GPU Based Generation of State Transition Models Using Simulations for Unmanned Surface Vehicle Trajectory Planning

Atul Thakur[a], Petr Svec[a], and Satyandra K. Gupta[1b]

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

Abstract

This paper describes GPU based algorithms to compute state transition model for unmanned surface vehicles (USVs) using 6 degree of freedom (DOF) dynamics simulations of vehicle-wave interaction. State transition model is a key component of Markov Decision Process (MDP), which is a natural framework to formulate the problem of trajectory planning under motion uncertainty. USV trajectory planning problem is characterized by the presence of large and somewhat stochastic forces due to ocean waves, which can cause significant deviations in their motion. Feedback controllers are often employed to reject disturbances and get back on the desired trajectory. However, the motion uncertainty can be significant and must be considered in the trajectory planning to avoid collisions with the surrounding obstacles. In case of USV missions, state transition probabilities need to be generated on-board, to compute trajectory plans that can handle dynamically changing USV parameters and environment (*e.g.*, changing boat inertia tensor due to fuel consumption, variations in damping due to changes in water density, variations in sea-state, etc.). The 6 DOF dynamics simulations reported in this paper are based on potential flow theory. We also present a model simplification algorithm based on temporal coherence and its GPU implementation to accelerate simulation computation performance. Using the techniques discussed in this paper we were able to compute state transition probabilities in less than 10 minutes. Computed transition probabilities are subsequently used in a stochastic dynamic programming based approach to solve the MDP to obtain trajectory plan. Using this approach, we are able to generate dynamically feasible trajectories for USVs that exhibit safe behaviors in high sea-states in the vicinity of static obstacles.

Keywords:
USV, vehicle simulation, fluid-rigid interaction, GP-GPU, trajectory planning, MDP, state transition map, stochastic dynamic programming, motion uncertainty

1. Introduction

USVs operate in ocean environment with disturbances caused by waves, currents, wake of other ships, etc. The disturbances impart significant uncertainty in vehicle's motion. Due to the presence of high motion uncertainty, an action of a USV may not lead to the exact desired motion despite of using a feedback controller. Vehicle dynamics and motion uncertainty together make the task of trajectory planning very challenging, particularly in highly cluttered environments. Consider Figure 1, which shows the influence of vehicle dynamics and motion uncertainty on the planned trajectories in a USV mission. Circles M, P, and Q denote three consecutive waypoints of a mission. When the USV reaches P, it needs to find the optimum trajectory (with minimum length and risk of collision) between P and Q. One way, to solve this problem, would be to find the optimum trajectory, without explicitly considering the ocean disturbances (shown as trajectory A). Disturbances may be insignificant in the case when the sea-state is calm (*e.g.*, sea-states 1 to 3) or the USV is heavy enough to get deviated. If sea-state is very rough (*e.g.*, sea-state 4 and higher) or the USV is lighter, then the ocean disturbances may not allow the vehicle to closely follow the intended trajectory (A) as the disturbances may lead to collisions with an obstacle in a narrow region. A trajectory, which is safer with respect to the ocean disturbances but longer in length is shown in Figure 1 as trajectory B. It is thus evident that the physics of interaction between USV and ocean waves greatly influences

[1]Corresponding author

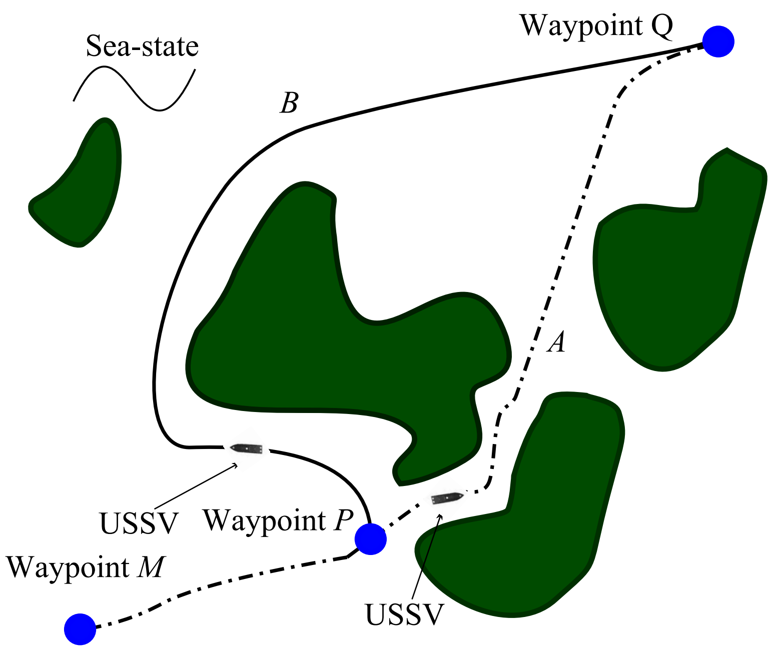


Figure 1: Trajectory plans for different ocean wave disturbance conditions. Trajectory *A* is shorter but riskier and may lead to collisions in the event of high sea-state whereas *B* is longer but more conservative to minimize the risk of collision. If sea-state is calm (*e.g.*, sea-states 1 to 3) or USV is heavy enough so that the disturbances are insignificant then trajectory *A* should be chosen. If sea-state is rough (*e.g.*, sea-state 4 and higher) or USV is light then trajectory *B* should be chosen.

the choice of trajectory plan. The variations take place in dynamical parameters of USV-ocean interaction such as inertia tensor of USV due to fuel loss, damping parameters due to change in water density, etc. during mission. In addition to that sea-state changes during the execution of mission due to change in weather. Based upon dynamics based interaction between USV and ocean waves and changing sea-state, a suitable trajectory, which is safe and still not overly conservative among *A* and *B*, needs to be planned.

Above outlined physics-aware trajectory planning problem in highly uncertain ocean environments can be solved by combining MDP [1, 2, 3] framework and a dynamically feasible motion primitive based state space representation [4, 5]. MDP is a natural framework to formulate the problem of trajectory planning under motion uncertainties [1, 3]. The use of motion primitives in MDP based framework, thus, allows generating trajectories that explicitly consider the constraints imposed by the vehicle dynamics. This is unlike planning on a rectangular grid that might yield a dynamically infeasible trajectory. In this paper we incorporate vehicle dynamics and motion uncertainty into motion-primitives using Monte Carlo runs of high-fidelity 6 DOF dynamics simulation of interaction between USV and ocean waves. We represent uncertainty in motion primitives by using a state transition function. This function maps each possible discretized state of the vehicle and a given action to a list of possible resulting states with respective transition probabilities.

