Kelpie: A ROS-Based Multi-Robot Simulator for Water Surface and Aerial Vehicles

Ricardo Mendonça[1], Pedro Santana[1,2], Francisco Marques[1], André Lourenço[1], João Silva[1], José Barata[1]
[1]CTS-UNINOVA, New University of Lisbon, Portugal
[2]Instituto Universitário de Lisboa (ISCTE-IUL), Portugal
{r.mendonca, facmarques, p110251}@campus.fct.unl.pt, pedro.santana@iscte.pt, jab@uninova.pt

***Abstract*—Testing and debugging real hardware is a time consuming task, in particular for the case of aquatic robots, for which it is necessary to transport and deploy the robots on the water. Performing waterborne and airborne field experiments with expensive hardware embedded in not yet fully functional prototypes is a highly risky endeavour. In this sense, physics-based 3D simulators are key for a fast paced and affordable development of such robotic systems. This paper contributes with a modular, open-source, and soon to be freely online available, ROS-based multi-robot simulator specially focused for aerial and water surface vehicles. This simulator is being developed as part of the RIVERWATCH experiment in the ECHORD european FP7 project. This experiment aims at demonstrating a multi-robot system for remote monitoring of riverine environments.**

***Index Terms*—Physics-Based Simulation, Multi-Robots, Unmanned Surface Vehicles, Unmanned Aerial Vehicles, ROS**

I. Introduction

Field robotics is rapidly expanding into the aerial [15], [19], [22] and the aquatic [1], [3], [5], [11], [17], [23] domains. A great deal of the current success of Unmanned Aerial Vehicles (UAV) comes from the adoption of multi-rotor configurations, which allow vertical take-off and landing, high stability and maneuverability. These characteristics are considerably useful for tasks like environmental monitoring, wild life tracking, and search & rescue missions. The limited energetic autonomy of multi-robot UAVs makes them of limited reach in the aquatic domain. Conversely, Unmanned Surface Vehicles (USV) are able to provide long lasting operation in the aquatic domain (refer to [2] for a survey on USVs). This complementary nature of UAVs and USVs can be exploited in the context of heterogeneous robots teamwork [12], [16], [18]

Although appealing, field robotics, in particular aquatic and aerial, suffers from a rather challenging development process, mostly due to the remoteness and harshness of the environments in which these robots are to operate. In addition to the involving and time consuming process of taking the hardware out to the field, there is also the risk of damaging expensive equipment throughout the debugging process. The use simulation environments is key to mitigate these problems. Moreover, simulators allow testing and tuning the system to perform in exceptional situations, i.e., situations that are barely replicated in actual field trials. Testing in exceptional situations means checking how the system responds to hardware failures, power shortage, or extremely odd environment configurations.

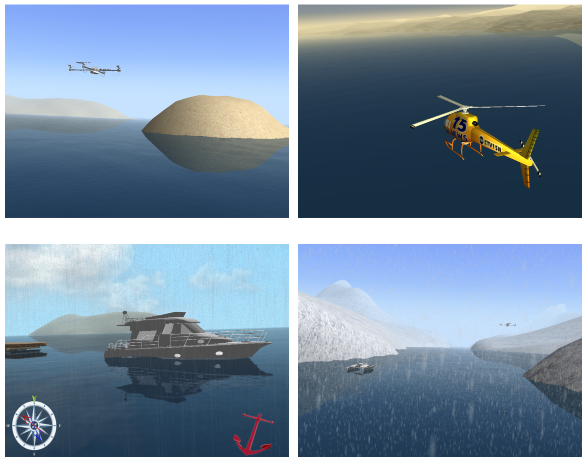
Fig. 1. Snapshots of some environments simulated with Kelpie.

Another key advantage of simulations in multi-robot systems is the ability to assess the scalability and robustness of the system in the face of varying team members cardinality.

Simulation tools for USV-UAV teams require physics-based 3D modelling, advanced rendering capabilities, real-time capabilities, flexibility, ability to running on a set of distributed computational units, and seamless integration with the robot control system. The last requirement grants a rapidly and interchangeably linking between the control system and either the simulated or the physical devices (e.g., sensors, actuators). To our knowledge, no previous simulator has managed to cope with all these requirements simultaneously. Conversely, this paper presents Kelpie, a novel open source, soon to be online, multi-robot simulator capable of coping with all these requirements (see Fig. 1). Kelpie is being developed for the RIVERWATCH experiment in the ECHORD[1] european FP7 project. This experiment aims at demonstrating a multi-robot system for remote monitoring of riverine environments.

This paper is organised as follows. Section II surveys previous simulators for aerial and aquatic robotics. Then, Section III describes the Kelpie simulator architecture. Several application cases for the simulator are presented in Section IV. Finally, some conclusions and future work avenues are drawn in Section V.

