Toward Maritime Robotic Simulation in Gazebo

Brian Bingham
Naval Postgraduate School
bbingham@nps.edu

Carlos Agüero
Open Robotics
caguero@openrobotics.org

Michael McCarrin
Naval Postgraduate School
mrmccarr@nps.edu

Joseph Klamo
Naval Postgraduate School
mrmccarr@nps.edu

Joshua Malia
Naval Postgraduate School
joshua.malia@navy.mil

Kevin Allen
University of Florida
kmallen@ufl.edu

Tyler Lum
University of British Columbia
tyler.lum@alumni.ubc.ca

Marshall Rawson
University of Florida
marshallrawson@ufl.edu

Rumman Waqar
University of London
waqar@ualberta.ca

***Abstract*—Simulation plays an important role in the development, testing and evaluation of new robotic applications, reducing implementation time, cost and risk. For much of the robotics community, the open-source Gazebo robot simulator has emerged as the *de facto* standard environment for prototyping and testing robotic systems. While Gazebo offers strong support for terrestrial, aerial and space robotics applications, less support is available for marine applications involving vehicles at and below the water surface. To address this deficiency, we present the Virtual RobotX (VRX) simulation, a general purpose open-source development and testing tool, based on Gazebo, capable of approximating the behavior of unmanned surface vessels operating in complex ocean environments. We highlight the application of these capabilities using the VRX challenge reference implementation, a new simulation-based robot competition designed to complement the physical Maritime RobotX Challenge.**

I. Introduction

The open-source Virtual RobotX (VRX) simulator[1] is designed to support the development, testing and evaluation of unmanned surface vehicles operating in ocean environments. VRX extends the Gazebo robot simulator through the addition of new domain-specific elements including environmental models, vessel dynamics representations, sensor emulation and general purpose ocean object models. The new capabilities featured in the VRX simulator include:

- Wave representations, based on ocean spectra, to influence vessel motion, visual rendering and sensor feedback.
- Water surface visual representation, including approximation of reflection and refraction.
- Stochastic wind speed representation to influence vessel motion.
- Parameterized six degree-of-freedom surface vessel model.
- Approximation of buoyancy forces on geometric objects to simulate floating objects.
- Lidar (3D) simulation, including interaction with water surface.
- A configurable propulsion system with a parameterized non-linear thrust model.

These new capabilities are applicable to a wide range of ocean robotic scenarios. The specific reference implementation, which serves as a demonstration of these new features, is the VRX challenge, a new simulation-based robot competition designed to complement the physical Maritime RobotX Challenge. This competition uses a common vessel platform, the wave adaptive modular vessel (WAM-V), and includes a series of robotic tasks meant to stimulate innovation in maritime autonomy. To support this competition, the VRX simulator approximates a number of aspects of the WAM-V platform including a 3D visual model, dynamic and hydrodynamic characteristics, collision model, coefficients to characterize wind effects and non-linear static thruster behavior. The propulsion and sensor configuration is parameterized to allow users to easily specify the exact configuration of the vessel under test. In addition the VRX simulator provides models (visual, collision and dynamic) of the elements used in competition tasks, including navigation

The Maritime RobotX Challenge and Virtual RobotX Challenge projects are sponsored by the Office of Naval Research.

[1]https://bitbucket.org/osrf/vrx/

aids, buoys, obstacles and docks.

Although simulation is never a substitute for physical field experimentation, the VRX simulator offers sufficient fidelity to enable developers to prototype new solutions, with the aim of transitioning smoothly to on-water deployments for tuning and refinement. The models used in the simulator are designed so that the *kind* of response experienced in simulation is equivalent to that experienced in the field, even if the *degree* of the response is different. The result is a simulation environment simple enough to execute faster than real-time simulations with sufficient fidelity to guide choices on the kind of autonomy necessary for the salient challenges of the ocean environment. The simulator design adheres to the following rubric: autonomy that does not work in simulated world will not work in the physical world, and autonomy that does work in the physical world will work in the simulated world.

A. Related Work

1) Gazebo Simulation: Gazebo was created in 2002 to support development of ground robot applications for indoor and outdoor environments [1]. Since then Gazebo has become a mature open-source project that is developed and relied upon by the global robotics community for a wide variety of applications. Gazebo employs a highly modular approach to provide the four key components needed for robot simulation.

1) To resolve collision, contact, and reaction forces among rigid bodies, Gazebo supports the use of multiple physics engines.
2) Gazebo has an extensive library of common robot sensors, such as camera, laser, sonar, GPS, and IMU, as well as standard noise models that can be parameterized as needed.
3) Gazebo supports multiple interfaces allowing users to interact programmatically with the simulation, including C++ (for writing plugins), a custom network transport, and Robot Operating System (ROS) messaging.
4) Gazebo includes a graphical interface that allows exploration and manipulation of the 3D simulated world.

Gazebo has been extended to support a variety of domain-specific applications such as the DARPA Robotics Challenge (DRC) [2], the NASA Space Robotics Challenge (SRC) [3], the DARPA Hand Proprioception & Touch Interfaces (HAPTIX), the NIST Agile Robotics for Industrial Automation Competition (ARIAC), the DARPA Service Academies Swarm Challenge (SASC), the DARPA Robotic Servicing of Geosynchronous Satellites (RSGS) and NASA Lunar exploration [4].

2) Marine Robotics Simulation: While a standard simulator for marine robotics is not yet available, a number of groups have recognized the need for simulation as part of the development process and highlighted the unique challenges of marine simulations, particularly for underwater applications [5]–[7]. Some of these simulations build upon the foundation of Gazebo [8], [9] and others use different underlying simulation frameworks to emulate the dynamics. An overview of the current technology is provided by [10].

