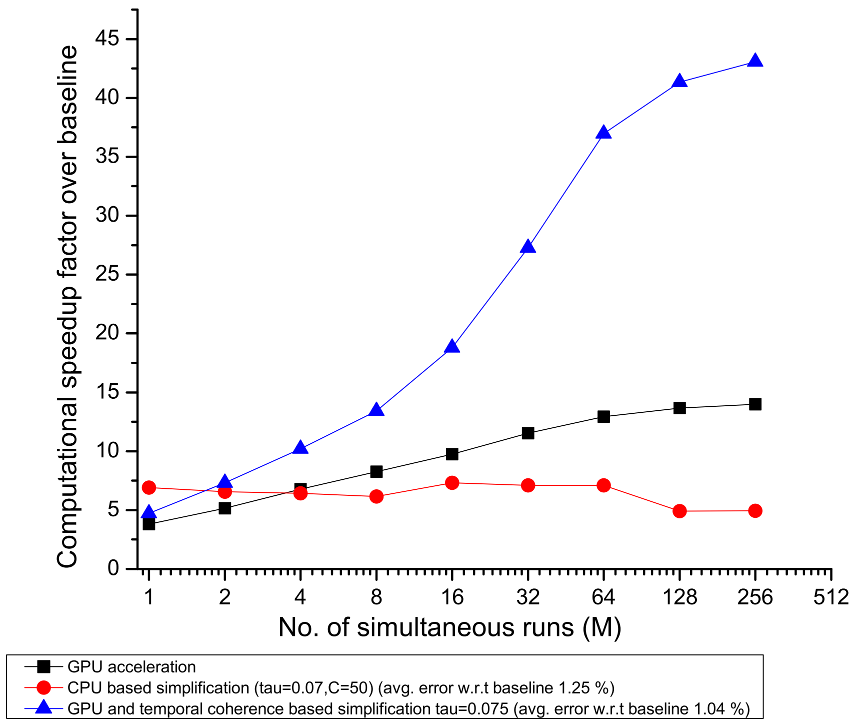


Figure 5.5: *Results of GPU acceleration.*

(iii.) The error associated with the model simplification performed on GPU and CPU is computed by taking the mean squared percentage error between the time series of the computed forces using the simplified method and the baseline method (see Equation 4.27 [TG11, TG10]. The model simplification based on temporal coherence implemented on GPU led to an error of 1.04% for the threshold $\tau = 0.075$ and $d\tau = 0.1$. Also, the figure shows variation of computational speedup using model simplification algorithms based on clustering and temporal coherence on CPU (see Chapter 4) for simplification parameters $C = 60$, $\tau = 0.07$, and $d\tau = 0.1$. Model simplification performed on CPU leads to an average factor of speedup of 6.4 and average force error of 1.25% over the CPU baseline. It can be seen in Figure 5.5 that for single run the CPU based simplification approach outperforms the purely GPU based approach by a factor of 1.8 and for two runs the CPU based simplification approach is better than purely GPU based approach by a factor of 1.3. Again the reason is the appropriation of computing time spent on the constant time data transfer operations over larger number of runs. It is thus evident that using purely GPU based approach may not be enough in applications in which a single USSV needs to be run in a VE as model can become more complex stretching the GPU to its limits. In applications where some error is tolerable, model simplification can significantly speedup the application at the cost of small errors.

(iv.) For larger number of runs, which are pertinent to applications such as transition probability estimation, GPU based approach gives very high speedup factor (in case of Figure 5.5 about 43.1 with average mean square force error of 1.04%.

(v.) Figure 5.5 also shows that the computing speedup due to temporal coherence gradually saturates as the number of simultaneous runs increases. The reason is that the possibility of many heighfields being less than the threshold simultaneously reduces as the number of heightfields increase.

5.4 Application of Generated State Transition Map in USSV Trajectory Planning

In this section we present the application of state transition probabilities obtained from model simplification techniques developed in Section 5.3 in the area of

USSV trajectory planning.

5.4.1 MDP Formulation

In this section we present the MDP formulation for the USSV planning problem and the algorithms to compute the various components of the MDP.

5.4.1.1 State-Action Space Representation

For the dynamics computations, the state of the USSV is defined as an augmented vector of the pose and the velocity according to Equation 5.7. The sizes of the vectors representing the pose and the velocity are six each, making the size of the state vector as twelve. In addition, the USSV state-action space is continuous and it is very difficult to search an optimal policy in such a high dimensional and continuous space. Some of the properties of the problem at hand make it possible to simplify the state-action space representation. In this section we present the state-action space dimension reduction and a suitable discretization to pose the problem of the USSV trajectory planning as a MDP.

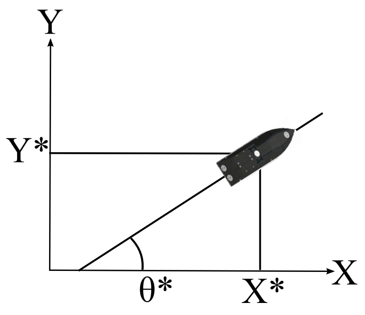
The motion goals for the USSV are usually specified in terms of the target pose $[x,\ y,\ \theta]^T$) and the target velocity $[v_x,\ v_y,\ \omega_z]^T$ of the USSV on the ocean's nominal water plane. This means that the state space for the MDP can be reduced to the size six as $[x,\ y,\ \theta,\ v_x,\ v_y\ \omega_z]^T$. This is because the motion is computed using the 6-DOF simulator and the influence of the motions in all the degrees of freedom is taken into account for the purposes of determining the instantaneous location and the orientation on the nominal water plane of the ocean. The motion of the boat computed by the 6-DOF simulator is used to determine the transition model for the MDP. The state transition map is then used by the stochastic dynamic programming (SDP) solver to determine an optimal trajectory. We can thus ignore the heave, roll, and the pitch in the description of the state space for the MDP to simplify the planning problem. Further, the desired velocity of the boat is assumed to be constant for the purposes of this chapter. In most of the tasks to which the USSVs are subject to, the velocities do not change significantly during the operations and that justifies the choice of the constant desired velocity of the boat. During the simulations we found that the velocity remains within about 8% limits with a hand tuned PID controller for the chosen set of control actions under the sea-state 3 (wave height varying between 0.5 to 1.25 m). Also we tuned the PID controller such that the angular velocity is kept within a set bound. We chose the circle of acceptance to be 1.0 m, which can be varied based on the vehicle dynamics [Fos94]. This makes it possible to ignore the angular velocity ω_z of the boat from the MDP state representation. The velocity in the sway direction and the angular velocities in the roll and the pitch directions are generally very small and can be ignored in the MDP state space. Nevertheless, the transition model computation is performed using all the twelve degrees of freedom. The state space for the planning purposes in this chapter is thus reduced to a 3-tuple given by Equation 5.10.

$$s = [x \; y \; \theta]^T \tag{5.10}$$

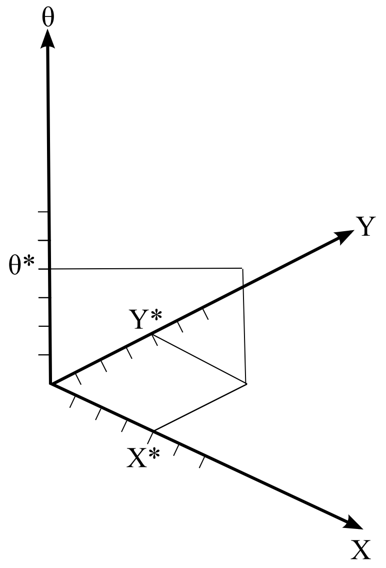
where x, y are the coordinates of the USSV's center of gravity in the XY plane, and θ is the orientation of the boat in the XY plane measured counterclockwise about the Z axis.

The bounding box dimensions for a typical USSV is around $12 \times 4 \times 4$ m. In order to have sufficient granularity in the state space, the grid's resolution overlaid on the XY plane should be equivalent to the USSV length (nearly equal to 12 m). We chose the grid dimension to be 15.0 m. The orientation discretization is chosen to be 0.524 *radians*. The state space is depicted in Figure 5.6, in which the XY plane contains the location of the center of gravity and the θ axis denotes the orientation of the boat. Pose of the boat in the XY plane is shown in Figure 5.6(a) and the corresponding location in the 3-D state space is shown in Figure 5.6(b).