State transition function can be obtained, by either running field experiments or by using computer simulations. Performing field experiments is the most accurate method, but is expensive and may be infeasible when needed to be performed during a mission. Moreover, the number of experiments may be very large when multiple sea-states and vehicle dynamics parameters need to be taken into account, which is the case during long missions. Computer simulations are inexpensive and can be improved, by incorporating experimental data. The complexity of a mission and an environment require generating the state transition map on-line, based on the information gathered by the sensors during vehicle operation. It may be infeasible to run all possible simulations off-line (*i.e.* before the mission) to generate the state transition map. This is because, the sea-state is decided by several factors, namely, the amplitudes, frequencies, wave directions, and the wave numbers of the wave components forming the ocean wave [6, 7]. Each of these factors is continuous and has non-linear influence on the ocean wave. In addition to this, the ocean interacts with USVs in a non-linear fashion. The initial conditions for the simulations can thus, become combinatorially prohibiting for an off-line estimation of state transition map. A potential flow theory based high fidelity 6 DOF dynamics simulator can generate a fairly accurate state transition map and aid in generating both safe and low cost trajectories [8, 9, 10]. The major problem with using such a high fidelity simulator is the slow computational performance of the simulation. This is mainly because of the fluid to rigid body interaction computation. One way to accelerate the computations is by using parallelization. Performing parallel computations would require computing clusters to be placed on the base and communicating with the vehicle over a network, which might be generally unreliable due to possible communication disturbances. The alternative of placing clusters directly on-board might add to the weight of the payload, which may not be desired or generally not possible. Another way to perform on-board computing is by using GPU. Recently, expensive scientific computations in various robotics problems are performed using GPUs, which are very powerful and lightweight as well [11, 12, 13, 14]. In addition to using GPU computing, we also employ model simplification techniques [9] to further improve computational performance of the simulation. A faster computation of state transition map may be required for some parts of the mission, where computing speed-up available using GPU alone may not be enough. The accelerated dynamics simulation is then used to establish connectivity among the vehicle's discretized states to develop a state transition map represented using a connectivity graph. The trajectory planning problem is then solved using value iteration of the stochastic dynamic programming (SDP) [1].

The key contributions of this paper are: (1) incorporation of 6 DOF dynamics simulation in state transition map estimation, (2) use of GPU based parallelization schemes to make the simulation faster for on-line computation of state transition map, (3) incorporation of temporal coherence based model simplification techniques with GPU acceleration for computing state transition map even faster, and (4) solution of trajectory planning problem using developed state transition map data structure, so that the computed trajectory plan satisfies the dynamics constraints as well as handles motion uncertainties.

2. Literature Review

In this paper we focus on the following issues related to trajectory planning problem of the USVs under motion uncertainty, namely, (i) simulation of the USVs, and (ii) physics-aware trajectory planning algorithm. We assume that the state of the vehicle can be estimated perfectly at all times. We discuss the related work in both the areas in this section.

2.1. 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 namely (1) influence of rigid body motion on the flow of fluid in which it moves, and (2) the influence of the fluid motion on the rigid body motion. Simulation approach in which both the couplings are considered 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 representative techniques for fluid-rigid body interaction simulation. Two-way coupling based approaches not necessarily developed for vehicle and ocean interaction simulation are: Euler's momentum equation, Navier-Stokes law, Smoothed Particle Hydrodynamics (SPH) technique, and Lattice Boltzmann Method (LBM). In Euler's momentum equation based technique, the momentum equation is solved for fluids numerically. Batty *et al.* reported a computation time of 25 s per frame using a grid size of 60×90 [15]. Carlson *et al.* reported numerical solution of Navier-Stokes equations with computation time of 27.5 s per frame for a domain of size 64×64 [16]. 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 [17]. In LBM, 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 [18]. Geist *et al.* developed a real-time approach for wave surface generation and attained 25 Hz for grid of size 1024^2 [19]. 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 speed-up on GPU code with respect to multi-threaded CPU code [20]. Gladkov *et al.* reported direct simulation Monte Carlo (DSMC) for solving Boltzmann equations on per particle basis (to solve rarefied fluid flow problems) by using GPU to obtain speed-up by a factor of 65 over serial implementation.

A survey on boat simulation was reported by Beck and Reed [21]. Craighead *et al.* reported another recent survey on open source boat simulators [22]. Some of the key USV simulation techniques are RANS based techniques, strip theory based techniques, kinematic model based techniques, and potential flow theory based techniques. 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.*, [23]. Kim [24] reported computation time of 24 hours using 84 processors on Mauis IBM-SP3 computer for 360 simulation time steps. Some of the other implementations of RANS code can be found in [25, 26, 27]. There are many research papers reported in the area of the underactuated controller design for the USVs that utilize 3 DOF simplified models which neglect the rolling, pitching, and heaving motions [28, 29, 30, 31, 32, 33]. Strip theory is mainly used for slowly moving slender geometries [34, 35, 36]. In potential flow theory, fluid flow is assumed to be irrotational and inviscid [37]. Potential flow theory based techniques are used by several researchers to perform the motion simulation of USVs [8, 9, 10]. Thakur and Gupta reported real time computational performance of USV simulation using clustering, temporal coherence, and multi-core parallelization based model simplification techniques in potential flow theory based simulation framework [9].

In nutshell:

- Euler's equation, Navier-Stokes, and RANS equation based techniques yield highly accurate results but one of their limitations is the dependence of the computation time on the domain size and the slow speed of computation.
- SPH technique results into good quality animations but the problem with the approach is the requirement of a large number of particles to simulate the fluid which in turn increases the computational time.
- The 3 DOF simulations are computationally very fast since they neglect the effect of fluid flow on the roll, pitch, and yaw and as a result are not accurate enough.
- Strip theory based techniques are computationally very fast but are not suitable for taking into account the variations in 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 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. We thus, use potential flow theory based model in this paper.

2.2. *Physics-Aware Trajectory Planning*

There is a large body of literature [1] in robot motion planning and we are presenting here only a review of representative research papers. We shall mainly focus on research related to robot trajectory planning considering differential constraints or in other words physics-aware robot trajectory planning. The literature for trajectory planning under differential constraints falls into following categories [38] 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), (4) mathematical programming, and (5) model predictive control (MPC). In state space sampling techniques, the robot state space is discretized and then searched for the low cost collision free trajectory. Several schemes for state space discretization have been reported. In a simple grid based approach, the state space is discretized into regular cells and trajectory is searched in that space [39]. In navigation function based approach, a navigation function is defined over the discretized state space and determined using algorithms such as value iteration followed by determination of a trajectory. Interpolation is used [40] to make the planning domain continuous. In rapidly exploring random trees (RRT), a biased stochastic search is performed in configuration space to generate a search tree [41, 42, 43]. In decoupled approach, planning is executed in two phases. In the first phase, a discrete path or set of waypoints [44] is determined using graph search technique such as A* [45, 46] on a discretized representation of the state space 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 waypoints using gradient based optimization approaches. Suzuki *et al.* used A* based approach for waypoint generation and RTABU search based optimization approach for trajectory generation [45]. 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 [47]. 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 [48]. Some other related works can be found in Refs. [4, 49, 50]. In mathematical programming approach, trajectory planning problem is posed as a numerical optimization problem with the robot dynamics as constraints and solved using techniques such as mixed integer linear programming, nonlinear programming, and other constrained optimization techniques [51, 52, 53, 54]. In model predictive control, the trajectory planning problem is posed again as an optimization problem, but optimized over finite horizon. This way the solution obtained is suboptimal but takes lesser computation time than optimizing over infinite horizon [55, 56, 57].

Under motion uncertainty, MDP framework is used to express the trajectory planning problem and solved using dynamic programming (DP) algorithms [58]. However, since the state space of a planning problem under motion uncertainty is usually very large, most of the practical algorithms have been developed [59, 60, 61] to compute an optimal or close-to-optimal solution to the problem by running value iteration over a carefully chosen subset of the state space.

In the USV trajectory planning domain a three layered architecture for Dijkstra algorithm based global planning and A* based local planning is presented by Casalino *et al.* [62]. 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 [63]. Soltan *et al.* developed nonlinear sliding mode control based trajectory planner for a 3 DOF dynamics model [64]. 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 [65]. Autonomous guidance based on feedback control is developed by Sandler *et al.* [66].