3) Wave Induced Forcing: The proposed approach to approximating wave induced motion uses evaluation of restoring forces at discrete locations along the vessel hull. A similar approach is used by [11] to generate representative ship motion for the purposes of operator training in virtual reality. A related method was used by [12] and validated using a higher fidelity, non-real-time model and experimental results. This approach is even further simplified in [13] to generate roll, pitch and heave motion by evaluating the wave height at just four locations along a mono-hull. In this previous work the vessel hull is approximated as a cuboid and the water height is evaluated at 1×1 m grid points. Single forces for heave, pitch and roll are obtained from the discrete height fields. Because of the geometry of the WAM-V, a catamaran with approximately circular demi-hulls, we can solve for the displacement directly as a function of hull location relative to water height.

II. Environment Modeling

A key aspect of extending the Gazebo robotics simulator to support maritime robotics is the ability to represent the influence of the ocean environment on the robotic system. The two important environmental influences we model are the waves and wind.

A. Wave Modeling

We adapt the approach proposed in [14] to represent ocean waves based on oceanographic statistical descriptions that are visually realistic. The model balances physical fidelity, visual realism and computational complexity.

1) Gerstner Waves: Gerstner waves, a common representation in computer graphics [15], [16], represent the water surface as a trochoidal shape. While closely related to a regular sinusoidal shape, Gerstner waves of larger amplitude exhibit sharper crests and broader troughs providing for additional visual realism. We model the water surface using a summation of Gerstner waves where an undisturbed horizontal location is $\mathbf{x}_0 = (x_0, y_0)$ with a vertical height of $\zeta_0 = 0$. The wave field is represented as horizontal ($\mathbf{x}$) and vertical (ζ) displacements relative to this undisturbed location, and can be expressed as

$$\mathbf{x}(\mathbf{x}_0, t) = \mathbf{x}_0 - \sum_{i=1}^{N} q_i(\mathbf{k}_i/k_i) A_i \sin(\mathbf{k}_i \cdot \mathbf{x}_0 - \omega_i t + \phi_i) \tag{1a}$$

$$\zeta(\mathbf{x}_0, t) = \sum_{i=1}^{N} A_i \cos(\mathbf{k}_i \cdot \mathbf{x}_0 - \omega_i t + \phi_i), \tag{1b}$$

where each component wave has an associated amplitude (A_i), wavenumber (k_i), angular frequency (ω_i), steepness (q_i) and random phase (ϕ_i). The wave number k is related to the wavelength λ by $k = 2\pi/\lambda$. The wavevector ($\mathbf{k}_i$) is a horizontal vector in the direction of travel with magnitude k_i. A single steepness parameter (q) is specified between $[0.0, 1.0]$ where a value of $q = 0.0$ yields regular sinusoidal wave shapes and a value of $q = 1.0$ yields maximum wave crest steepness. The individual steepness values are constrained by $q_i = \min(q, 1/(k_i A_i))$ to prevent loops forming in the wave crests, a phenomenon which is not physically realizable. We use the linear, infinite water depth dispersion equation to relate the angular frequency and wavenumber as $\omega^2 = gk$ where g is the acceleration of gravity.

2) Wave Spectrum: If visual fidelity were the sole consideration, empirically selecting a small set of component wave amplitudes and directions for use in (1) can generate qualitatively realistic wave fields, but with limited connection between the simulated wave field and the physical ocean environment. One method to generate component wave characteristics representative of a particular ocean and weather condition is to generate these parameters by sampling a parametric wave spectrum [14], [17], [18]. The wave spectrum $S(\omega)$ captures the mean energy in a wave field as a function of angular frequency (ω). A number of standard ocean spectra can be used to describe the wave field environment [19]. In [18] a single-value Pierson-Moskowitz wave spectrum is sampled to generate the wave field. The Pierson-Moskowitz spectrum represents a fully developed sea in deep water and depends upon specifying the peak angular frequency (ω_p) or the significant wave height (H_s) which are related by

$$H_s = \frac{0.162(g)}{\omega_p^2}. \tag{2}$$

We find that the two-parameter Pierson-Moskowitz spectrum (often referred to as a Bretschneider spectrum) provides both physical relevance and user customizability while maintaining a simple formulation. The two-parameter spectrum is specified by the peak frequency and an independent significant wave height $\bar{H}_s$:

$$S_B(\omega) = \frac{1.25}{4}\left(\frac{\omega_p^4}{\omega^5}\right)(\bar{H}_s)^2 \exp\left[-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right]. \tag{3}$$

We introduce a user-specified non-dimensional gain value,

$$K_H = \frac{\bar{H}_s}{H_s}, \tag{4}$$

as the ratio of desired significant wave height to the significant wave height from the one-parameter Pierson-Moskowitz spectrum (2). The resulting expression for the simulated wave field is

$$S_B(\omega) = (K_H)^2 \frac{\alpha g^2}{\omega^5} \exp\left[-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right] \tag{5}$$

where $\alpha = 8.1 \times 10^{-3}$. For the purposes of generating linear, sinusoidal waves from a spectrum, the amplitude of a particular wave component (A_i) is related to the spectrum by

$$A_i^2 = 2\int_{\Delta_{\omega_i}} S(\omega) d\omega \approx 2S(\omega_i)\Delta_{\omega_i} \tag{6}$$

where ω_i are the sample locations along the spectrum and $\Delta\omega_i$ is width of the frequency band associated with the individual samples.