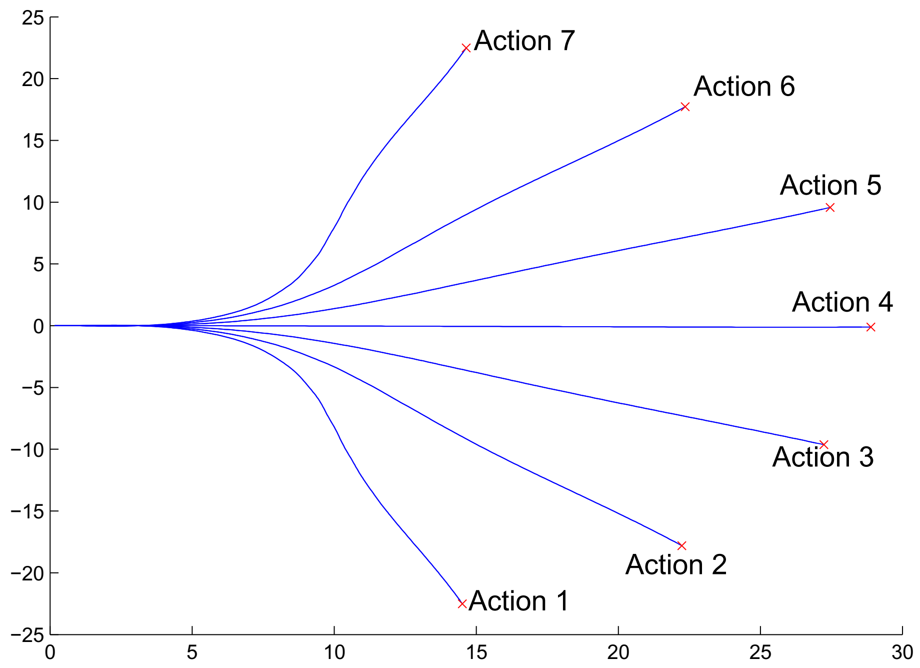
The action space is discretized as a set of relative pose commands from an initial state. We chose the set of relative final pose as seven radial pose vectors having radial distance of 30.0 m (in terms of the trajectory length of the boat) with final desired steering angles varying from -2.142 *radians* to 2.142 *radians* in the increments of 0.428 *radians*. We chose the desired path lengths and the steering angles by first running the boat for 10 s along the polar directions so that the boat can cover as many surrounding grids as possible. Using this technique we generated a set of seven way points which sufficiently cover the space around the boat and are dynamically reachable in about 10 s. We then tuned the PID controller so that the boat can reach those locations. An important implementation detail is that during the event of a high sea-state making the USSV's angular velocity uncontrollably high, the PID controller is switched to stabilize the USSV and once its angular velocity comes within set bounds, the PID is switched to position tracking again. The discretized set of actions are shown in Figure 5.6(c) and are also known as the



(a) *Typical pose of USSV in XY plane.*



(b) *Location of USSV in state space.*



(c) *Action discretization.*

Figure 5.6: *State-action space of MDP.*

control set [PKK09].

5.4.1.2 Determination of State Transition Map

The state transition map for the continuous space was expressed in the form of Equation 5.7. In this section we shall describe the computation scheme to determine the state transition map for the given USSV in the discrete state space. We first present the definition of the state transition probability.

Definition 19 Given an initial state x_t and an action u_t at time t, the probability $p(x_{t+\Delta t}|x_t, u_t)$ of ending up in the state $x_{t+\Delta t}$ is called the state transition probability. We assume that the time taken to execute the action u_t is Δt.

Figure 5.2 shows a USSV situated initially in the state x_t. When an action u_t is applied to it for 256 sample runs, the trajectories traced by the USSV are shown by the blue lines. The USSV traces a slightly different trajectory for each sample run due to the ocean wave and USSV interaction force as explained in the previous section (see Equation 4.14). The variation in the resulting states for a given initial state and an action yields a probability distribution over the resulting states. The probability of reaching to an arbitrary resulting state $x_{t+\Delta t}$ is shown in Figure 5.2.

For the deterministic dynamics, the researchers have suggested the use of the state lattice [PKK09]. A state lattice is a graph where nodes represent the states and the arcs represent the connectivity between the states that respects the deterministic dynamics constraints. This kind of graph representation makes use of the well known graph search algorithms like A*, Dijkstra, D*, etc. to solve the robot motion planning problems. We use a similar data structure to consider the dynamics by embedding the information of the state transition probabilities in the arcs of the state lattice and then make use of the stochastic dynamic programming to solve the problem of trajectory planning under the environmental uncertainties.

The steps to compute the state transition map are described below.

Algorithm 5.3 - State transition map computation

Input

(a.) Set of trajectories $T = (T_1, T_2, ..., T_P)$ for a given action set $\Upsilon = (U_1, U_2, ..., U_P)$ using Algorithm 1 and 2,

(b.) List of discretized states $S = [s_1, s_2, ..., s_L]^T$ representing the region of USSV operation.

Output State transition map.

Steps

(i.) Perform geometric transformation of each trajectory in T to the poses represented by each states in S. Figure 5.7 shows a portion of the state space with the rectangular grids representing region on the nominal water plane, while the layers representing the orientation of the boat varying from 0 to 2π radians. The USSV when begins from state s_i and takes action u_j can end up in multiple states shown by the blue crosses based on the trajectories T. State s_e represents the nominal ending state under the action u_j when there is no environmental disturbance.

(ii.) Construct graph by connecting the states (nodes) reachable from another states (nodes) using the sample actions (arcs) in T.

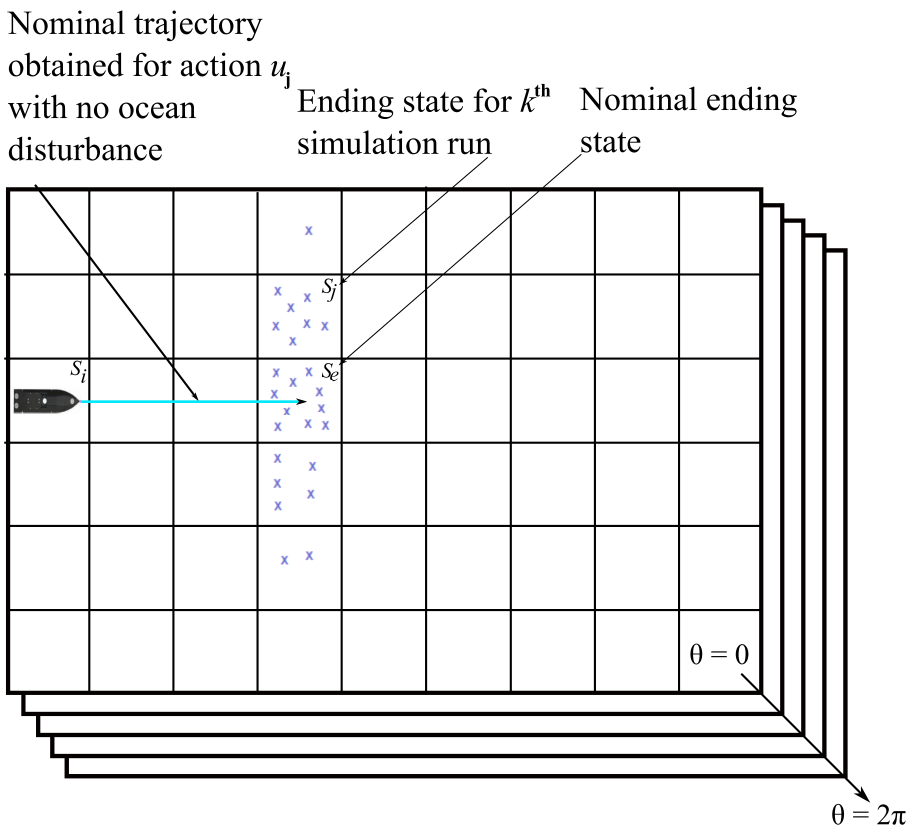

Nominal trajectory obtained for action u_j with no ocean disturbance
Ending state for kth simulation run
Nominal ending state
S_j
S_i
S_e
$\theta = 0$
$\theta = 2\pi$

Figure 5.7: *State transition map computation.*

(iii.) Determine the transition probabilities by finding out the ratio of the number of connections between the two connected states and the total sample count of the actions taken. In Figure 5.7, let the state s_j be connected to the state s_i for $n(s_j$ times out of N sample runs. Compute the probability p_{ij} of transition from state s_i to state s_j using $p_{ij} = \frac{n(s_j)}{N}$.

(iv.) Return the state transition map.