[1]ECHORD homepage: http://www.echord.info/

II. Related Work

Most simulators used as research tools perform numerical simulation and generate visualisation data mostly in the form of 2D or 3D plots (e.g., Simulink®). While this kind of simulators are useful to support the design and development of low-level control architectures, they fail to scale towards 3D physics-based simulation with photo-realistic rendering capabilities.

Good rendering capabilities are essential to enable the simulation of vision and other perceptual stimuli upon which perceptual algorithms can be reasonably debugged and validated. As a result, simulators based on advanced 3D computer graphics engines are preferable to support the development of high-level functionalities in robotic systems, such as: object detection and tracking, environmental exploration and interaction, path planning and obstacle avoidance. An emerging trend in this direction is to build simulators from existing rendering engines, such as game engines [7].

Previous simulators for aquatic robots are split into two main categories: underwater robots and surface robots. The vast majority of simulators, such as the open source SubSim [4], Neptune [20], IGW [6], MVS [10], and the commercial DeepWorks [8], lies on the underwater category. These simulators include hydrodynamic models and implement some useful sensors, such as underwater sonars and vision cameras. As a representative of surface robot simulators there is WaveSim [21], which is also able to simulate underwater vehicles. A limitation of WaveSim is that it relies on basic geometric primitives as approximations to represent vehicles and object models in the physics engine. For instance, surface vehicles are represented as boxes and underwater robots as cylinders.

Kelpie, the simulator herein presented shares some of the concepts underlying WaveSim and goes beyond in several directions. First, Kelpie is fully compliant with the Robot Operating System (ROS), which is a free and open source software framework that is becoming a *de facto* standard in robotics. ROS provides standard operating system services such as hardware abstraction, low-level device control, message-passing between processes and commonly used functionalities, enabling a distributed computing development framework. Second, Kelpie provides more accurate physics simulation and rendering quality by using the geometric meshes of the vehicle models instead of simple geometric shapes. Finally, Kelpie also allows the simulation of UAVs.

Kelpie's core architecture is based on Gazebo [9], a robotics simulator distributed with ROS. However, Kelpie distinguishes from Gazebo by supporting the rendering of water regions and the simulation of vessel dynamics (e.g., drag forces caused by water resistance and buoyancy), as well as, flight dynamics for airborne vehicles. Note that Kelpie is not necessarily constrained to these two type of vehicles, as it also supports the simulation of terrestrial and underwater robots.

Please refer to [7] for a survey on unmanned vehicles simulators and to [14] for the specific case of underwater robotics.

III. The Kelpie Simulator

The Kelpie simulator is able to account for several aerial, surface, underwater, and ground virtual robots and their typical sensors. In Kelpie, these robots are simulated according to the laws of physics and immersed on high quality rendered 3D heterogeneous environments (e.g., water and terrain regions). To ensure full integration with contemporary autonomous robots control systems, Kelpie is fully compliant with ROS. Each of these properties of Kelpie is dissected next.

A. System Architecture

In the ROS framework, nodes interact to each other through a publish-subscribe messaging and service mechanism. Messages are associated to topics, which are containers with unique identifiers to and from messages are push and pulled. In a ROS network, Kelpie is just a node with reconfigurable set of interfaces, i.e., topics. Hence, several other ROS nodes are able to interact with Kelpie, i.e., virtual robots actuators and sensors, as if they would be interacting with the actual robots. Other ROS nodes may be collecting data from the simulation and storing them to offline analysis or setting simulation control parameters, such as those required to change time, season, or weather.

Fig. 2 illustrates the major components of the ROS node responsible for Kelpie: a ROS interface, a computer graphics renderer, a physics engine, and a XML parser.

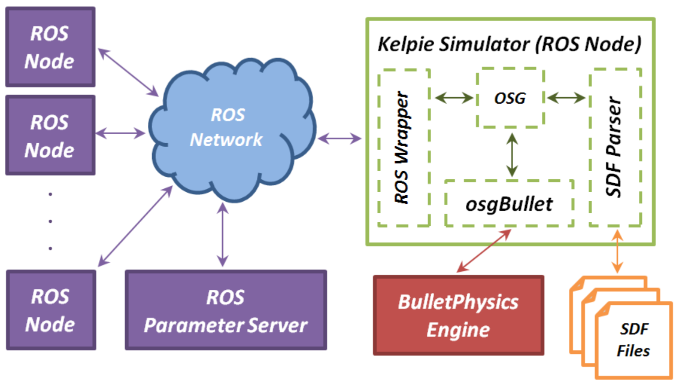

Fig. 2. System architecture.