The user interface for defining a particular simulated wave field consists of specifying the two-parameter Pierson-Moskowitz spectrum via the peak period, $T_p = (2\pi)/\omega_p$, and wave height gain multiplier K_H from (4). Example spectra are shown in Fig. 1 to illustrate the ability to independently specify the characteristic period (frequency) and wave height of the desired wave field. Once the spectrum is defined, the individual wave amplitude values are determined by sampling the wave spectrum according to (6).

3) Directional Spectra: For modeling the direction of wave components within the wave field we follow the general approach of [18], but pre-calculate the relationship between wave characteristics (period, frequency, etc.) and directionality in order to reduce computational complexity.

The direction of wave travel can be taken into account using the directional wave spectrum expressed as

$$E(\omega, \theta) = S(\omega) D(\omega, \theta) \tag{7}$$

where θ represents the wave direction and $D(\omega, \theta)$ is the directional spreading function.

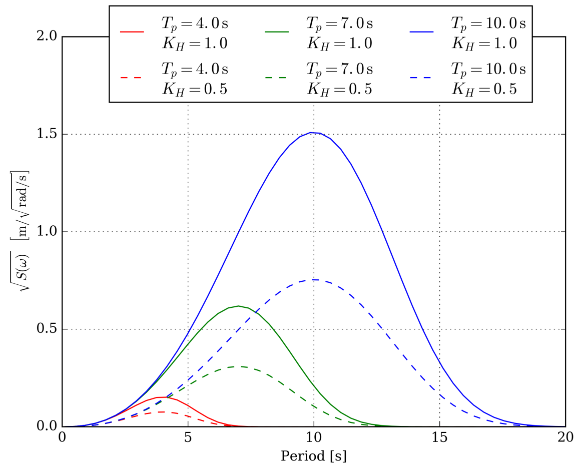

Fig. 1. Two-parameter Pierson-Moskowitz wave spectrum shown as a function of wave period for a series of peak period (T_p) values, using two different wave height gains (K_H).

The form of $D(\omega, \theta)$ is based on the empirical relations that were proposed in [20]. The directionality is a function of the normalized frequency $\bar{\omega} = \omega/\omega_p$ and can be expressed as a function of the difference between the angle and the mean angle, $\Delta\theta = \theta_m - \theta$. The directionality function is then

$$D(\bar{\omega}, \Delta\theta) = N(s(\bar{\omega})) \cos{(\Delta\theta/2)}^{2s(\bar{\omega})} \tag{8}$$

where the normalizing function

$$N(s(\bar{\omega})) = \left(\frac{1}{2\sqrt{\pi}}\right) \frac{\Gamma(s(\bar{\omega})+1)}{\Gamma(s(\bar{\omega})+1/2)} \tag{9}$$

uses the gamma function $\Gamma()$ and the function

$$s(\bar{\omega}) = \begin{cases} 17.01\,\bar{\omega}^5 & \text{if } \omega \leq \omega_p \\ 17.01\,\bar{\omega}^{-2.5} & \text{if } \omega > \omega_p \end{cases}. \tag{10}$$

The shape of the directionality depends upon the component frequency. Spectral components near the peak frequency contain the majority of the wave energy and have a more narrow directional distribution, tending to travel in the mean direction. Spectral components on either side of the peak frequency tend to have larger variation in direction of travel, in particular lower frequency (longer period) components have a wider distribution of directions than high frequency (shorter period) components.

To characterize the width of the directional spreading function we use the second moment

$$\mu_2(\bar{\omega}) = \int_{-\pi}^{\pi} (\Delta\theta)^2 D(\bar{\omega}, \Delta\theta) d(\Delta\theta) \tag{11}$$

which we evaluate numerically. This relationship is precomputed and used as a lookup table for the component waves which make up the wave field. For each wave component in (1) the direction of the wavevector is determined by the user-specified mean direction and an additional random direction component. This additional component is generated as a sample from a zero mean, normally distribution with a variance set equal to $\mu_2(\bar{\omega})$.

4) Specifying a Wave Environment: To summarize, the user specifies the directional two-parameter Pierson-Moskowitz spectrum through the peak period (T_p), wave height gain (K_H) and mean wave direction. The user also specifies the wave steepness (q). The remaining parameters of the Gerstner waves (1) are determined based on sampling the wave spectrum and using the linear deep water dispersion relation. The resulting wave field description provides a three dimensional representation of the ocean surface. The representation is used by both the visual and physical aspects of the simulator to excite the physical motion of bodies at the water surface and to generate the user interface and sensor (camera) views.

B. Wind Modeling

Wind is a significant disturbance for objects at the sea surface. We consider the total wind speed $V_w(t)$ consisting of the sum of the constant mean wind speed ($\bar{v}$) and the temporally varying, zero-mean, variable wind speed ($v_g(t)$), i.e., $V_w(t) = \bar{v} + v_g(t)$. The user specifies the following aspects of the wind:

- the constant wind direction (β_w),
- the constant mean wind speed ($\bar{v}$), and
- the standard deviation (σ_g) and time constant (τ_g) of the variable component of the wind speed.