In nutshell:

- Most of the trajectory planning algorithms described above assume deterministic environmental conditions or conservatively approximated 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 or motion primitives and employ simulations to estimate state transition probabilities.

One of the challenges in incorporating simulation based state transition map is slower computational speed. Dynamics simulation based state transition map estimation is computationally slow due to the fluid-rigid body interaction computations. This can be alleviated using model simplification techniques. In this paper, we focus on trajectory planning based on the Maneuver Automaton framework from which we use only maneuvers, not trims, similar to the lattice based planning in Ref. [4]. We extend this framework to consider the motion uncertainty due to ocean waves using MDP framework. A similar approach for trajectory planning algorithm for unmanned balloons under the influence of stochastic winds (by estimating transition probabilities) have been developed by Wolf *et al.* [67].

3. Problem Statement and Solution Approach

3.1. Problem Statement

Given,

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

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

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

(iv.) goal state x_G, and

(v.) obstacle map Ω such that,

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

compute following:

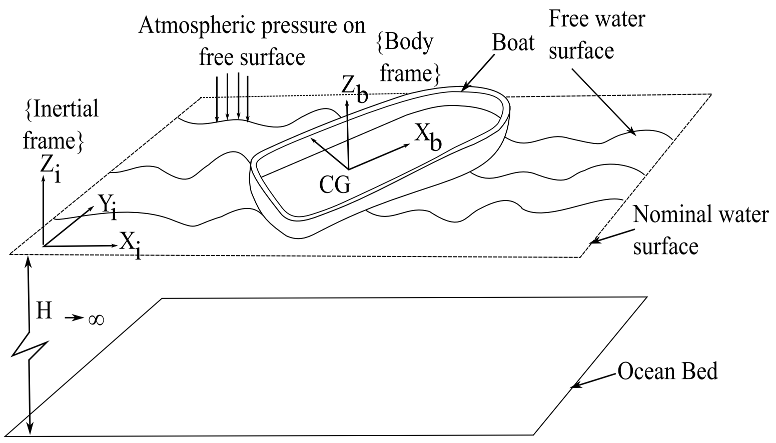


Figure 2: Description of coordinate systems used in the presented model: Inertial and body coordinate systems are shown.

(i.) State transition map over X and u: 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 probability $p(x_k|x_i, u_i)$, $\forall x_k$, $x_i \in X$, and $u_i \in u$. Perform GPU based computing acceleration and develop model simplification techniques for on-line estimation of 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 target location x_G from any given starting location of the USV. 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 Ref. [1]. We assume perfect state information is available at all times.

3.2. Approach

The approach is enumerated in the following steps.

(i.) Enhance the given USV 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 USV to find the optimal trajectory from each discretized state $x \in X$.

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

4. Dynamics Simulation of USVs

In this section we present the governing equations of the implemented dynamics model [9]. We extend the equations to handle the arbitrary number of wave components and to incorporate uncertainty into the system.

4.1. Motion Equations: Interaction Between USVs and Ocean Waves

We implemented the 6 DOF dynamics model for vehicles given by Fossen [7]. In this model, a vehicle is assumed to be a rigid body. The coordinate system used in the model is shown in Figure 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 USV is attached to boat's center of gravity (CG).

The governing dynamics equation of boat's motion in ocean waves is given in Equation 1.

$$\begin{aligned} &M_H\dot{v} + C_H(v)v + D_H(v)v + g(p) = F_E + F_P \\ &\dot{p} = J_p(v) \end{aligned} \tag{1}$$

where,

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

$v = [v_x, v_y, v_z, \alpha_x, \alpha_y, \alpha_z]^T$ 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 ms^{-1} and $v_r = [\alpha_x, \alpha_y, \alpha_z]^T$'s is angular velocity in $rads^{-1}$,

$R = \begin{bmatrix} c_y c_z & s_x s_y c_z - c_x s_z & c_x s_y c_z + s_x s_z \\ c_y s_z & s_x s_y s_z + c_x c_z & c_x s_y s_z - s_x c_z \\ -s_y & s_x c_y & c_x c_y \end{bmatrix}$ 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,

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

$\vec{x}\times \equiv S(\vec{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$ 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 CG in the body frame of reference,

m is mass of the USV in kg,

I_b is 3×3 matrix representing the inertia tensor of the USV 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 USV,

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

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

where ρ is density of water in kgm^{-3}, L_x, L_y, and L_z are length, width and height of the bounding box of the hull in m respectively,

$M_H = M_{RB} + M_A$ is (6×6) matrix representing 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 vehicle interaction force expressed in the body frame in N, and

m is mass of boat in kg,
F_P is 6×1 actuation force vector expressed in the body frame in N.
The fluid flow is computed based on potential flow theory [37]. Based on potential flow theory, the ocean wave is represented using a spatio-temporally varying height field η composed of Q wave components and given by the following equation.

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

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 potential ϕ is computed using the following equation.

$$\phi = \sum_{j=1}^{Q} \frac{gA_j}{\omega_j} \exp(k_j z) \sin(k_j x \cos\theta_{w,j} + ky \sin\theta_{w,j} - \omega_j t + \psi_j) \tag{3}$$

Velocity potential is then used to compute force acting on USV due to ocean wave using following equation.

$$F_W = \begin{bmatrix} \rho \oint_{S_B} \left[\frac{\partial\phi}{\partial t} + 0.5\nabla\phi.\nabla\phi\right] d\vec{S} \\ \rho \oint_{S_B} \left[\frac{\partial\phi}{\partial t} + 0.5\nabla\phi.\nabla\phi\right] \left(\vec{r} \times d\vec{S}\right) \end{bmatrix} \tag{4}$$

where S_B is instantaneous wet region of the USV. We assume that force acting on boat is only due to ocean waves ($F_E = F_W$) and ignore forces due to currents, wakes, etc. However, given a suitable model, simulation framework is capable of taking other types of forces into account.

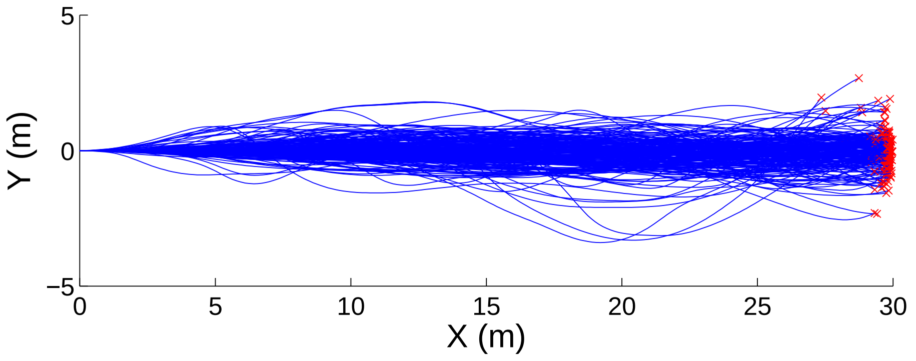
The term F_P on the right hand side of Equation 1 is the force due to the actuation (the thrust and the rudder angle). There are many actuator models available for the USVs and can be plugged into Equation 1 [7, 8]. We used a model given in Equation 5.

$$F_P = \left[K_1 \parallel n_{prop} \parallel n_{prop}, 0, 0, 0, 0, K_2 * K_1 \parallel n_{prop} \parallel n_{prop}\theta_{rud}\right]^T \tag{5}$$