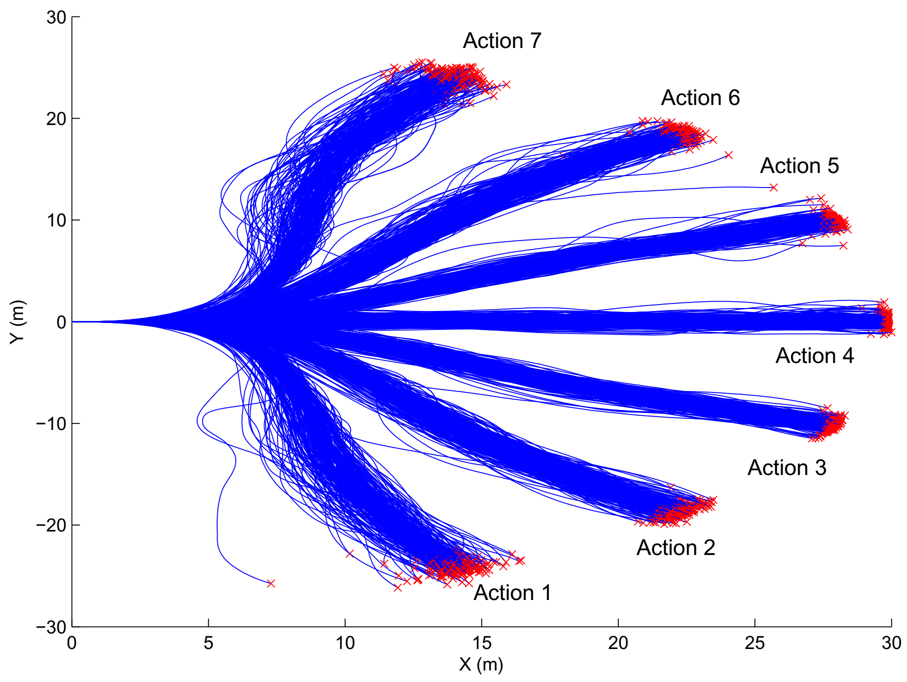
In this way the state transition map is obtained, whose nodes are the states and the arcs are the connectivity that satisfies the dynamics constraints. Each connection has a probability associated with it due to the system dynamics and the presence of the uncertainty due to ocean waves in the system as described before. It should be noted that the maximum number of children nodes of a given parent node for a state space with L nodes can be up to L based on the variations in the samples of the action and level of uncertainty in the environment. This makes the suggested data structure very flexible in the sense of capturing the dynamics constraints and the environmental uncertainty for extreme situations such as rough sea-state.

5.4.1.3 Reward or Cost Function

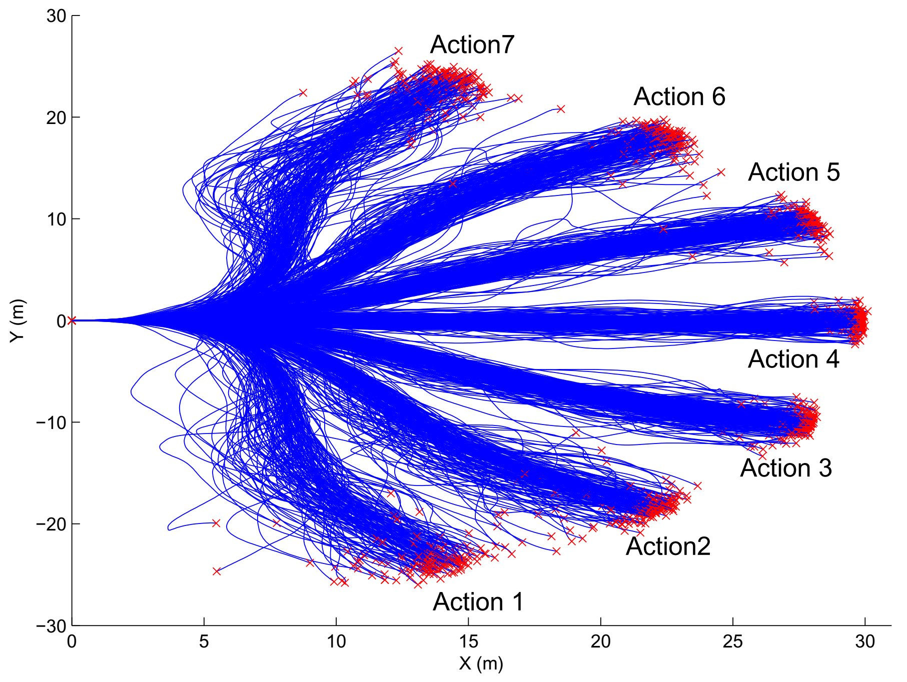
The final element of MDP is the immediate reward for transitioning from a given state to another state by taking an action. The time spent by the USSV to perform an action can be determined by the length of the trajectory traversed by it. This entails that larger the length of the trajectory, smaller should be the reward for the action generating the trajectory, and thus we should consider the negative value of the trajectory length as the reward. We chose the negative of the average length of the S trajectories traversed by the USSV for each action in the control set to determine the reward.

5.4.2 Results and Discussion

For the given action set Υ described in Section 5.4.1.1 we determine the set of trajectories T using Algorithm 1 and 2 for the sea-states 3 and 4. The average wave height for the sea-state 3 ranges from 0.5 m to 1.25 m whereas for the sea-state 4 the average wave height ranges from 1.25 m to 2.5 m [Fos94]. The resulting set of trajectories for 256 Monte Carlo simulation runs are shown in Figure 5.8. The variations in the trajectories for the sea-state 4 is larger compared to that for the sea-state 3 due to the higher average ocean wave height. We chose sample size of



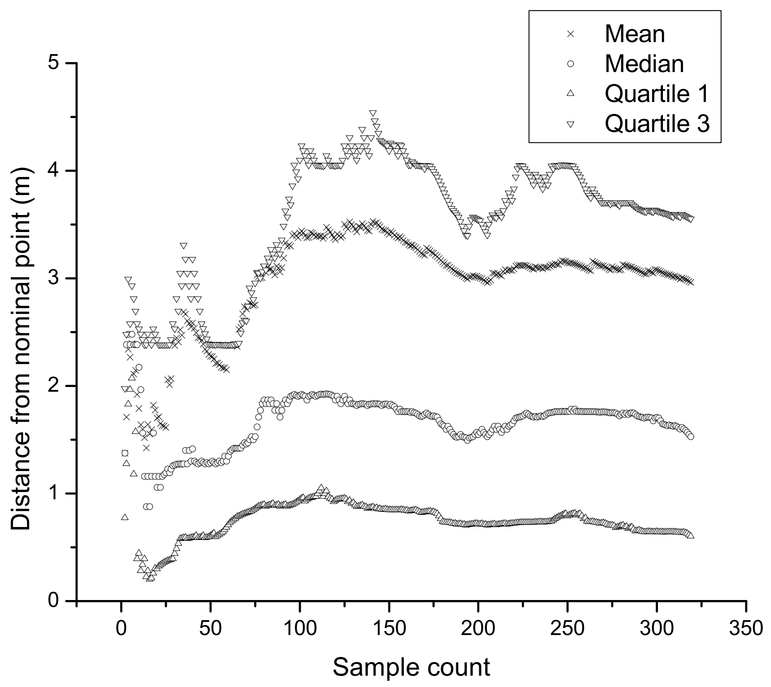
(a) *Computed at sea-state 3.*



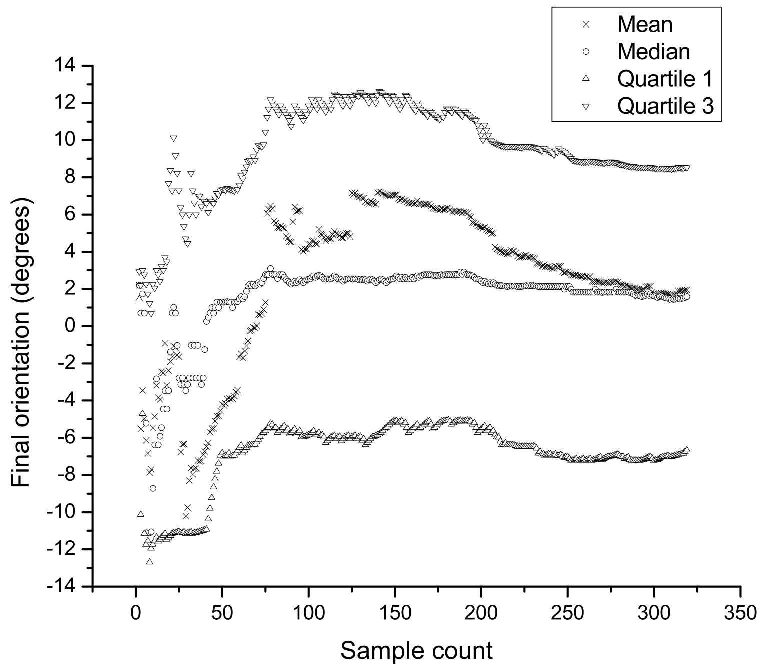
(b) *Computed at sea-state 4.*

Figure 5.8: *Control set computed for different sea-states.*

256 because the variability of the pose data can be captured using a sample size of around 256. This is illustrated in Figure 5.9.