The variable component of the wind speed is modeled as a first-order, linear approximation of the Harris spectrum [21] with time constant τ_g as reported in [22]. We express the spectrum as the transfer function

$$h(s) = \frac{K_w}{1 + \tau_g s} \tag{12}$$

where K_w is the magnitude of the response. In discrete time the variable wind speed at time $k+1$ is generated as

$$v_g[k+1] = v_g[k] + (1/\tau_g)(-v_g[k] + K_w n[k])\delta t \tag{13}$$

where $n[k]$ is a pseudo-random number generated with a Gaussian distribution, zero mean and unity variance and δt is the simulation time step. The spectrum of the variable wind speed generated by (13) is

$$S_g(\omega) = \begin{cases} \frac{K_w^2}{1+(\omega\tau_g)^2}\delta t, & |\omega| < \omega_s/2 \\ 0, & |\omega| \geq \omega_s/2 \end{cases} \tag{14}$$

where ω_s is the sampling frequency of $\omega_s = 2\pi/\delta t$. Assuming that $\omega_s >> 2\pi/T$, we can find the value of the parameter K_w necessary to produce the variable wind component with the user specified standard deviation and time constant as

$$K_w = \sigma_g \sqrt{\frac{2\tau_g}{\delta t}}. \tag{15}$$

III. Vehicle Model

To approximate the motion of a vehicle in the ocean environment, we adapt Fossen's six degree-of-freedom robot-like vectorial model for marine craft [23] expressed as

$$\begin{aligned}&\underbrace{\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{rigid body forces}} + \\ &\underbrace{\boldsymbol{M}_A\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r}_{\text{hydrodynamic forces}} + \underbrace{\boldsymbol{g}(\boldsymbol{\eta})}_{\text{hydrostatic forces}} \\ &= \boldsymbol{\tau}_{propulsion} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{waves}\end{aligned} \tag{16}$$

where

$$\boldsymbol{\eta} = [x, y, z, \phi, \theta, \psi]^T \tag{17a}$$

$$\boldsymbol{\nu} = [u, v, w, p, q, r]^T \tag{17b}$$

are position and velocity vectors respectively for surge, sway, heave, roll, pitch and yaw. The total velocity, $\boldsymbol{\nu}$, is the sum of an irrotational water current velocity, $\boldsymbol{\nu}_c$, and the vessel velocity relative to the fluid, $\boldsymbol{\nu}_r$, i.e., $\boldsymbol{\nu} = \boldsymbol{\nu}_r + \boldsymbol{\nu}_c$. The forces and moments due to propulsion (control), wind and waves are represented as $\boldsymbol{\tau}_{propulsion}$, $\boldsymbol{\tau}_{wind}$ and $\boldsymbol{\tau}_{waves}$.

Traditionally, surface vessel models are often separated into *maneuvering* models (representing the surge, sway and yaw degrees-of-freedom) and *seakeeping* models (representing the heave, pitch and roll degrees-of-freedom). For the purposes of supporting development of autonomy, it is important that the unified simulation model include both maneuvering and seakeeping components, but our approach to formulating each component is slightly different. The maneuvering aspects of the model influence the vessel motion control. Because the motion control should account for rigid-body and hydrodynamic terms, these aspects of the model include additional detail. If the model is to demonstrate the appropriate type of dynamics to exercise these aspects of motion control, it is necessary to include the added mass and non-linear damping hydrodynamic terms; neglecting these hydrodynamic terms would result not just a different degree of response, but a different kind of vessel response.

On the other hand, while inclusion of the seakeeping aspects of the model is critical for exercising the sensory perception capabilities of the autonomy solution, the hydrodynamics in this portion of the model can be simplified while preserving the appropriate type of response. In the seakeeping degrees-of-freedom we neglect the added-mass and non-linear damping terms without changing the kind of vessel response the simulation produces to induce sensor motion than can impact perception. Including these terms would change the degree of the response, perhaps with more fidelity, but not a different type response. For the intended uses of the VRX simulation these changes in degree do not justify the additional complexity and challenges associated with estimating these model terms empirically.

A. Hydrodynamic Forces

The hydrodynamic forces in (16) include the added mass terms due to inertia of the surrounding fluid and hydrodynamic damping. The hydrodynamic derivatives are expressed using SNAME (1950) notation [23]. The simplified added mass matrix is expressed as

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot u} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot v} & 0 & 0 & 0 & Y_{\dot r} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N_{\dot v} & 0 & 0 & 0 & N_{\dot r} \end{bmatrix} \tag{18}$$

where $N_{\dot v} = Y_{\dot r}$. The Coriolis-centripetal matrix for the added mass is expressed using the same hydrodynamic derivatives

$$\boldsymbol{C}_A(\boldsymbol{\nu}_r) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & Y_{\dot v} v_r + Y_{\dot r} r \\ 0 & 0 & 0 & 0 & 0 & -X_{\dot u} u_r \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -Y_{\dot v} v_r - Y_{\dot r} r & X_{\dot u} u_r & 0 \end{bmatrix}. \tag{19}$$

The hydrodynamic damping includes forces due to radiation-induced potential damping, skin friction, wave drift damping, vortex shedding and lifting forces [23], [24]. These effects are aggregated in the hydrodynamic damping matrix

$$\boldsymbol{D}(\boldsymbol{\nu}_r) = \boldsymbol{D}_l + \boldsymbol{D}_l(\boldsymbol{\nu}_r) \tag{20}$$

expressed as a sum of linear and quadratic terms:

$$\boldsymbol{D}_l = - \begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & 0 & 0 & Y_r \\ 0 & 0 & Z_w & 0 & 0 & 0 \\ 0 & 0 & 0 & K_p & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q & 0 \\ 0 & N_v & 0 & 0 & 0 & N_r \end{bmatrix} \tag{21}$$

and

$$\boldsymbol{D}_n(\boldsymbol{\nu}_r) = - \left[\begin{matrix} X_{u|u|}|u_r| & 0 & 0 \\ 0 & Y_{v|v|}|v_r| + Y_{|r|v}|r| & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & N_{|v|v}|v_r| + N_{|r|v}|r| & 0 \end{matrix} \right. \\ \left. \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & Y_{|v|r}|v_r| + Y_{|r|r}|r| \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & N_{|v|r}|v_r| + N_{r|r|}|r| \end{matrix} \right] . \tag{22}$$

This approximation of the hydrodynamic effects is implemented as parameterized Gazebo plugin. The user defines the characteristics of the vessel under test through a vessel-specific configuration file that includes the hydrodynamic derivatives. During each time step of the simulation, the plugin is executed with access to the state of the vessel and environment. The hydrodynamic force terms in (16) are calculated based on this state information and the user-defined vessel characteristics. The resulting force and moment values are then applied to the vessel through the Gazebo application programming interface (API) for inclusion in the next iteration of the physics engine.