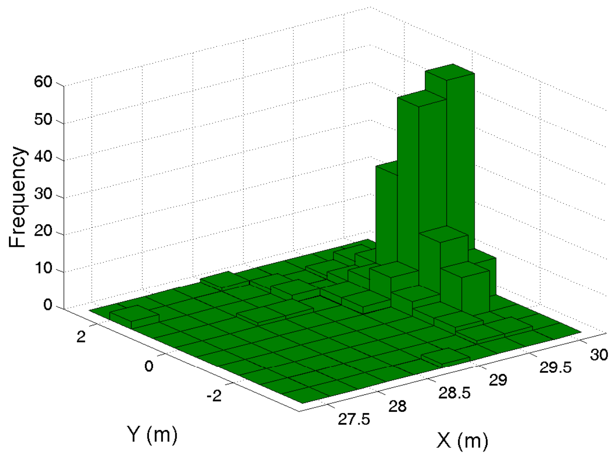
where K_1 is a constant and we chose it to be 1000, n_{prop} is the propeller's rpm, K_2 is a constant and we chose it to be 10, and θ_{rud} is the rudder angle.
We can express the parametric form [1] of the 6 DOF model as follows:

$$\dot{x} = f(x, u) \tag{6}$$

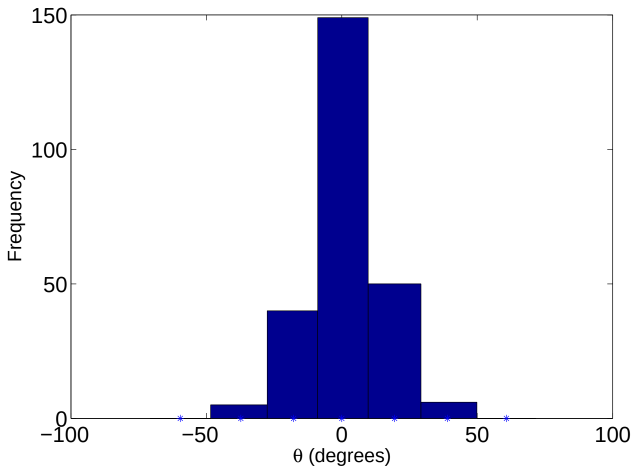
where $x = \left[p^T \; v^T\right]^T$ is the state of the USV,
$u = \left[v_f \; \Theta\right]^T$ is the commanded control action to go with forward velocity of v_f at a heading angle of Θ, and
$f = \left[(M_H^{-1}(-C_H(v)v - D_H(v)v - g(p) + F_E + F_P)^T \; J_p(v)^T\right]^T$.
We specify the desired 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 paper we assume that the USVs 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 PID controller.
We chose PID controller because of its widespread use and ease of implementation, however, in order to execute the commanded control actions any other controller such as the backstepping [68], sliding mode, etc. can be used [28].



(a) Sample trajectories.



(b) Frequency distribution of USV's ending positions for sample trajectories.



(c) Frequency distribution of USV's ending orientations for sample trajectories.

Figure 3: Uncertainty in USV's motion model for action along X axis generated using sample size of 256 (computed for sea-state 4 with average ocean wave height of $1.8m$ and boat moving at the velocity of $3ms^{-1}$).

4.2. Uncertainty in the Motion Model

In Section 4.1, the ocean wave (the ocean wave height and the velocity field) was initialized using given wave amplitudes, frequencies, and directions. An ocean wave, once initialized evolves deterministically with time and can be predicted exactly using the solution of Laplace equation (see Equation 2) [37]. However, the ocean waves initialized with the same parameters might look different due to the presence of the uniformly random phase lags ψ_j between each wave component. This leads to prediction of slightly different trajectories in each simulation run of the USVs 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 3(a), in which the USV is acted upon by a PID controller to move along a straight line for 256 different simulation runs in ocean wave built up of identical wave components. The variation in the trajectories of the USV in each simulation run is due to the uncertainty introduced by random phase lags (see Equation 2) in the ocean wave components despite of the other ocean parameters and the PID control objective being exactly identical. Figure 3(b) shows the histogram of final positions reached by the USV when commanded to reach at $(30, 0)$ with an orientation of 0^0 due to the disturbances caused by the ocean waves. Figure 3(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:

$$\dot{x} = f(x, u, w) \tag{7}$$

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

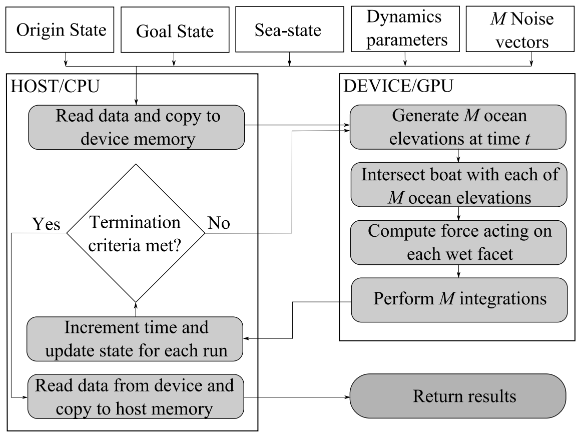

Origin State
Goal State
Sea-state
Dynamics parameters
M Noise vectors
HOST/CPU
Read data and copy to device memory
Termination criteria met?
Yes
No
Increment time and update state for each run
Read data from device and copy to host memory
DEVICE/GPU
Generate M ocean elevations at time t
Intersect boat with each of M ocean elevations
Compute force acting on each wet facet
Perform M integrations
Return results

Figure 4: Implementation of USV simulator on GPU.

4.3. Simulator Implementation on GPU

As described in Section 4.2, and shown in Figure 3, the USV 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 the USV's motion. In this section, we describe Monte Carlo simulation based approach to estimate the influence of nondeterministic effects of ocean on USV's 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 [9]. We describe the enhancements made in the potential flow theory based simulation model [8, 9, 10] 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 7 for M Monte Carlo simulation runs as follows:

$$\dot{X} = F(X, U, W) \tag{8}$$

where, F is the modified dynamics function representing simultaneous simulation runs.

Let $F_{WT} = (F_{W,1}^T, F_{W,2}^T, ..., F_{W,M}^T)$ 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 (see Figure 4).

Algorithm 1 - GPU based Monte Carlo Simulation of USV's dynamics
Input

(a.) Initial state vector tuple of USV 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 USV, and

(j.) sets of amplitudes A_q, frequencies ω_q, directions θ_q corresponding to each wave component, where index q varies from 0 to $Q-1$,

Output Set of M trajectories

Steps

(i.) Initialize the state vector tuple of USV $X = X_0$, phase lag tuple W, time $t = 0$, and trajectory length vector $L = [0, 0, ..., 0]^T$. Copy all configuration variables such as ocean wave parameters, dynamics parameters and geometry parameters to constant memory cache of GPU so that data is not required to be transferred in each simulation time step.

(ii.) Transform the USV 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 the number of polygonal facets representing the USV's geometry. We perform computations of this step on GPU.

(iii.) Determine the instantaneous wet surfaces ($S_{B,j}$) of the USV by finding out the facets lying beneath and on the wave surface (computed by superimposing given Q ocean wave components using Equation 2) corresponding to j^{th} phase lag vector w_j and use Equation 4 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 Equation 5.

(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 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 simulated instances of USV are within the radius of acceptance r from the respective target positions then return M trajectories else go to step (ii).