(a) *Variation in sample median and quartile distance from nominal position.*



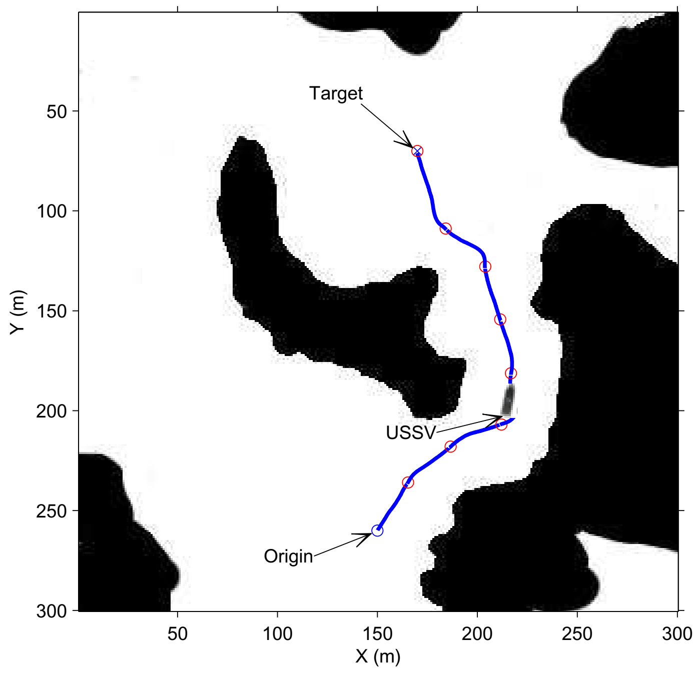
(b) *Variation in sample median and quartile orientation from nominal orientation.*

Figure 5.9: *Variation in sample distance and orientation from nominal pose vs sample size.*

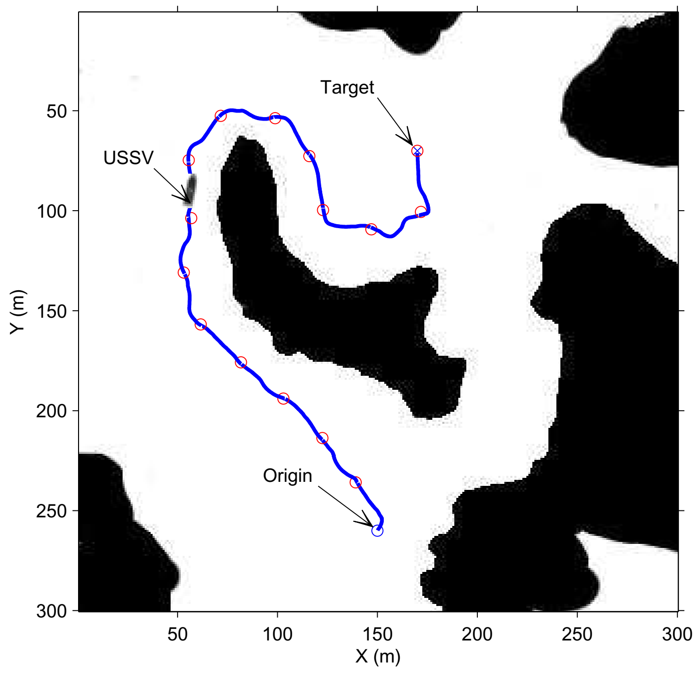
We used $\tau = 0.075$ and $d\tau = 0.1$ to obtain Figure 5.8. The time taken to compute the set of trajectories for each sea-state was about 8.8 minutes.

The MDP formulated in the Section 5.4.1 can be solved using the numerous algorithms including the value iteration, policy iteration, hierarchical techniques, approximate techniques, etc. [Pow07]. We chose the basic value iteration for computation of the solution. The details of the value iteration algorithm could be found in many references including [LaV06, TBF05]. The policies for each sea-state are computed using value iteration over the obtained respective state transition maps. The optimal policies are then used for determining the trajectory to the target. The obtained trajectory is optimal given the uncertainties due to the ocean waves and USSV interaction and the unmodeled effects such as the variations in the angular velocities and the linear velocity due to the implemented PID controller limitations. The dynamics based motion model developed in this chapter includes these variations in the transition probability. The transition probability is then used to compute the optimal policy in the MDP framework, which has been proved by the researchers to yield global optima. It should be noted that the optimality is achieved in the resolution sense meaning that the chosen resolution of the state-action space influences the achieved optimality. Finer the resolution better will be the optimal solution but greater will be the computational complexity.

The resulting trajectories for sea-states 3 and 4 are shown in Figure 5.10. In case of sea-state 3 a shorter and riskier trajectory (through a narrow passage in the midst of obstacles) is computed by the algorithm as shown in Figure 5.10(a), whereas in case of sea-state 4 a longer but safer path is computed as shown in Figure 5.10(b).



(a) *Computed at sea-state 3.*



(b) *Computed at sea-state 4.*

Figure 5.10: *Executed trajectories obtained by using feedback plan for sea-states 3 and 4. Feedback plans are computed using the algorithms developed in this chapter*

The disturbances due to the ocean waves are computed during the simulation which causes unpredictable deviations in the trajectory. The computed policy makes it possible for the USSV to take best action to safely reach the target even after it gets deviated due to the unpredictable ocean waves. The data structure and the algorithms presented in this chapter enables incorporating the effects of dynamics and uncertainty in ocean waves in the computation of state transition map, helping in performing physics aware trajectory planning.

The time taken to perform Monte Carlo simulations of the USSV dynamics to compute state transition map using the algorithm developed in this chapter was 9 minutes. The number of states were chosen as 4800 ($20 \times 20 \times 12$) and the number of actions as 7. The value iteration took 28 s to converge when computed on the computed state transition map.

5.5 Summary

In this chapter we extended the USSV dynamics model presented in Chapter 4 to incorporate ocean waves with multiple wave components and random phase lags. We developed GPU based algorithm to perform fast Monte-Carlo simulations to estimate state transition probability of USSV operating in high sea-states using dynamics simulations. We further improved the computational performance by developing model simplification algorithm based on temporal coherence. The overall computational speedup obtained is by a factor of 43.1 by introducing an error of 1.04% over the CPU baseline. We used the developed simulator to estimate state transition map and used it in traditional simulation based trajectory planning framework [LaV06].

The chapter presents a trajectory planning case study to demonstrate the physics-aware trajectory generated in an ocean environment with the sea-state 3 and 4, using the developed state transition data structure. The planning system is flexible and is capable of generating dynamically feasible trajectory plans for any given sea-state and USSV geometry.

The current framework is capable of handling the dynamic environmental disturbances due to the ocean waves with static obstacles such as islands and shorelines. The trajectory planner cannot handle dynamic obstacles optimally and a faster re-planning approach can be used to tackle with this issue.

Chapter 6
Conclusions

This chapter presents the intellectual contributions and anticipated benefits from the work reported in this dissertation.

6.1 Intellectual Contributions

This dissertation aims toward building theoretical framework and development of algorithms to perform context dependent physics preserving automatic model simplification. The developed algorithms can be applied to rigid body dynamics simulation which is widely used in VEs. The VEs are very useful tools for design, planning, and training. Hence, the developed algorithms can prove to be useful tools in design, planning, and training applications. Some of the key intellectual contributions are listed below.

6.1.1 Contact Preserving Model Simplification Technique for Rigid Body Dynamics Simulations

This dissertation introduces a context dependent contact preserving model simplification technique for interactive rigid body simulation. The technique is based on the accessibility based approach. An important requirement for model simplification algorithms (in rigid body simulation application) is that the contact points between unsimplified and simplified models must remain exactly identical. By preserving the contact points, the reaction forces occurring due to collision among parts are also preserved. This in turn will preserve the physics. The accessibility based simplification technique developed in this dissertation will determine and remove the regions in the part models, which cannot come in contact during rigid body motion. The inaccessible regions do not contribute to the generation of contact points and thus their off-line removal does not influence simulation accuracy. This however improves the computational performance of contact point determination algorithms as the size of the hierarchical tree representing the collision shape of the part gets pruned. Our technique thus complements the existing hierarchical data-structure based approaches for collision detection by pruning the *inaccessible* nodes of hierarchical trees before the simulation, thereby reducing worst case collision detection time which occurs when parts are in close proximity.

6.1.2 Clustering and Temporal Coherence Based Model Simplification for Rigid Body to Fluid Interaction Simulation

This dissertation introduces model simplification algorithms based on spatial and temporal coherence properties of fluid pressure acting on the floating rigid bodies to simplify the computations. We obtained real-time performance of the simulator using a simplification approach based on the clustering, temporal coherence, and parallelization approach to perform a physics-preserving model simplification for the potential flow based 6-DOF, time domain simulation of USSVs. The average computation time was reduced by a factor of about 28.50 introducing an average error in force magnitude of about 5.8% with respect to the baseline computations, using the simplification techniques described in this dissertation. The approach can take care of any arbitrary hull geometry as long as it can be expressed as a polygonal model. We established theoretically that the time taken for computing the forces increases linearly with an increasing cluster count and reduces linearly with an increasing tolerance, whereas the error introduced reduces with cluster count in an inverse square root fashion and increases with an increasing tolerance in a linear fashion. The numerical simulation results are in agreement to these theoretical results. Cluster count and tolerance can thus be used as high level approximation parameters in the simplification scheme to control computation time to achieve a desired level of accuracy.

6.1.3 GPU Based Parallelization and Temporal Coherence Based Simplification to Accelerate USSV Simulator Performance

Fluid to rigid body interaction computation is highly data parallel operation. This dissertation presents a GPU implementation of the simulator to exploit the data parallelism of the operations involved in simulation computation. In addition to that, temporal coherence based model simplification is incorporated into the GPU implementation framework to further improve the simulation performance. The combination of GPU parallelization and temporal coherence based model simplification led to an overall computational speedup improvements by a factor of 43.1 over CPU baseline (optimized with OpenMP based parallelization).