This simplified six degree-of-freedom model and the Gazebo plugin are generally applicable to surface vessels. Applying this model to a specific vessel requires determining values for the hydrodynamic derivatives. These values can be approximated from first principles (e.g., potential flow theory) or estimated through experimental testing (e.g., scale model testing). In [25] the authors estimate the hydrodynamic derivatives for a three degree-of-freedom model of the WAM-V vessel by manually tuning the coefficients so that the model outputs agree with experimental measurements from at-sea maneuvering tests. Each of the of linear drag terms in (21) are estimated, but only the surge term in the quadratic drag matrix (22) is identified, implying that the remaining quadratic terms are neglected. These specific values are adopted for modeling the WAM-V within the simulation as part of the VRX challenge use-case.

B. Hydrostatic and Wave Forces

The presence of ocean waves has two important effects in the context of developing autonomy for surface vessels: motion control effects and perception effects. These two effects are captured in the vessel model hydrostatic ($\boldsymbol{g}(\boldsymbol{\eta})$) and wave excitation ($\tau_{waves}$) terms in (16) and by the wave modeling presented in Section II-A. The visual model is implemented as a custom OpenGL shader to render the water surface based on the wave state. The physical model uses the wave surface state at the vessel location to approximate the induced motion.

For the WAM-V catamaran, a discretization of each demi-hull is performed based on the user-supplied grid resolution, N. Algorithm 1 outlines the structure of the Gazebo plugin used to generate the effect of incident waves. On line 11 the wave force is generated based on the position and velocity of the vessel grid point, the velocity at the corresponding location on the wave field surface and the demi-hull geometry. This restoring force is applied to the corresponding vessel location using the Gazebo API to generate the wave induced motion.

Algorithm 1 Wave Forcing

```
 1: pose = GetWorldPose()
 2: vel = GetWorldVelocity()
 3: for $i = 0$ to 2 do                    ▷ For each WAM-V hull.
 4:     for $j = 0$ to $N$ do                ▷ For each grid point.
 5:         $\mathbf{x}$ = GridPosition(pose.position,$i$,$j$)
 6:         $z$ = GridHeight(pose.position,
 7:                 pose.orientation, $i$,$j$)
 8:         $\dot{z}$ = GridVelocity(vel.linear, vel.angular, $i$,$j$)
 9:         $\zeta$ = WaveHeight($\mathbf{x}$,t)
10:         $\dot{\zeta}$ = WaveVelocity($\mathbf{x}$,t)
11:         $f$ = WaveForce($z - \zeta$, $\dot{z} - \dot{\zeta}$)
12:         ApplyForceAtPosition($f$,$\mathbf{x}$)
13:     end for
14: end for
```

This approach to approximating the influence of ocean waves is a simplification intended to balance physical fidelity and visual realism with real-time execution requirements. The result is consistent simulated motion and sensor feedback to exercise the pertinent aspects of autonomy, particularly motion control and sensor-based perception. To preserve real-time execution, the model neglects wave influence factors typically included in applications where the precise motion of a particular vessel or structure design are of critical importance, e.g., studies that include slamming, green water on deck and dynamic stability. A variety of naval architecture design software applications provide methods to assess the affect of vessel design on resulting motion, but these solutions typically run much slower than real-time. Previous work by the authors [26] compared the proposed approach with a commercial naval architecture design software package under similar sea conditions, taking into account model forces and motions to quantify the trade-offs in complexity and fidelity. The results quantified the fidelity differences between the approaches and demonstrated that the proposed method could sufficiently exercise robotic autonomy while executing in real-time.

C. Wind Forces

The model used for simulating the wind environment—consisting of horizontal wind speed (V_w) and direction (β_w)—is described in Section II-B. We consider the influence of this wind on the maneuvering degrees-of-freedom of the model. We neglect the wind-induced motion in heave, pitch and roll. While the influence on heave and pitch are typically very small, for certain vessels, under certain conditions, the wind can have an effect on roll. For such cases the same coefficient-based model presented below can be extended.

We adapt the model and notation described in [22]. The relative (apparent) wind velocities are

$$u_{rw} = u - u_w \tag{23a}$$
$$v_{rw} = v - v_w \tag{23b}$$

where u_w and v_w are the x and y components of the simulated wind velocity in the vessel body frame, expressed as

$$u_w = V_w \cos(\beta_w - \psi) \tag{24a}$$
$$v_w = V_w \sin(\beta_w - \psi). \tag{24b}$$

The surge, sway and yaw components of the wind force vector ($\boldsymbol{\tau}_{wind}$ in (16)) are dependent upon the apparent wind and the coefficients for each mode. For symmetrical vessels, these wind coefficients can be considered constant. Using dimensional wind coefficients $\bar{c}_x$, $\bar{c}_y$ and $\bar{c}_n$ we express the forcing terms as

$$X_{wind} = \bar{c}_x u_{rw} |u_{rw}| \tag{25a}$$
$$Y_{wind} = \bar{c}_y v_{rw} |v_{rw}| \tag{25b}$$
$$N_{wind} = -2.0 \bar{c}_n u_{rw} v_{rw}. \tag{25c}$$

This wind forcing model is implemented as another standalone Gazebo plugin. At runtime the user specifies the wind characteristics (direction, mean speed, standard deviation and time constant) and the vessel-specific wind coefficients in (25). The wind speed and direction at simulation time is calculated according to the wind generation model in (13), the components of the apparent wind are calculated in the vessel body-frame and the resulting forces from (25) are applied to the simulated vessel for inclusion in the next cycle of the physics engine update.