The ROS client interface provides access to core functionalities and implements the concept of topic-based communications, providing the compliance needed to integrate the simulator with the ROS framework. For instance, the simulator uses the ROS client interface to store and make publicly available some simulation parameters (e.g., direction and speed of water currents) in the ROS system-wise parameter shared server. These simulation parameters can then be modified at run-time by other ROS nodes.

Kelpie also exploits the ROS diagnostic toolchain, via its client interface, to ease the development of diagnostic nodes for robotic systems. This toolchain provides the means for collecting, publishing, troubleshooting and viewing diagnostic-related data. For instance, Kelpie can publish diagnostic messages related to several robot model's parameters (e.g., battery level and actuator's temperature).

Real-time computer graphics rendering is managed by OpenSceneGraph (OSG)[2], which is an open source graphics toolkit based on the scene graph concept and it is written in standard C++ and OpenGL[3]. Furthermore, the open source osgOcean library[4] is used to generate the above and below water rendering effects, as depicted in Fig. 3.

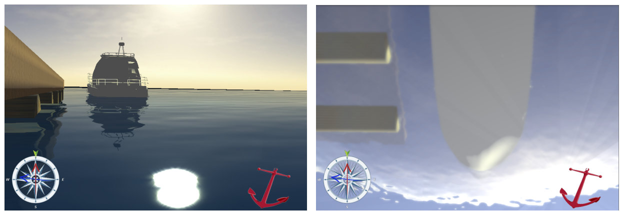
Fig. 3. Water effects in Kelpie, as seen from above (left) and from below (right) the surface.

The physics simulation is supported by the BulletPhysics library[5], which is an open source physics engine featuring 3D discrete and continuous collision detection with diverse collision shapes, as well as soft and rigid body dynamics. To integrate the BulletPhysics engine into OSG-based models, the osgBullet[6] library is used. Concretely, this library maps the state of a BulletPhysics dynamical model to the parameters of the corresponding OSG geometric model of the object, given external stimuli (e.g., forces and torques).

Finally, a XML parser is used to save and load Simulation Description Format (SDF) files describing the scene, the virtual robots and their sensors. This gives the ability for Kelpie to import existing Gazebo SDF world files.

B. Buoyancy Simulation

Kelpie uses the Archimedes' principle to simulate buoyancy in water. Briefly, this law of physics states that a fluid exerts an upward force (i.e., buoyant force) that equals the weight of the fluid displaced by an immersed body. For instance, the surface vehicle depicted in Fig. 4 suffers a buoyant force $\mathbf{F}_b$ from the surrounding water. This force is directed upward and has a magnitude equal to the weight of the water displaced by the submerged hull. Formally, the magnitude of the buoyant force vector $\mathbf{F}_b$ is given by

$$||\mathbf{F}_b|| = \rho \cdot v_f \cdot g, \tag{1}$$

where g is the gravitational acceleration constant, ρ is the water's density and v is the volume of the water displaced.

When the surface vehicle is floating, the buoyant force, $\mathbf{F}_b$, and the gravitational force, $\mathbf{F}_g$, share the same magnitude, though opposite directions (see Fig. 4). If the buoyant force's magnitude gets lower than the magnitude of the gravitational

[2]OSG homepage: http://www.openscenegraph.org

[3]OpenGL homepage: http://www.opengl.org

[4]osgOcean homepage: code.google.com/p/osgocean/

[5]BulletPhysics homepage: http://bulletphysics.org/wordpress/

[6]osgBullet homepage: osgbullet.googlecode.com/

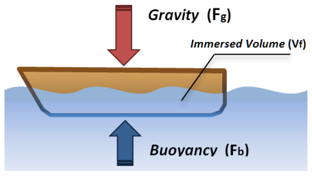

Fig. 4. Buoyancy simulation.

force, then the sinking occurs. Hence, to assess buoyancy, the simulator needs to compute the buoyant and the gravitational forces that the physics engine must exert on the vehicle at each time step. The gravity force is easy to compute as it only requires the vehicle's mass to be known, which is given a priori. The buoyant force is computed with Eq. 1 given the water's density, which is also known a priori, and the volume of the displaced water, v_f, which needs to be computed at each simulation step. The volume of displaced water corresponds to the volume of the vehicle's immersed region, which is hard to determine exactly. Currently, this quantity is approximated by the volume of a bounding box.

C. Wind and Water Currents

The vehicle's motion and attitude on the water's surface is a function of its propellers actuation and forces caused by external factors, such as winds, water currents, and waves. In its current version, Kelpie does not consider sailed vehicles. Thus, wind forces are only considered indirectly via changes applied to the amplitude and frequency of water waves. For instance, if a surface vehicle receives lateral winds it will suffer an oscillating roll torque (see Fig. 5), whereas a frontal wind will cause an oscillating pitch torque.