6.1.4 Monte-Carlo Simulation Based State Transition Map Generation for USSV Trajectory Planning Under Motion Uncertainty

Nondeterministic ocean wave forces cause deviations from the intended trajectory of USSVs leading to motion uncertainty. Estimating the motion uncertainty using experimental approach is most reliable, however very expensive and sometimes infeasible due to the presence of diverse operating conditions. Computer simulations are inexpensive and can be made accurate enough by incorporating experimental knowledge. During a long mission, USSV might be required to pass through many sea-states and undergo environmental changes such as currents, wakes of other vessels, mass changes due to fuel loss, etc. Enumerating all the cases beforehand to perform off-line simulations may be infeasible particularly due to continuous nature of the sea-state variables and their nonlinear interactions. This necessitates on-line simulation of USSVs to generate state transition map for a given sea condition. The information about the state transition map is very crucial for reliable trajectory planning of USSVs. In this dissertation we present a Monte-Carlo simulation based approach for estimating state transition map from dynamics simulations. The use of GPU has two advantages, namely, it provides immense computing power and it is lightweight unlike computing clusters.

6.2 Anticipated Benefits

This dissertation addresses the issue of physics preserving model simplification for attaining real-time refresh rates for rigid body simulation based on general model simplification principles, namely, clustering, temporal coherence, and parallelization. The developed techniques can be used in robot motion planning applications. Since the developed model simplification approaches are general enough, they can be used for many robotics applications which have interactions similar to rigid body to rigid body or rigid body to fluid flow. Turnaround time for generating simulation scenarios can be significantly reduced for virtual environment using the work done in this dissertation. The virtual environments can be useful for training purposes, which can isolate human beings from the dangerous situations at a very low cost. The faster refresh rates obtained by using model simplification strategies developed in this dissertation will be helpful to improve the immersivity of the virtual environments leading to their enhanced effectiveness. The faster virtual environments thus developed can be used in the area of defense (*e.g.*, simulation of USSVs), healthcare (surgical simulations, drug delivery simulations, etc.), entertainment (realistic games), etc. The USSV simulation frame work developed in this dissertation can be useful for many applications like physics-aware robot motion planning, USSV training using tele-operation, controller design, hull design, capsizing probability determination, etc.

6.3 Future Directions

The principles of physics-preserving model simplification namely clustering, temporal coherence, accessibility based pruning of hierarchical trees, and simulation context dependent model simplification developed in this dissertation are general and can be applied to wide variety of physics based simulations. Some of the possible future research directions are described below:

(i.) This dissertation focused on displacement physics based fluid-rigid body interaction model, which does not take into account the hydroplaning phenomenon. One of the possible research directions can be to extend and enhance the model simplification techniques for high speed USSVs undergoing hydroplaning phenomenon.

(ii.) This dissertation focused on simulation of USSVs as a rigid body. There have been a surge in research and development of bio-inspired robots that operate in water such as amphibots in recent years [CBGI05, CI08, MMG10, SCS10]. A possible research direction can be to perform simulation of amphibots in real-time. Model simplification techniques can be applied to such a simulator for faster simulation, which may prove to be a useful tool for designing and testing the geometry and mechanism of such robots. Gait synthesis is another major research thrust in bio-inspired robotics. Often machine learning techniques are used for performing gait synthesis [GJ09, SML10]. Most machine learning techniques require a high fidelity dynamics simulation for solving gait synthesis problem. Since, the simulation needs to be run many times by the machine learning algorithms for synthesis, the simulations need to be computationally fast [Mit97, SB98]. Model simplification techniques can hence be used in solving robot gait synthesis problems.

(iii.) Lately, robot motion planning research focus is shifting towards physics-aware model predictive robot motion planning as opposed to motion planning purely based on geometric or kinematic constraints of robots. A dynamics based simulation can provide much more realistic constraints as compared to the purely geometric constraints. Dynamics constraints become crucial in case of fluid rigid body interaction problems, robots moving at very high speeds, robots moving on treacherous terrains, etc. One major roadblock in using dynamics simulation in robot motion planning is the computational expense associated with the dynamics simulations. Model simplification techniques can be suitably applied to the robotics problems to incorporate physical constraints into motion planning algorithms to generate physics-aware robot motion plans.

Bibliography

[AAP11] M. Alexander, J. Allison, and P. Papalambros. Reduced representations of vector-valued coupling variables in decomposition-based design optimization. *Structural and Multidisciplinary Optimization*, pages 1–13, 2011. 10.1007/s00158-011-0636-9.

[ABA02] C. Andujar, P. Brunet, and D. Ayala. Topology-reducing surface simplification using a discrete solid representation. *ACM Trans. Graph.*, 21(2):88–105, 2002.

[ABE09] D. Attali, J. D. Boissonnat, and H. Edelsbrunner. Stability and computation of the medial axis: a state-of-the-art report. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, pages 109–125, 2009.

[ACVB09] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009.

[AMM10] H. Ashrafiuon, K. R. Muske, and L. C. McNinch. Review of nonlinear tracking and setpoint control approaches for autonomous underactuated marine vehicles. In *American Control Conference (ACC), 2010*, pages 5203 –5211, 302010-july2 2010.

[Arm94] C. G. Armstrong. Modelling requirements for finite-element analysis. *Computer-Aided Design*, 26(7):573–578, 1994.

[Bar94] R. R. Barton. Metamodeling: a state of the art review. In *Proceedings of the 26th conference on Winter simulation*, WSC '94, pages 237–244, San Diego, CA, USA, 1994. Society for Computer Simulation International.

[bat10] Battelle explosive ordnance disposal virtual environment., February 2010. http://www.battelle.org/.

[BB08] P. J. Bandyk and R. F. Beck. Nonlinear ship motions in the time-domain using a body-exact strip theory. *ASME Conference Proceedings*, 2008(48234):51–60, 2008.

[BBB07] C. Batty, F. Bertails, and R. Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26(3):100, 2007.

[BCN06] M. R. Benjamin, J. A. Curcio, and P. M. Newman. Navigation of unmanned marine vehicles in accordance with the rulesof the road. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3581 –3587, May 2006.

[BdB98] R. Bidarra, R. K. deKraker, and W. Bronsvoort. Representation and management of feature information in a cellular model. *Computer-Aided Design*, 30(4):301–313, 1998.

[Ber97] G. V. Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[Bla74] R. W. Blanning. The sources and uses of sensitivity information. *Interfaces*, 4(4):pp. 32–38, 1974.

[BLOY01] M. S. Branicky, S. M. LaValle, K. Olson, and Y. Yang. Quasi-randomized path planning. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1481 – 1487, 2001.

[Blu67] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Forms*, pages 362–380. MIT Press, Amsterdam, 1967.

[BM06] R. R. Barton and M. Meckesheimer. Metamodel-based simulation optimization. In S. G. Henderson and B. L. Nelson, editors, *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, pages 535 – 574. Elsevier, 2006.

[BMNB05] R. Bidarra, J. Madeira, W. J. Neels, and W. F. Bronsvoort. Efficiency of boundary evaluation for a cellular model. *Computer-Aided Design*, 37(12):1266–1284., 2005.

[BR01] R. Beck and A. Reed. Modern seakeeping computations for ships. In *Twenty-Third Symposium on Naval Hydrodynamics*. Naval Studies Board (NSB), 2001.

[BSRB06] F. Borrelli, D. Subramanian, A.U. Raghunathan, and L.T. Biegler. MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles. In *American Control Conference, 2006*, page 6 pp., june 2006.

[BT93] R. Bidarra and J. C. Teixeira. Intelligent form feature interaction management in a cellular modeling scheme. In *SMA '93: Proceedings on the second ACM symposium on Solid modeling and applications*, pages 483–485, New York, NY, USA, 1993. ACM.

[BTT09] M. Becker, H. Tessendorf, and M. Teschner. Direct forcing for lagrangian rigid-fluid coupling. *Visualization and Computer Graphics, IEEE Transactions on*, 15(3):493 –503, may-june 2009.

[CAD05] Iti transcendata, interoperability solutions for cad/cam/cae., August 2005. http://www.cadfix.com.

[CBGI05] A. Crespi, A. Badertscher, A. Guignard, and A. J. Ijspeert. Amphibot i: an amphibious snake-like robot. *Robotics and Autonomous Systems*, 50(4):163 – 175, 2005. Biomimetic Robotics.

[CCM97] P. R. Cavaleanti, P. C. P. Carvalho, and L. F. Martha. Non-manifold modeling: an approach based on spatial subdivision. *Computer-Aided Design.*, 29(3):209–220, 1997.

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

[CGP93] J. Chen, E. L. Grsz, and F. B. Prinz. Integration of parametric geometry and non-manifold topology in geometric modeling. In *Proceedings on the second ACM symposium on Solid modeling and applications. New York. NY.*, 1993.

[CI08] A. Crespi and A.J. Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *Robotics, IEEE Transactions on*, 24(1):75 –87, feb. 2008.