Analogous to the hydrodynamic model, this approach to representing wind-induced forces is generally applicable to surface vessels and a vessel-specific application requires estimation of the wind coefficients. For the WAM-V model, wind coefficients have been estimated based on experimental testing [25]. We use these numerical values for the WAM-V for the purposes of the VRX challenge reference implementation.

D. Propulsion Forces

The propulsion system consists of a set of vessel-mounted thrusters to implement vessel motion control by generating external forces on the vehicle. Design of the propulsion system includes specification of the thruster type, thruster characteristics, and thruster location on the vessel. To accommodate user-defined configurations, we model the force from an individual thruster as a function of a thrust command between -1.0 (full reverse) and 1.0 (full forward). At each time step the resulting external force is included in the model according to (16). Implemented as a Gazebo plugin, this force is applied at the geometric location of the propeller joint as illustrated in Fig. 2. Each thruster can also be articulated by commanding the angle of the thruster relative to the mount, allowing for the possibility of vectored thrust solutions. The motion of the angular articulation is simulated through a proportional-derivative controller to move the thrust vector to the desired location with temporal dynamics representative of typical electromechanical dynamics.

While Fig. 2 shows a typical configuration with two thrusters at the stern of the vessel, users may add multiple thrusters at other locations to reflect other possible propulsion system designs. For example, an additional bow or lateral thruster is often added for additional control authority; alternately, a four-thruster configuration with angled thrusters at the bow and stern of each hull can be implemented when maneuverability is prioritized over speed and efficiency.

The characteristics of an individual thruster are specified by a user-defined relationship between commanded effort and the resulting thrust force. Two static mappings

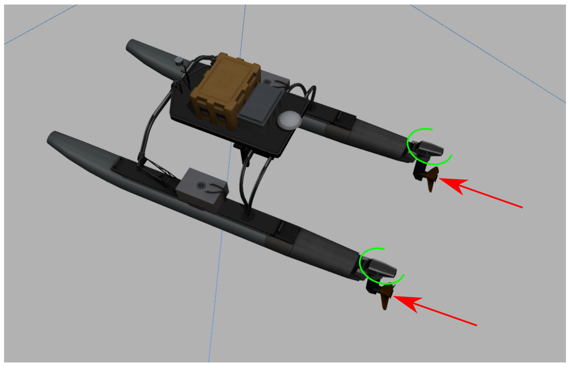

Fig. 2. Illustration of propulsion forces for two-thruster configuration. Two independent forces are applied to the virtual body as indicated by the red arrows. The angle of the each thruster relative the mounting point can also be controlled to allow for vectored thrust, as indicated by the green arcs.

are included, with functionality for custom mappings for more specific cases. The first mapping is a linear mapping where the command values (-1.0 to 1.0) are scaled linearly to the user supplied parameters for maximum forward and reverse thrust values.

The second, non-linear static mapping uses generalized logistic functions (GLFs) to approximate the relationship between command and force often provided as the result of static bollard-pull tests. The GLF is parameterized as

$$T = A + \frac{K - A}{\left(C + \exp(-B(x - M))\right)^{1/\nu}} \qquad (26)$$

where T is the thrust in Newtons; x is the commanded effort; and A, K, C, B, M, and ν are user-defined constants used to specify the GLF. To capture the asymmetry associated with forward and reverse, two independent GLFs are used, one for positive commands (0 to 1.0) and a second for negative commands (-1 to 0). The user-specified functional parameters allow the thrust model used to approximate a wide variety of thruster designs based on empirical data.

To illustrate this process we specify the GLF thrust mapping to approximate the steady-state bollard-pull experiments presented in [25]. To identify the GLF parameters, a Nelder-Mean simplex optimization was performed to minimize the squared error between the model and the GLF expressions (26) as a function of the GLF parameters. The results are shown in Fig. 3.

This approach to simulating propulsion force approximates the steady-state non-linearities to exercise the motion controllers under test. However, it does not address effects such as transient thruster dynamics for

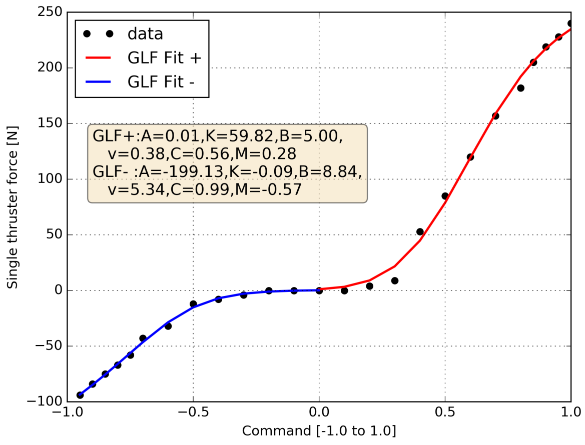

Fig. 3. Generalized logistic function fit of empirical bollard-pull thrust performance data from [25].

slow-speed control [27], varying advance ratios due to forward vessel speed for high-speed operation or changes in thrust during turning due to the changing inflow angle that can generate meaningful side force. Such additions to the propulsion model can be readily implemented, but rely upon the availability of more detailed propulsion characterization.