4.4. *Simulation Results of GPU Based Parallelization*

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 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 USV 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 6 components having amplitude of 0.2 m while rest 14 with amplitude of 0.1 m. 16 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 *rad* . 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 1 shows the comparison of the computational performance on GPU as compared to the CPU based computation. OpenMP based multi-threading enabled 85% average CPU usage, while running the baseline simulations. GPU based approach resulted into speed-up by factor ranging from 3.8 to 14.0, for the presented test case. Table 1 also shows that the speed-up factor increases with increase in the number of Monte Carlo runs, because of the highly data parallel nature of the computations. For larger number of simulation runs, the cost of memory transfer is appropriated and hence, speed-up is larger compared to smaller number of runs.

Table 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)	Speed-up
1	13.7	3.6	3.8
2	27.1	5.3	5.2
4	54.2	8.0	6.8
8	107.4	13.0	8.3
16	214.4	22.0	9.8
32	425.6	36.9	11.5
64	846.6	65.4	12.9
128	1691.2	123.8	13.7
256	3386.3	241.9	14.0

4.5. Model Simplification on GPU

More than 99% of the computation time in the USV simulation is spent in computing the forces acting on the USV due to the ocean waves [9]. The ocean is represented as a spatio-temporally varying heightfield in this paper. One of the major factors influencing wave forces is the variation in the wave heightfield in addition to the fluid velocity around the USV. 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 USV 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 idea of temporal coherence in a more concrete way, we define *instantaneous ocean heightfield* and *heightfield distance vector* as follows.

Definition 1 Let the ocean wave be specified by Q components of given amplitudes, frequencies, and directions. Let state vector tuple X denote the instantaneous states of the USV for each Monte Carlo simulation run, B_j be the bounding boxes of the USV located at the poses 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 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 2 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.

$$\vec{h}_d = ||G_{1,j} - G_{2,j}|| \tag{9}$$

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 USV 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 4.3). 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 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 some times, 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 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 the previous value of force else update $G_p = G$ and $dG_p = dG$ and return the decision to recompute force.

4.6. Results of Model Simplification on GPU

We chose $m = 2$, $n = 5$, $M = 256$, and $d\tau = 0.1$, and performed the simulations under identical ocean and USV dynamics parameter settings and varied τ from 0.00 to 0.10 in the increments of 0.025. The computation speed-up 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(a). The mean square error in force computation introduced due to increasing threshold τ is shown in Figure 5(b). The temporal coherence based simplification introduces errors in the computation of the final pose of the USV in Monte Carlo runs. Figure 5(c) shows the variation in the Euclidean distance of final positions of USV from the nominal position and the difference of each final USV orientations from the nominal orientation obtained by the Monte Carlo simulation runs. The nominal pose is the commanded pose to which USV should reach if there are no disturbances. In the test case, the nominal position is $(30, 0)$ and the nominal orientation is $0\ rad$. The variation in distance and orientation difference increases with increasing threshold. This is because, greater the threshold, fewer number of times force is computed and hence, more will be the inaccuracy.

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.

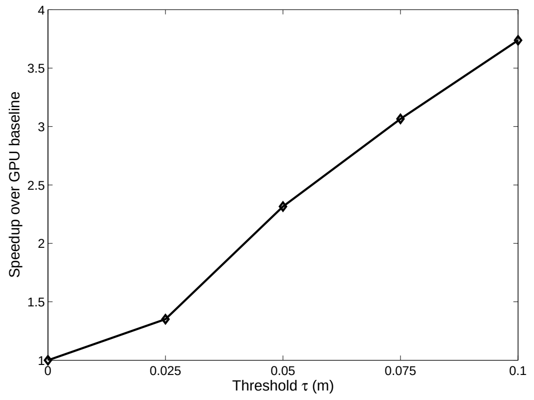
It should be noted in Figure 5, that the computing gains increase slowly in the range $0 < \tau < 0.025$, because, for smaller threshold, algorithm is unable to reuse force values computed in the previous steps and owing to the same reason, the variation in the final pose and nominal pose is also comparatively less pronounced. For 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 6 compares the computational gains obtained using temporal coherence on the GPU and CPU based simplification [9] approaches with the baseline computed on CPU (see Table 1). The main observations and respective analysis from the Figure 6 are explained below.

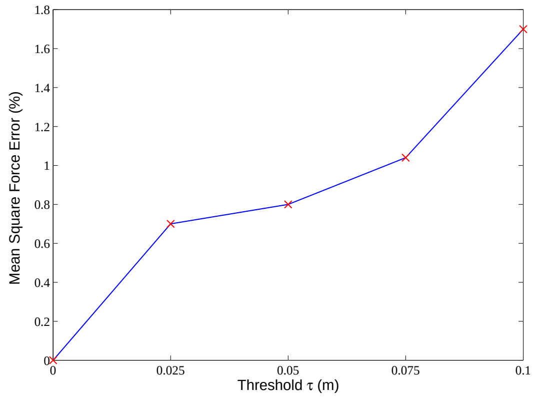
(i.) The computational speed-up factor obtained using temporal coherence on GPU is in the range of 4.7 to 43.1 and increases with the number of Monte Carlo runs. The increase in the speed-up can be attributed to the fact that the main computational cost of GPU operations is the data transfer between GPU and CPU. In the USV 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 speed-up factor increases.

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

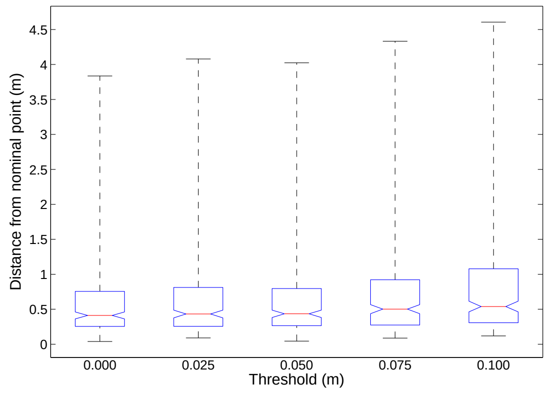
(iii.) The error associated with the model simplification performed on GPU and CPU is computed by taking the mean squared percentage error between the time series of the computed forces using the simplified method and the baseline method [9]. The model simplification based on temporal coherence implemented on GPU



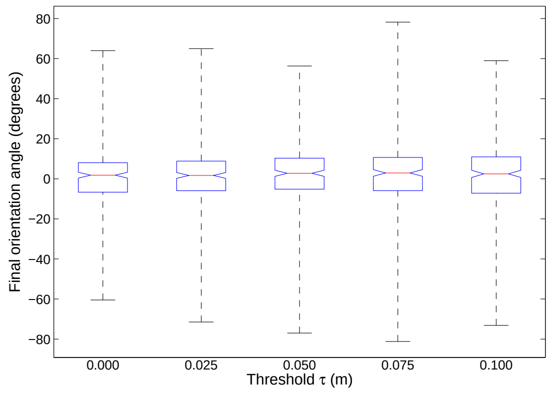
(a) Computing speed-up over GPU baseline vs threshold.



(b) Variation of mean square force error with threshold.



(c) Variation of distance from nominal point vs threshold.



(d) Variation of orientation from nominal point vs threshold.

Figure 5: Model simplification results of GPU based temporal coherence.