[CJP08] B. Chapman, G. Jost, and A. R. V. D. Pas. *Using OpenMP portable shared memory parallel programming.* Cambridge, Massachusetts, USA: The MIT Press, 2008.

[CKL02] 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.

[CKL04] C. S. Chong, A. S. Kumar, and K. H. Lee. Automatic solid decomposition and reduction for non-manifold geometric model generation. *Computer-Aided Design*, 36(13):1357–1369., 2004.

[CMBG07] J. Craighead, R. Murphy, J. Burke, and B. Goldiez. A survey of commercial open source unmanned vehicle simulators. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 852 –857, 2007.

[CMS98] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[CMT04] M. Carlson, P. J. Mucha, and G. Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph.*, 23(3):377–384, 2004.

[CTS09] G. Casalino, A. Turetta, and E. Simetti. A three-layered architecture for real time path planning and obstacle avoidance for surveillance usvs operating in harbour fields. pages 1 –8, May 2009.

[CVR09] C. Colombo, M. Vasile, and G. Radice. Optimal low-thrust trajectories to asteroids through an algorithm based on differential dynamic programming. *Celestial Mechanics and Dynamical Astronomy*, 105:75–112, 2009. 10.1007/s10569-009-9224-3.

[CWNS07] P. M. Carrica, R. V. Wilson, R. W. Noack, and F. Stern. Ship motions using single-phase level set with dynamic overset grids. *Computers & Fluids*, 36(9):1415 – 1433, 2007.

[DAP00] J. Donaghy, C. G. Armstrong, and M. A. Price. Dimensional reduction of surface models for analysis. *Engineering with Computers.*, 16(1):24–35, 2000.

[Dig11] The digital Michelangelo project., July 2011. http://graphics.stanford.edu/projects/mich/.

[DJP02] K. D. Do, Z. P. Jiang, and J. Pan. Underactuated ship global tracking under relaxed conditions. *IEEE Transactions on Automatic Control*, 47(9):1529–1536, Sep 2002.

[DKK$^+$05] H. Date, S. Kanai, T. Kisinami, I. Nishigaki, and T. Dohi. High-quality and property controlled finite element mesh generationfrom triangular meshes using the multiresolution technique. *Journal of Computing and Information Science in Engineering*, 5(4):266–276, 2005.

[DKKN06] H. Date, S. Kanai, T. Kisinami, and I. Nishigaki. Flexible feature and resolution control of triangular meshes. In *Proceedings of the Sixth IASTED International Conference on Visualization, Imaging and Image Processing*, Palma de Mallorc. Spain., 2006.

[DMB$^+$96] 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.

[DPS94] P. Dabke, V. Prabhakar, and S. Sheppard. Using features to support finite element idealizations. In *Proceedings of the ASME Conference on Computers in Engineering*, Minneapolis. MN. USA., 1994.

[DSG97] S. Dey, M. S. Shepard., and M. K. Georges. Elimination of the adverse effects of small model features by then local modification of automatically generated meshes. *Engineering with Computers.*, 13(3):134–151, 1997.

[DXCR93] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *J. ACM*, 40:1048–1066, November 1993.

[ED05] M. G. Earl and R. D'Andrea. Iterative milp methods for vehicle-control problems. *Robotics, IEEE Transactions on*, 21(6):1158 – 1167, dec. 2005.

[Eri04] C. Ericson. *Real-time collision detection.* The Morgan Kaufmann Series in Interactive 3D Technology., 2004.

[ESV98] J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):133–144, 1998.

[FCF$^+$08] 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.

[FDF99] E. Frazzoli, M.A. Dahleh, and E. Feron. A hybrid control architecture for aggressive maneuvering of autonomous helicopters. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 3, pages 2471 –2476 vol.3, 1999.

[FDF01] E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *American Control Conference, 2001. Proceedings of the 2001*, volume 1, pages 43 –49, 2001.

[FLM03] M. Foskey, M. C. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 96–107, New York, NY, USA, 2003. ACM.

[Fos94] T. I. Fossen. *Guidance and control of ocean vehicles.* Wiley, Chicester, England, 1994.

[FRL00] L. Fine, L. Remondini, and J. C. Leon. Automated generation of FEA models through idealization operators. *International Journal for Numerical Methods in Engineering*, 49(1-2):83–108, 2000.

[FS04a] D. Ferguson and A. Stentz. Focussed processing of MDPs for path planning. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)*, pages 310–317, 2004.

[FS04b] T. I. Fossen and Ø. N. Smogeli. Nonlinear time-domain strip theory formulation for low-speed manoeuvering and station-keeping. *Modeling, Identification and Control*, 25(4):201–221, 2004.

[GAB+08] S. K. Gupta, D. K. Anand, J. E. Brough, M. Schwartz, and R. A. Kavetsky. *Training in virtual environments: A safe, cost-effective, and engaging approach to training.* CALCE EPSC Press, College Park,, July 2008.

[GAT+10] S. K. Gupta, D. K. Anand, A. Thakur, P. Svec, and M. Schwartz. A simulation based framework for discovering planning logic for unmanned surface vehicles. In *ASME Engineering Systems Design and Analysis Conference, July 12-14 2010, Istanbul, Turkey.*, 2010.

[GBT93] A. Gomes, R. Bidarra, and J. C. Teixeira. A cellular approach for feature-based modeling. In *In Graphics Model@ and Visualization in Science and Technology.*, 1993.

[GCTW10] R. Geist, C. Corsi, J. Tessendorf, and J. Westall. Lattice-boltzmann water waves. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Ronald Chung, Riad Hammoud, Muhammad Hussain, Tan Kar-Han, Roger Crawfis, Daniel Thalmann, David Kao, and Lisa Avila, editors, *Advances in visual Computing*, volume 6453 of *Lecture Notes in Computer Science*, pages 74–85. Springer Berlin / Heidelberg, 2010.

[GGR11] M. Garcia, J. Gutierrez, and N. Rueda. Fluid-structure coupling using lattice-boltzmann and fixed-grid FEM. *Finite elements in analysis and design*, 47(8):906 – 912, 2011. Computational Mechanics and Design.

[GH97] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[GJ09] A. German and M. Jenkin. Gait synthesis for legged underwater vehicles. In *Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on*, pages 189 –194, april 2009.

[GKM10] C. Goerzen, Z. Kong, and B. Mettler. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent &amp; Robotic Systems*, 57:65–100, 2010. 10.1007/s10846-009-9383-1.

[GLM96] 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.

[GNU$^+$09] O. Grottke, A. Ntouba, S. Ullrich, W. Liao, E. Fried, A. Prescher, T. M. Deserno, T. Kuhlen, and R. Rossaint. Virtual reality-based simulator for training in regional anaesthesia. *British Journal of Anaesthesia*, 103(4):594–600, 2009.

[Gor02] J. J. Gorski. Present state of numerical ship hydrodynamics and validation experiments. *Journal of Offshore Mechanics and Arctic Engineering*, 124(2):74–80, 2002.

[GRGT11] M. Geveler, D. Ribbrock, D. Goddeke, and S. Turek. Lattice-boltzmann simulation of the shallow-water equations with fluid-structure interaction on multi- and manycore processors. In Rainer Keller, David Kramer, and Jan-Philipp Weiss, editors, *Facing the*

Multicore-Challenge, volume 6310 of *Lecture Notes in Computer Science*, pages 92–104. Springer Berlin / Heidelberg, 2011.

[H.05] Lee S. H. Feature-based multiresolution modeling of solids. *ACM Transactions on Graphics.*, 24:1417–1441, 2005.

[HD06] D. S. Holloway and M. R. Davis. Ship motion computations using a high Froude number time domain strip theory. *Journal of Ship Research*, 50(1):15–30, 2006.

[HHK+95] 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.

[HK05] 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.

[HK07] T. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *International Journal of Robotics Research*, 26(1):141–166, February 2007.

[HKLR02] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[HM07] V. Hayward and K. E. Maclean. Do it yourself haptics: part I tutorial. *Robotics Automation Magazine, IEEE*, 14(4):88 –104, dec. 2007.

[Hof89] C. M. Hoffmann. *Geometric and solid modeling: an introduction.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[Hub96] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[IIYF01] K. Inouea, T. Itoha, A. Yamadaa, and T. andShimadac K. Furuhatab. Face clustering of a large-scale cad model for surface mesh generation. *Computer-Aided Design.*, 33(3):251–261., 2001.

[JD03] N. Joshi and D. Dutta. Feature simplification techniques for freeform surface models. *Journal of Computing and Information Science in Engineering*, 3(3):177–186, 2003.

[JG97] D. Jung and K. Gupta. Octree-based hierarchical distance maps for collision detection. *Journal of Robotic Systems*, 14:789–806, 1997.

[JTT01] P. Jimenez, F. Thomas, and C. Torras. 3d collision detection: a survey. *Computers & Graphics*, 25(2):269 – 285, 2001.

[KGL+98] 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.

[KGM+03] K. H. Kim, J. Gorski, R. Miller, R. Wilson, F. Stern, M. Hyman, and C. Burg. Simulation of surface ship dynamics. In *User Group Conference, 2003. Proceedings*, pages 188–199, June 2003.