IV. VRX Reference Implementation

Launched in 2012 with support from the Office of Naval Research, the Maritime RobotX Challenge is hosted biannually by RoboNation to promote research and advancement in autonomous surface vehicles, as well as to encourage student interest and foster relationships within the robotics community. Participating teams compete to design and implement systems capable of accomplishing a variety of tasks ranging from basic navigation and control to more complex operations involving obstacle avoidance, perception and underwater object retrieval.

The purpose of the VRX challenge is to provide competitors with the opportunity to prototype and test software solutions in advance of physical on-water deployments. A major goal of the project is to facilitate improved performance in the physical competition. As such, it implements a suite of models and plugins that closely mirrors the location, course elements, and types of tasks encountered by participants in RobotX 2018. Fig. 4 shows a visual comparison of an example vehicle and elements from RobotX alongside a similar scene rendered in the VRX environment with Gazebo.

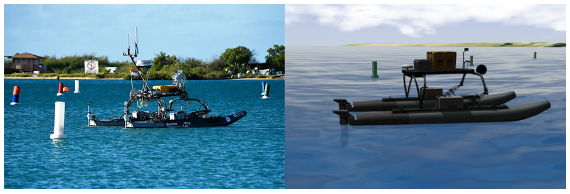

Fig. 4. A visual comparison of a physical WAM-V and course elements from RobotX 2018 (left) with a similar scene rendered in the VRX environment using Gazebo (right).

The first VRX Competition will be held in November 2019. Similar to RobotX, it will be organized around a series of tasks that evaluate autonomous navigation and perception capabilities of participants' solutions. These tasks include station-keeping, waypoint guidance, object localization and characterization, navigation and rules of the road, and docking. The VRX environment also serves as a proof of concept for the theory of operations herein described. Tasks require teams to develop solutions that correctly compensate for the influence of waves and wind when controlling the motion of the vehicle. The station-keeping and path planning tasks, in particular, bring this requirement into relief. In order for these tasks to provide useful evaluations of participant solutions, the VRX environment must succeed in simulating the impact of wind and wave forces with sufficient fidelity and speed. The object localization and characterization task isolates environmental effects on perception, and therefore relies heavily on the availability of synchronized visual and physical models. Finally, the more complex navigation and docking tasks test the ability of the VRX environment to render a real-time simulation that gives a sufficiently faithful approximation of environmental influence on both motion control and perception effects at the same time[2].

Fig. 5 and Fig. 6 demonstrate the combined effect of our environment and vehicle models within the VRX simulation environment. Fig. 5 depicts the base Sand Island world on which the competition is built, including course elements and the WAM-V. The inset graph included in the display shows the wind speed varying over time in accordance with user specified parameters and the model given in Section II-B. Fig. 6 was created from the Dock task plugin using a peak period of 5 seconds and a gain ratio of 0.7. The scene illustrates the effect of the wave field on vehicle motion as well as the disturbances to the perception system produced by the resulting pitch, roll and heave of the vehicle.

[2]A complete description of the VRX 2019 challenge tasks is available at https://bitbucket.org/osrf/vrx/wiki/documentation

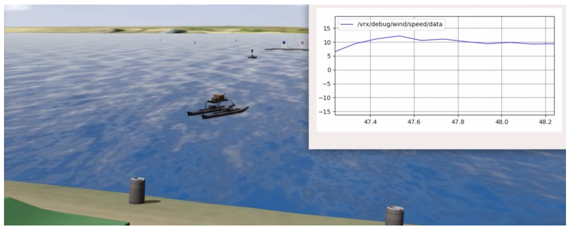

Fig. 5. The VRX Sand Island world rendered in Gazebo with a plot of wind speed over time.

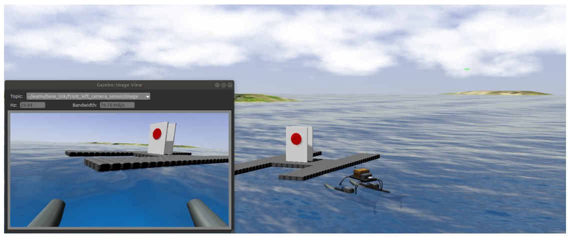

Fig. 6. The VRX Competition Dock task with a non-trivial wave environment ($K = 0.7$, $T_p = 5$). Note the WAM-V is slightly rolled and pitched due to wave forcing, and the effect of this disturbance is reflected in the camera view.

V. Conclusion

In this paper we describe a simulation approach for robotic surface vessels operating in marine environments designed to support rapid development, testing and evaluation of new autonomy solutions. The contributions of this research are a wave model for visual and physical effects based on Gerstner waves and the two-parameter Pierson Moskowitz spectrum, a wind model to represent mean and variable (gust) components based on the Harris spectrum, a parameterized propulsion model, a six degree-of-freedom vessel model to capture environmental and control influence on vessel motion and sensor perception, and a demonstrative reference implementation: the new 2019 VRX challenge robot competition.

The VRX simulation is implemented as a set of Gazebo plugins, object models (visual, collision and rigid body representations), and environmental scenarios (world scenes) released as an open-source project. The simulation is highly parameterized, allowing users to specify the details of the environmental (wave and wind spectra constants), the vessel (hydrodynamic and wind coefficients), the propulsion (thruster location and authority) and the sensors (type and location). This parameterization generalizes the simulation so that it is adaptable to a variety of surface marine autonomy applications.

Acknowledgment

The VRX simulator is an open-source project that welcomes community contributions. We thank all the contributors for their time and energy. A current list of contributors is maintained at https://bitbucket.org/osrf/vrx/wiki/Contributors.