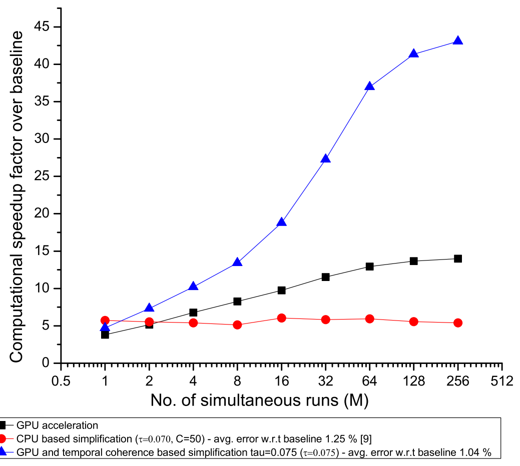


Figure 6: Results of GPU acceleration.

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 speed-up using model simplification algorithms based on clustering and temporal coherence on CPU [9] for simplification parameters $C = 60$, $\tau = 0.070$, and $d\tau = 0.1$. The approximation parameters C, τ, and $d\tau$ are chosen such that the errors due to GPU and CPU based simplification is similar, for fair comparison of associated speed-up in each case. Model simplification performed on CPU lead to an average factor of speed-up of 6.4 and average force error of 1.25% over the CPU baseline. It can be seen in Figure 6 that for single run the CPU based simplification approach outperforms the purely GPU based approach by a factor of 1.5 and for two runs the CPU based simplification approach is better than purely GPU based approach by a factor of 1.1. 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 USV 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 speed-up 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 speed-up factor (in case of Figure 6, about 43.1 with average mean square force error of 1.04%).

(v.) Figure 6 also shows that the computing speed-up 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. Application of Generated State Transition Map in Trajectory Planning of USVs

In this section we present the application of state transition probabilities obtained from model simplification techniques, developed in Section 4 in the area of trajectory planning of USVs.

5.1. MDP Formulation

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

5.1.1. State-Action Space Representation

For the dynamics computations, the state of the USV is defined as an augmented vector of pose and velocity according to Equation 7. The sizes of the vectors representing pose and velocity are 6 each, making the size of the state vector as 12. In addition, the USV state-action space is continuous and it is very difficult to search an optimal policy, in such a high dimensional and continuous space. In this section we present the state-action space dimension reduction and a suitable discretization to pose the problem of trajectory planning of USVs as a MDP.

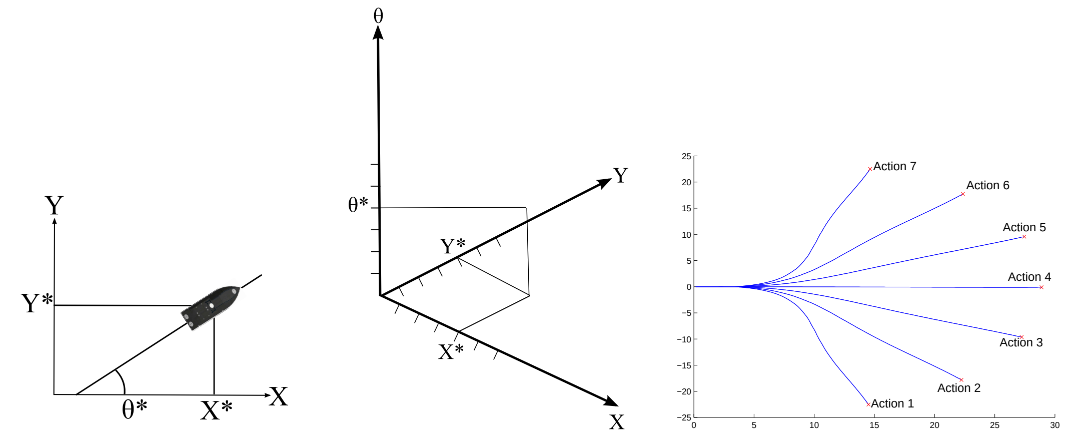
The motion goals for the USVs are usually specified in terms of the target pose $[x, y, \theta]^T$) and the target velocity $\left[v_x, v_y, \omega_z\right]^T$ on the ocean's nominal water plane. This means that the state space for the MDP can be reduced to $\left[x, y, \theta, v_x, v_y\ \omega_z\right]^T$. During operation the velocity of USVs do not change significantly during the operations and that justifies the choice of the constant desired velocity of the boat. Also we tuned the PID controller such that the angular velocity is kept within a set bound. The sway velocity 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 12 DOFs. The state space for the planning purposes in this paper is thus, reduced to a 3-tuple given by Equation 10.

$$s = [x\, y\, \theta]^T \tag{10}$$

where (x, y) are the coordinates of the USV's CG in the XY plane, and θ is the orientation of the boat in the XY plane.

We chose the grid dimension to be 15.0 m (USV length is nearly equal to 12 m). The orientation discretization is chosen to be 0.524 rad. The state space is depicted in Figure 7, in which the XY plane contains the location of the CG and the θ axis denotes the orientation of the boat. Pose of the boat in the XY plane is shown in Figure 7(a) and the corresponding location in the 3-D state space is shown in Figure 7(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 7 radial pose vectors having radial distance of 30.0 m with final desired steering angles varying from -2.142 to 2.142 rad in the increments of 0.428 rad. We chose the desired path lengths and the steering angles by first running the boat for 10 s along polar directions. Using this technique we generated a set of 7 waypoints which sufficiently cover the space around the boat and are dynamically reachable in 10 s.

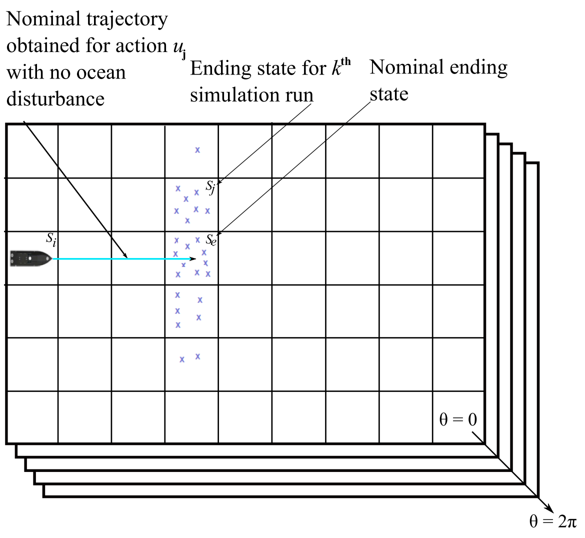


(a) Typical pose of USV in XY plane.

(b) Location of USV in state space.

(c) Action discretization.

Figure 7: State-action space of MDP.


Nominal trajectory obtained for action u_j with no ocean disturbance
Ending state for k^{th} simulation run
Nominal ending state
s_j
s_i
s_e
θ = 0
θ = 2π
Figure 8: State transition map computation.

5.1.2. *State Transition Map Computation*

The state transition map for the continuous space was expressed in the form of Equation 7. In this section we shall describe the computation scheme to determine the state transition map for the given USV in the discrete state space. We first present the definition of the state transition probability.

Definition 3 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 8 shows a USV 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 USVs are shown in the figure. The USVs trace a slightly different trajectory for each sample run due to the ocean wave and USVs interaction force as explained before (see Equation 4). The variation in the resulting states for a given initial state and an action yields a probability distribution over the resulting states.

We use a data structure similar to state lattice [4] to consider the dynamics but enhance it 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 motion uncertainty. The steps to compute the state transition map are described below.