[KGZ97] M. R. Katebi, M. J. Grimble, and Y. Zhang. H ∞ robust control design for dynamic ship positioning. *Control Theory and Applications, IEEE Proceedings*, 144(2):110–120, Mar 1997.

[KHI+07] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe. Collision detection: A survey. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 4046 –4051, oct. 2007.

[KHM+98] 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.

[Kim02] K. H. Kim. Simulation of surface ship dynamics using unsteady RANS codes. In *Reduction of Military Vehicle Acquisition Time and Cost through Advanced Modelling and Virtual Simulation, Paris, France*, 2002.

[KKF05] P. Krishnamurthy, F. Khorrami, and S. Fujikawa. A modeling framework for six degree-of-freedom control of unmanned sea surface vehicles. In *Proc. and 2005 European Control Conference Decision and*

Control CDC-ECC '05. 44th IEEE Conference on, pages 2676–2681, December 12–15, 2005.

[Kle75] J. P. C. Kleijnen. A comment on blanning's "metamodel for sensitivity analysis: The regression metamodel in simulation. *Interfaces*, 5(3):pp. 21–23, 1975.

[Kle08] J. P. C. Kleijnen. Kriging metamodels. In Frederick S. Hillier, editor, *Design and Analysis of Simulation Experiments*, volume 111 of *International Series in Operations Research and Management Science*, pages 139–156. Springer US, 2008.

[KLH+05] S. Kim, K. Lee, T. Hong, M. Kim, and M. andSong Y. Jung. An integrated approach to realize multi-resolution of B-Rep model. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling. Cambridge. MA.*, 2005.

[KMGL99] S. Kumar, D. Manocha, W. Garrett, and M. Lin. Hierarchical back-face computation. *Computers & Graphics*, 23:681–692, 1999.

[KSA+06] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and Ra. Warner. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483, 2006.

[LAFMD06] G. Li, S. Azarm, A. Farhang-Mehr, and A. Diaz. Approximation of multiresponse deterministic engineering simulations: a dependent metamodeling approach. *Structural and Multidisciplinary Optimization*, 31:260–269, 2006. 10.1007/s00158-005-0574-5.

[LAPL05] K. Y. Lee, C. G. Armstrong, M. A. Price, and J. H. Lamont. A small feature suppression/unsuppression system for preparing B-Rep models for analysis. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 113–124, 2005.

[LaV06] S. M. LaValle. *Planning algorithms.* Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[Lee99] K. Lee. *Principles of CAD/CAM/CAE Systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[Lee05] S. H. Lee. A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modeling techniques. *Computer-Aided Design*, 37:941–955, 2005.

[LFP00] A. Loria, T. I. Fossen, and E. Panteley. A separation principle for dynamic positioning of ships: Theoretical and experimental results. *IEEE Transactions on Control Systems Technology*, 8:332–343, 2000.

[LG98] M. C. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *In Proc. of IMA Conference on Mathematics of Surfaces*, pages 37–56, 1998.

[LGT04] M. Likhachev, G. Gordon, and S. Thrun. Planning for Markov decision processes with sparse stochasticity. *Advances in Neural Information Processing Systems (NIPS)*, 17, 2004.

[LK01a] S. M. Lavalle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *The International Journal of Robotics Research*, 20(9):729–752, 2001.

[LK01b] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[LL98] Y. G. Lee and K. Lee. Geometric detail suppression by the fourier transform. *Computer-Aided Design*, 30(9):677–693, 1998.

[LL01] S. H. Lee and K. Lee. Partial entity structure: A compact boundary representation for non-manifold geometric modeling. *Journal of Computing and Information Science in Engineering.*, 1(4):356–365., 2001.

[LL02] S. Lee and K. Lee. Wrap-around operation to make multi-resolution model of part and assembly. *Computers & Graphics*, 26(5):687–700, 2002.

[LL05] S. H. Lee and K. Y. Lee. Feature-based multiresolution and multi-abstraction non-manifold modeling system to provide integrated environment for design and analysis of injection molding products. In *Proceedings of the First Korea-China Joint Conference on Geometric and Visual Computing*, Busan. Korea., 2005.

[LL06a] S. H. Lee and K. Lee. Feature-based multiresolution techniques for product design. *Journal of Zhejiang University SCIENCE A.*, 7:1535–1543, 2006.

[LL06b] S. R. Lindemann and S. M. LaValle. Multiresolution approach for motion planning under differential constraints. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 139 –144, may 2006.

[LLA08] M. Li, G. Li, and S. Azarm. A kriging metamodel assisted multi-objective genetic algorithm for design optimization. *Journal of Mechanical Design*, 130(3):031401, 2008.

[LLK06] S. H. Lee, K. Lee, and S. C. Kim. History-based selective boolean operations for feature-based multi-resolution modeling. In *Proceedings of ICCSA 2006. Glasgow. United Kingdom.*, 2006.

[LLKK04] 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.

[LLP02] S. H. Lee, K. S. Lee, and S. Park. Multiresolution representation of solid models using the selective Boolean operations. In *Proceedings of the 2002 Spring Conference of Korean Society of Precision Engineering*, 2002.

[LMC$^+$04] T. Lim, H. Medellin, J. R. Corney, J. M. Ritchie, and J. B. C. Davies. Decomposition of complex models for manufacturing. In *Proceedings of the Shape Modeling and Applications. Cambridge. MA.*, 2004.

[LPA03] K. Y. Lee, M. A. Price, and C. G. Armstrong. CAD-TO-CAE integration through automated model simplification andadaptive modeling. In *Proceedings of International Conference on Adaptive Modeling and Simulation*, Barcelona. Spain., 2003.

[LPN03] E. Lefeber, K.Y. Pettersen, and H. Nijmeijer. Tracking control of an underactuated ship. *Control Systems Technology, IEEE Transactions on*, 11(1):52–61, Jan 2003.

[Lue01] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics Applications*, 21(3):24–35, 2001.

[LWCR02] D. P. Luebke, B. Watson, J. D. Cohen, and M. andVarshney A. Reddy. *Level of Detail for 3D Graphics.* Elsevier Science Inc., New York, NY, USA, 2002.

[Mas93] H. Masuda. Topological operators and boolean operations for complex-based non-manifold geometric models. *Computer-Aided Design*, 25(2):119–129, 1993.

[MCC98] A. V. Mobley, M. P. Carroll, and S. A. Canann. An object oriented approach to geometry defeaturing for finite element meshing. In *Proceedings of 7th International Meshing Roundtable. Dearborn. MI.*, 1998.

[MH08] K. E. MacLean and V. Hayward. Do it yourself haptics: Part II tutorial. *Robotics Automation Magazine, IEEE*, 15(1):104 –119, Mar. 2008.

[Mit97] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[MMG10] A. Maity, S. Majumder, and S. Ghosh. An experimental hyper redundant serpentine robot. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 3180 –3185, oct. 2010.

[MPN02] F. Mazenc, K. Pettersen, and H. Nijmeijer. Global uniform asymptotic stabilization of an underactuated surface vessel. *Automatic Control, IEEE Transactions on*, 47(10):1759–1762, Oct 2002.

[New77] J. N. Newman. *Marine Hydrodynamics.* MIT Press, Cambridge, MA, 1977.

[ode08] Opendynamics rigid body dynamics engine., November 2008. http://www.ode.org/.

[PG95] I. Palmer and R. Grimsdale. Collision detection for animation using sphere trees. *Computer Graphics Forum*, 14(2):105–116, 1995.

[PKK09] M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[PKV10] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Motion planning with dynamics by a synergistic combination of layersof planning. *Robotics, IEEE Transactions on*, 26(3):469 –482, Jun. 2010.

[Pow07] W. Powell. *Approximate dynamic programming: Solving the curses of dimensionality.* Wiley, 2007.

[PPB10] C. Passenberg, A. Peer, and A. Buss. A survey of environment-, operator-, and task-adapted controllers for teleoperation systems. *Mechatronics*, 20(7):787 – 801, 2010. Special Issue on Design and Control Methodologies in Telerobotics.

[Put94] M. L. Puterman. *Markov decision processes: Discrete stochastic dynamic Programming.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[RAM$^+$06] T. T. Robinson, C. G. Armstrong, G. McSparron, A. Quenardel, H. Ou, and R. M. McKeag. Automated mixed dimensional modelling for the finite element analysis of swept and revolved CAD features. In *SPM '06: Proceedings of the 2006 ACM symposium on Solid and physical modeling*, pages 117–128, 2006.

[RC10] W. Ren and H. Chen. Finite element model updating in structural dynamics by using the response surface method. *Engineering Structures*, 32(8):2455 – 2465, 2010.

[Rez96] M. Rezayat. Midsurface abstraction from 3D solid models: general theory and applications. *Computer-Aided Design*, 28:905–915, 1996.

[RG01] J.N. Reddy and D.K. Gartling. *The finite element method in heat transfer and fluid dynamics (2nd ed.)*. CRC Press, Boca Raton (2001), 2001.