References

[1] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154.

[2] C. E. Agüero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J. L. Rivero, J. Manzo, E. Krotkov, and G. Pratt, "Inside the virtual robotics challenge: Simulating real-time robotic disaster response," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 494–506, April 2015.

[3] K. A. Hambuchen, M. C. Roman, A. Sivak, A. Herblet, N. Koenig, D. Newmyer, and R. Ambrose, "NASA's space robotics challenge: Advancing robotics for future exploration missions," in *AIAA SPACE and Astronautics Forum and Exposition*, 2017.

[4] M. Allan, U. Wong, P. M. Furlong, A. Rogg, S. McMichael, T. Welsh, I. Chen, S. Peters, B. Gerkey, M. Quigley, M. Shirley, M. Deans, H. Cannon, and T. Fong, "Planetary rover simulation for lunar exploration missions," in *2019 IEEE Aerospace Conference*, March 2019, pp. 1–19.

[5] T. Tosik, J. Schwinghammer, M. J. Feldvob, J. P. Jonte, A. Brech, and E. Maehle, "MARS: A simulation environment for marine swarm robotics and environmental monitoring," in *OCEANS 2016 - Shanghai*, April 2016, pp. 1–6.

[6] E. H. Henriksen, I. Schjolberg, and T. B. Gjersvik, "UW MORSE: The underwater modular open robot simulation engine," in *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, Nov 2016, pp. 261–267.

[7] T. Watanabe, G. Neves, R. Cerqueira, T. Trocoli, M. Reis, S. Joyeux, and J. Albiez, "The Rock-Gazebo integration and a real-time AUV simulation," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, Oct 2015, pp. 132–138.

[8] K. J. DeMarco, M. E. West, and A. M. Howard, "A computationally-efficient 2D imaging sonar model for underwater robotics simulations in Gazebo," in *OCEANS 2015 - MTS/IEEE Washington*, Oct 2015, pp. 1–7.

[9] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A Gazebo-based package for underwater intervention and multi-robot simulation," in *OCEANS 2016 MTS/IEEE Monterey*, Sept 2016, pp. 1–8.

[10] D. Cook, A. Vardy, and R. Lewis, "A survey of AUV and robot simulators for multi-vehicle operations," in *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, 2014.

[11] S.-K. Ueng, D. Lin, and C.-H. Liu, "A ship motion simulation system," *Virtual Reality*, vol. 12, no. 1, pp. 65–76, 2008.

[12] D. Sandaruwan, N. Kodikara, C. Keppitiyagama, R. Rosa, M. Jayawardena, and P. Samarasinghe, "User perception of the physical behavioral realism of a maritime virtual reality environment," in *2012 UKSim 14th International Conference on Computer Modelling and Simulation*, March 2012, pp. 172–178.

[13] D. J. Yeo, M. Cha, and D. Mun, "Simulating ship and buoy motions arising from ocean waves in a ship handling simulator," *SIMULATION*, vol. 88, no. 12, pp. 1407–1418, 2012.

[14] S. Thon, J. . Dischler, and D. Ghazanfarpour, "Ocean waves synthesis using a spectrum-based turbulence function," in *Proceedings Computer Graphics International 2000*, June 2000, pp. 65–72.

[15] J. Tessendorf, C. C, and J. Tessendorf, "Simulating ocean water," 1999.

[16] D. Hinsinger, F. Neyret, and M.-P. Cani, "Interactive animation of ocean waves," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '02. New York, NY, USA: ACM, 2002, pp. 161–166.

[17] G. A. Mastin, P. A. Watterberg, and J. F. Mareda, "Fourier synthesis of ocean scenes," *IEEE Computer Graphics and Applications*, vol. 7, no. 3, pp. 16–23, March 1987.

[18] J. Fréchot, "Realistic simulation of ocean surface using wave spectra," in *GRAPP 2006 - Proceedings of the 1st International Conference on Computer Graphics Theory and Applications*, 01 2006.

[19] T. S. C. on Waves, "Final report and recommendation to the 23rd ITTC," in *Proceedings of the 23rd ITTC*, vol. II. International Towing Tank Conference, 2002.

[20] H. Mitsuyasu, F. Tasai, T. Suhara, S. Mizuno, M. Ohkusu, T. Honda, and K. Rikiishi, "Observations of the directional spectrum of ocean wavesusing a cloverleaf buoy," *Journal of Physical Oceanography*, vol. 5, no. 4, pp. 750–760, 1975.

[21] R. I. Harris, "The nature of the wind, the modern design of wind-sensitive structures," in *Construction Industry Research and Information Association*, 1971, pp. 29–55.

[22] T. I. Fossen, *Guidance and Control of Ocean Vehicles*. Wiley, 1994.

[23] ——, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[24] P. Krishnamurthy, F. Khorrami, and S. Fujikawa, "A modeling framework for six degree-of-freedom control of unmanned sea surface vehicles," Dec. 2005, pp. 2676–2681.

[25] E. I. Sarda, H. Qu, I. R. Bertaska, and K. D. von Ellenrieder, "Station-keeping control of an unmanned surface vehicle exposed to current and wind disturbances," *Ocean Engineering*, vol. 127, pp. 305 – 324, 2016.

[26] J. F. Malia, "Modeling of a dynamic wave environment for unmanned surface vessel control," Master's thesis, Naval Postgraduate School, 2018.

[27] L. L. Whitcomb and D. R. Yoerger, "Development, comparison, and preliminary experimental validation of nonlinear dynamic thruster models," *IEEE Journal of Oceanic Engineering*, vol. 24, no. 4, pp. 481–494, Oct 1999.