Algorithm 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 USV operation.

Output State transition map.
Steps

(i.) Perform geometric transformation of each trajectory in T. Figure 8 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 vehicle that vary from 0 to 2π *rad*. For each simulation run (beginning from s_i and taking action u_j), the USV ends up in different states, shown by crosses in Figure 8. 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.

(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 8, 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 dynamics constraints. Each connection has a probability associated with it, due to the system dynamics and the presence of uncertainty due to ocean waves. 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 data structure very flexible in the sense of capturing the dynamics constraints and the environmental uncertainty for extreme situations such as rough sea-state.

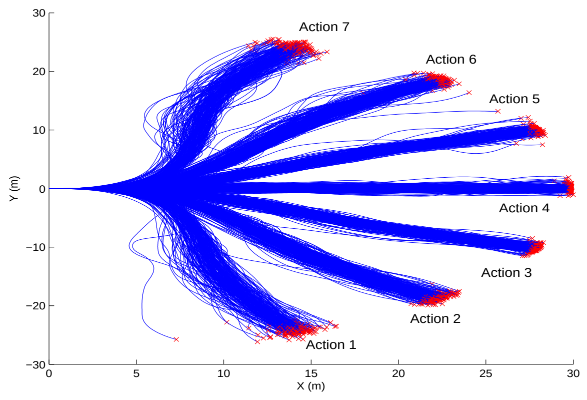
5.1.3. Reward Function

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 USV 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 USV for each action in the control set to determine the reward.

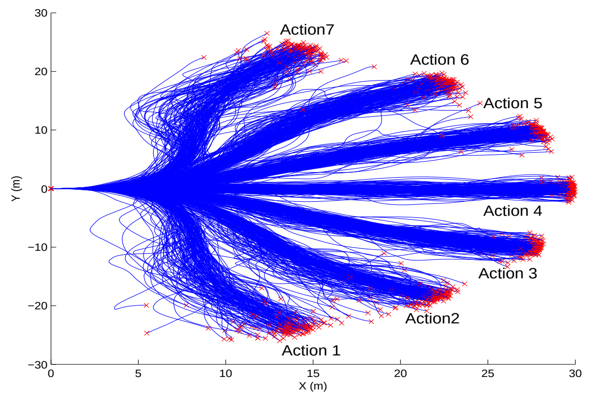
5.2. Results of Trajectory Planning

For the given action set Υ described in Section 5.1.1, 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 [7]. The resulting set of trajectories, for 256 Monte Carlo simulation runs is shown in Figure 9. The variation in the resulting trajectories due to different actions in the action set is shown in Figures 9(a) and 9(b) for sea-state 3 and 4 respectively. Variations in the final orientation in each Monte Carlo run due to ocean uncertainty for various actions are shown in Figures 9(c) and 9(d). 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 used $\tau = 0.075$ and $d\tau = 0.1$ to obtain Figure 9. The time taken to compute the set of trajectories for each sea-state was 9.1 minutes. We chose sample size of 256 because the variability of the pose data can be captured using a sample size of around 256. This is illustrated in Figure 10.

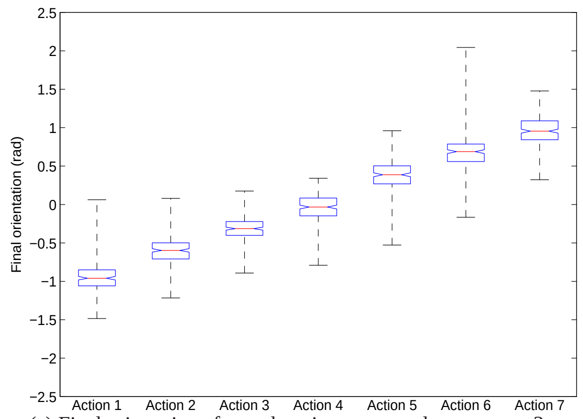
The MDP formulated in the Section 5.1 can be solved using numerous algorithms including the value iteration, policy iteration, hierarchical techniques, approximate techniques, etc. [69]. We chose the value iteration for computation of the solution. The details of the value iteration algorithm could be found in many references including [1, 3, 69]. The policies for each sea-state are computed using the value iteration over the obtained respective state transition maps. The optimal policies are then used for determining the trajectory to the target. Even if, the USV is deviated from planned trajectory (mainly, the orientation is changed significantly in high sea-state) due to forces caused by ocean waves, the computed policy can be used to regenerate an optimal trajectory to the target state. The obtained trajectory is optimal given the uncertainties (caused by ocean waves and 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 paper 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 [2, 69]. It should be noted that the optimality is achieved in the



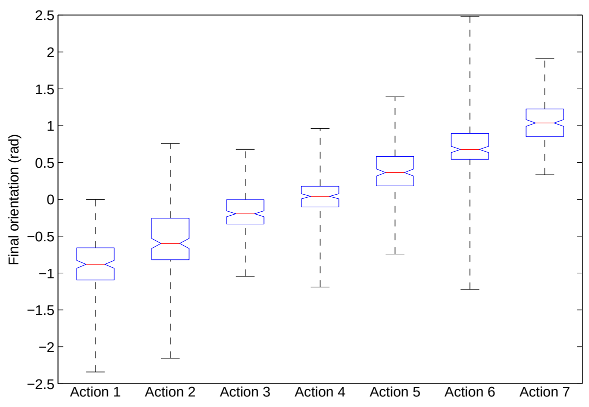
(a) Trajectories for each action computed at sea-state 3.



(b) Trajectories for each action computed at sea-state 4.

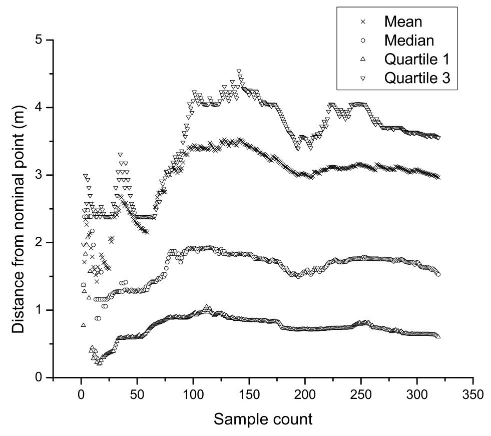


(c) Final orientations for each action computed at sea-state 3.

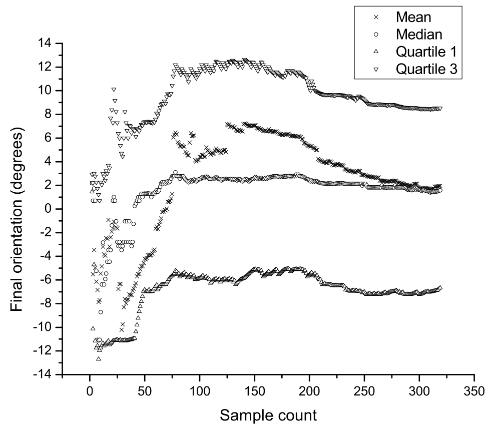


(d) Final orientations for each action computed at sea-state 4.

Figure 9: Control set computed for different sea-states.