[RG05] M. Ramanathan and B. Gurumoorthy. Constructing medial axis transform of extruded and revolved 3D objects with free-form boundaries. *Computer-Aided Design*, 37:1370–1387, 2005.

[RH02] A. Richards and J.P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1936 – 1941 vol.3, 2002.

[RHG$^+$01] K. Ribelles, P. S. Heckbert, M. Garland, T. Stahovich, and V. Shivastava. Finding and removing features from polyhedra. In *Proceedings of ASME DETC01*, Pittsburgh. PA. USA., 2001.

[RKC02] S. Redon, A. Kheddar, and S. Coquillart. Hierarchical back-face culling for collision detection. *Intelligent Robots and System.*, 3(3):3036–3041, 2002.

[RN09] S. Russell and P. Norvig. *Artificial intelligence: A modern approach.* Prentice Hall, 2009.

[RS98] S. Raghothama and V. Shapiro. Boundary representation deformation in parametric solid modeling. *ACM Transactions on Computer Graphics.*, 17(4):259–286., 1998.

[Saa03] Y. Saad. *Iterative methods for sparse linear systems.* Society for Industrial and Applied Mathematics. Philadelphia. PA., 2003.

[SAM09] R. A. Soltan, H. Ashrafiuon, and K. R. Muske. State-dependent trajectory planning and tracking control of unmannedsurface vessels. In *American Control Conference, 2009. ACC '09.*, pages 3597 –3602, 2009.

[SB98] R. Sutton and A. Barto. *Reinforcement Learning: An introduction.* MIT Press, 1998.

[SBB97] 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.

[SBO98] M. S. Shephard, M. W. Beall, and R. M. O'bara. Revisiting the elimination of the adverse effects of small model features in automatically generated meshes. In *Proceedings of 7th International Meshing Roundtable '98, SAND 98-2250, Sandia*, 1998.

[SCS10] Keehong Seo, Soon-Jo Chung, and Jean-Jacques Slotine. Cpg-based control of a turtle-like underwater vehicle. *Autonomous Robots*, 28:247–269, 2010. 10.1007/s10514-009-9169-0.

[SFM05] A. Sud, M. Foskey, and D. Manocha. Homotopy-preserving medial axis simplification. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 39–50, 2005.

[SGDS09] S. Sanner, R. Goetschalckx, K. Driessens, and G. Shani. Bayesian real-time dynamic programming. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, pages 1784–1789. Morgan Kaufmann Publishers Inc., 2009.

[She01] A. Sheffer. Model simplification for meshing using face clustering. *Computer-Aided Design*, 33:925–934, 2001.

[SK08] B. Siciliano and O. Khatib, editors. *Handbook of Robotics.* Handbook of Robotics, Springer, New York, NY, USA, 2008.

[SM95] 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.

[SMFH03] T. Schouwenaars, B. Mettler, E. Feron, and J. P. How. Robust motion planning using a maneuver automation with built-in uncertainties. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, pages 2211 – 2216 vol.3, june 2003.

[SML10] Yi Sun, Shugen Ma, and Xin Luo. Design of an eccentric paddle locomotion mechanism for amphibious robots. In *Robotics and Biomimetics*

(ROBIO), 2010 IEEE International Conference on, pages 1098 –1103, dec. 2010.

[SPKA01] T.W. Simpson, J.D. Poplinski, P. N. Koch, and J.K. Allen. Metamodels for computer-based engineering design: survey and recommendations. *Engineering with Computers*, 17:129–150, 2001. 10.1007/PL00007198.

[SS07] S. P. Singh and D. Sen. A comparative linear and nonlinear ship motion study using 3-D time domain methods. *Ocean Engineering*, 34(13):1863 – 1881, 2007.

[SSCS07] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli. Flying fast and low among obstacles. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2023 –2029, april 2007.

[SSK+05] J. Seo, Y. Song, S. Kim, K. Lee, Y. Choi, and S. Chae. Wrap-around operation for multi-resolution CAD model. *Computer-Aided Design & Applications*, 2:67–76, 2005.

[SSTG09] 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 Workhop*, September, 2009.

[SSTG11] P. Svec, M. Schwartz, A. Thakur, and S. K. Gupta. Trajectory planning with look-ahead for unmanned sea surface vehicles to handle environmental disturbances. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, September 2011.

[Sta] Stanford 3D scanning repository (2007). http://graphics.stanford.edu/data/3Dscanrep.

[Suz05] S. Suzuki. Online four-dimensional flight trajectory search and its flight testing. *AIAA GNC Conference and Exhibit, 2005*, 2005.

[SW10a] S. Shan and G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41:219–241, 2010.

[SW10b] Songqing Shan and G. Gary Wang. Metamodeling for high dimensional simulation-based design problems. *Journal of Mechanical Design*, 132(5):051009, 2010.

[Tau01] T. J. Tautges. Automatic detail reduction for mesh generation applications. In *Proceedings 10th International Meshing Roundtable*, pages 407–418, 2001.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2005.

[TBG09] 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.

[TC09] Cameron J. Turner and Richard H. Crawford. N-dimensional nonuniform rational b-splines for metamodeling. *Journal of Computing and Information Science in Engineering*, 9(3):031002, 2009.

[TCL99] 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.

[TG09] A. Thakur and S. K. Gupta. Context dependent contact preserving off-line model simplification for interactive rigid body dynamics simulations. In *ASME 2009 International Design Engineering Technical Conferences (IDETC) & Computers and Information in Engineering Conference (CIE) August 30 - September 2, 2009, San Diego, CA.*, 2009.

[TG10] A. Thakur and S. K. Gupta. A computational framework for real-time unmanned sea surface vehicle motion simulation. In *ASME 2010 International Design Engineering Technical Conferences (IDETC) & Computers and Information in Engineering Conference (CIE) August 15 -18, 2010, Montreal, Canada.*, 2010.

[TG11] A. Thakur and S. K. Gupta. Generation of state transition models using simulations for unmanned sea surface vehicle trajectory planning. In *ASME 2011 International Design Engineering Technical Confer-*

ences(IDETC) & Computers and Information in Engineering Conference (CIE) August 28 - 31, 2011, Washington DC, USA., 2011.

[Van94] G. Vanecek. Back-face culling applied to collision detection of polyhedra. *Journal of Visualization and Computer Animation*, 5(1):55–63, 1994.

[VL98] P. Veron and J. C. Leon. Shape preserving polyhedral simplification with bounded error. *Computers & Graphics.*, 22:565–585, 1998.

[vor10] Vortex simulation engine., February 2010. http://www.vxsim.com/en/simulators/.

[VS02] S. Venkataraman and M. Sohoni. Reconstruction of feature volumes and feature suppression. In *SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 60–71, 2002.

[VSR02] S. Venkataraman, M. Sohoni, and R. Rajadhyaksha. Removal of blends from boundary representation models. In *SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 83–94, 2002.

[WBK+10] M.T. Wolf, L. Blackmore, Y. Kuwata, N. Fathpour, A. Elfes, and C. Newman. Probabilistic motion planning of balloons in strong, uncertain wind fields. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1123 –1129, May 2010.

[WS07] G. G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[WSO03] D. R. White, S. Saigal, and S. J. Owen. Meshing complexity of single part CAD models. In *Proceedings of the 12th International Meshing Roundtable Conference. Santa Fe. NM.*, 2003.

[WWS05] G. D. Weymouth, R. V. Wilson, and F. Stern. RANS computational fluid dynamics predictions of pitch and heave ship motions in head seas. *Journal of Ship Research*, 49(18):80–97, June 2005.

[XKS09] B. Xu, A. Kurdila, and D. J. Stilwell. A hybrid receding horizon control method for path planning in uncertain environments. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4887 –4892, 2009.

[Yan10] F. Yang. Neural network metamodeling for cycle time-throughput profiles in manufacturing. *European Journal of Operational Research*, 205(1):172 – 185, 2010.

[YHL$^+$05] Y. Q. Yu, L. L. Howell, C. Lusk, Y. Yue, and He M. Gen. Dynamic modeling of compliant mechanisms based on the pseudo-rigid-body model. *Journal of Mechanical Design*, 127(4):760–765, 2005.

[YJJ$^+$10] L. Younghee, K. Jinkyung, K. Junghwan, J. K. Eun, G. K. Young, and I. Moon. Development of a web-based 3D virtual reality program for hydrogen station. *International Journal of Hydrogen Energy*, 35(5):2112 – 2118, 2010.

[YSLM04] S. Yoon, B. Salomon, M. Lin, and D. Manocha. Fast collision detection between massive models using dynamic model simplification. In *Proceedings of Eurographics Symposium on Geometry Processing. Nice. France.*, 2004.

[ZM02] H. Zhu and C. H. Menq. B-rep model simplification by automatic fillet/round suppressing for efficient automatic feature recognition. *Computer-Aided Design.*, 34(2):109–123., 2002.

[ZX10] D. Zhao and D. Xue. A comparative study of metamodeling methods considering sample quality merits. *Structural and Multidisciplinary Optimization*, 42:923–938, 2010. 10.1007/s00158-010-0529-3.