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

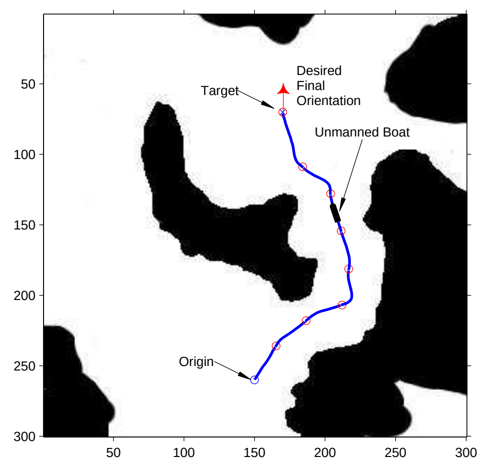


(b) Variation in sample median and quartile orientation from nominal orientation.

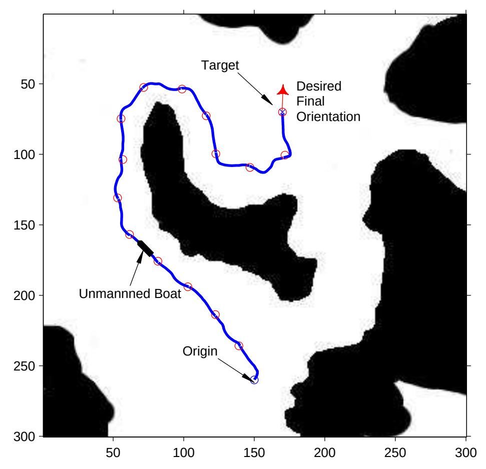
Figure 10: Variation in sample distance and orientation from nominal pose vs sample size.

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 11. The origin and target orientation of the vehicle is $\frac{\pi}{2}$. In case of sea-state 3, a shorter and riskier trajectory (through a narrow passage in the midst of obstacles) is computed by the approach discussed in this paper as shown in Figure 11(a). In case of sea-state 4 a longer but safer path is computed as shown in Figure 11(b). It should be noted that because of the dynamics constraints built into the search graph through the motion primitives discussed in this paper, the generated trajectories are always dynamically feasible. This is the reason that the trajectory in case of Figure 11(b)



(a) Trajectory computed between origin and target states at sea-state 3.



(b) Trajectory computed between origin and target states at sea-state 4.

Figure 11: Executed trajectories obtained by using feedback plan for sea-states 3 and 4. The feedback plans are computed using the algorithms developed in this paper

Table 2: Summary of computational performance obtained using GPU and temporal coherence based simplification over CPU computations. Number of actions in action set $P = 7$ and number of simulation runs $M = 256$.

	CPU Baseline	GPU Baseline	CPU with clustering ($C = 50$) and temporal coherence ($\tau = 0.070$)	GPU with temporal coherence ($\tau = 0.075$)
Computation time (min)	395.0	28.2	80.2	9.1
Speed-up factor over CPU baseline	1.0	14.1	4.9	43.4
Error introduced (%)	0.0	0.0	1.25	1.04

bends near the target, as the vehicle needs to go down in order to turn to the desired orientation of $\frac{\pi}{2}$. 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 USV 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 paper, 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 comparison of computational performance of GPU and CPU based model simplification techniques are shown in Table 2. The time taken to perform Monte Carlo simulations of the USV dynamics to compute state transition map using the algorithm developed in this paper was 9.1 minutes. It should be noted that the CPU baseline was computed by implementing the parallel version (using OpenMP) of the simulator code reported in Ref. [9]. The overall speed-up using GPU in conjunction with temporal coherence based model simplification technique was by a factor of 43.4 by introducing an error of 1.04%. 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.

6. Conclusions

In this paper, we presented GPU based algorithms to compute state-transition probabilities for USVs using 6 DOF dynamics simulations of interactions between ocean-wave and vehicle. We used the obtained state transition model in standard MDP based planning framework to generate physics-aware trajectory plans. We extended the dynamics model for USV simulation, reported in Ref. [9], to incorporate multiple wave components and random phase lags in ocean waves. The approach described in this paper is flexible and is capable of handling any USV geometry, dynamics parameters, and sea-state. We developed GPU based algorithm to perform fast Monte Carlo simulations to estimate state transition probabilities of USVs operating in high sea-states. We further improved the computational performance by developing model simplification algorithm based on temporal coherence. The overall computational speed-up obtained is by a factor of 43.4 by introducing a simulation error of 1.04% over the CPU baseline. Transition probabilities can be computed within 9.1 minutes by using the algorithms described in this paper. The paper also presents a case study to demonstrate the application of the developed state transition map to generate physics-aware trajectory plans for sea-states 3 and 4 in MDP based planning framework.

A limitation of the approach is that we use potential flow theory based model, which is not suitable for simulating planing phenomenon occurring in fast moving boats. The approach presented in this paper, however, is flexible to incorporate any other fluid flow model if available, to compute state transition model that can subsequently be used in trajectory planning. Another limitation is that the current framework is only capable of handling dynamic environmental disturbances with static obstacles, such as islands and shorelines. The trajectory planner, however, cannot handle dynamic obstacles with a sufficient efficiency, and a faster replanning approach can be used to tackle with this issue [70].

References

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

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

[3] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2005.

[4] 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.

[5] 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.

[6] O. M. Faltinsen. *Sea loads on ships and offshore structures.* Cambridge University Press, Cambridge, New York., 1990.

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

[8] 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.

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

[10] 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.

[11] J. Betz. Solving rigid multibody physics dynamics using proximal point functions on the GPU. Master's thesis, Rensselaer Polytechnic Institute, Troy, New York, 2011.

[12] J. Pan and D. Manocha. GPU-based parallel collision detection for real-time motion planning. In David Hsu, Volkan Isler, Jean-Claude Latombe, and Ming Lin, editors, *Algorithmic Foundations of Robotics IX*, volume 68 of *Springer Tracts in Advanced Robotics*, pages 211–228. Springer Berlin / Heidelberg, 2011.

[13] W. L. D. Lui and R. Jarvis. Eye-Full tower: A GPU-based variable multibaseline omnidirectional stereovision system with automatic baseline selection for outdoor mobile robot navigation. *Robotics and Autonomous Systems*, 58(6):747 – 761, 2010.

[14] J.T. Kider, M. Henderson, M. Likhachev, and A. Safonova. High-dimensional planning on the GPU. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2515 –2522, may 2010.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

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

[40] 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.

[41] 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.

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

[43] 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.

[44] R. Stelzer and T. Proell. Autonomous sailboat navigation for short course racing. *Robotics and autonomous systems*, 56(7):604–614, JUL 31 2008.

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

[46] M. Bennewitz, W. Burgard, and S. Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and autonomous systems*, 41(2-3):89–99, NOV 30 2002.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] A. Elnagar and A. Hussein. On optimal constrained trajectory planning in 3d environments. *Robotics and Autonomous Systems*, 33(4):195 – 206, 2000.

[52] 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.

[53] 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.

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

[55] 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.

[56] 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.

[57] O. Purwin and R. Andrea. Trajectory generation and control for four wheeled omnidirectional vehicles. *Robotics and Autonomous Systems*, 54(1):13 – 22, 2006.

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

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

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] M. Sandler, A. Wahl, R. Zimmermann, M. Faul, U. Kabatek, and E. D. Gilles. Autonomous guidance of ships on waterways. *Robotics and Autonomous Systems*, 18(3):327 – 335, 1996.

[67] 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.

[68] I. Zohar, A. Ailon, and R. Rabinovici. Mobile robot characterized by dynamic and kinematic equations and actuator dynamics: Trajectory tracking and related application. *Robotics and autonomous systems*, 59(6):343–353, JUN 2011.

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

[70] 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.