Formally, Kelpie approximates the water-wind-vehicle interaction dynamics by first computing the amplitude of waves as a function of time t and wind amplitude ν, as follows:

$$h(\nu, t) = h_m(\nu) \cdot \sin(\, \omega(\nu)\, t\,), \tag{2}$$

where $h_m(\nu)$ and $\omega(\nu)$ are the maximum amplitude of the wave and its frequency, respectively, given the wind speed ν. These two function were implemented according to the Beaufort scale, which relates wind speed to observed conditions at sea and land. This approximation disregards the complex interactions among waves of different frequencies and amplitudes, which are key for photorealistic simulation but of little value to the purpose of debugging perceptual and control algorithms of unmanned vehicles. The surface vehicle's pitch and roll angles, ϕ_{pitch} and ϕ_{roll}, are then modified as follows:

$$\phi_{pitch}(\mathbf{p}_m^y, \mathbf{p}_w, \nu, t) = (\mathbf{p}_m^y \cdot \mathbf{p}_w)\, h(\nu, t)\ \ \alpha_{pitch}; \tag{3}$$

$$\phi_{roll}(\mathbf{p}_m^x, \mathbf{p}_w, \nu, t) = (\mathbf{p}_m^x \cdot \mathbf{p}_w)\, h(\nu, t)\ \ \alpha_{roll}; \tag{4}$$

where the inner product is used to verify the alignment between the vehicle's directional vectors, $\mathbf{p}_m^x$ and $\mathbf{p}_m^y$ (see Fig. 6(a)), and the wind directional vector, $\mathbf{p}_w$. For instance, if both vectors are strictly parallel, the inner product returns 1.0 or

-1.0, either they yield the same direction or not, respectively. If both vector are perpendicular, it returns zero. Otherwise the returned value will range between zero and one. This allows the surface vehicle's attitude to change in its pitch and roll axis according to the vehicle's orientation in relation to the travel direction of the water waves along the time. The proportional factors, α_{pitch} and α_{roll}, are defined by the user according to the vehicle's kinodynamic properties. For instance, if the vehicle is very lightweight its attitude can change in the full range of the wave's amplitude. Otherwise, if the surface vehicle is heavy weighted then its attitude changes in a narrow range of water wave amplitude.

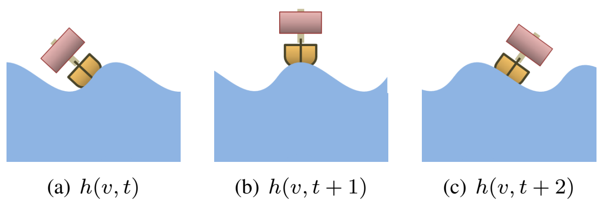
(a) $h(v,t)$ (b) $h(v,t+1)$ (c) $h(v,t+2)$

Fig. 5. Roll torque applied by wind to a surface vehicle.

Concerning water currents, they can exert linear and rotational forces (e.g., yaw torques) on the surface vehicle, depending on its orientation relative them. Formally, the yaw torque, τ, applied to the surface vehicle, given the current's force applied to the vehicle's keel, $\mathbf{F_c}$, is $\tau = r \times \mathbf{F_c}$, where r is the displacement vector from the point from which torque is measured (vehicle's center of mass) to the point where the force is applied (see Fig. 6(b)).

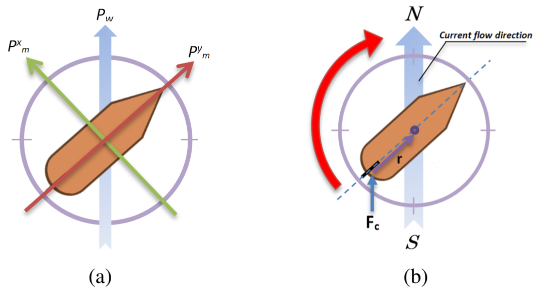

Fig. 6. (a) 2D directional vectors of the model, $\mathbf{p}_m^x$ and $\mathbf{p}_w^y$, and hypothetical wind vector $\mathbf{p}_w$ (b) Effect of water current on vehicle's yaw rotation (represented by red arrow).

Aerial robots are also affected by wind forces, but in a differently way compared to aquatic models. Aerial models do not display an oscillating behaviour. Instead, a steady force with magnitude and direction of wind is applied to them. Depending on the direction of the aerial vehicle's motion, this force can behave as a thrust/drag force F_y and as a lateral force F_x, as follows:

$$F_y\left(\mathbf{p}_m^y, \mathbf{p}_w\right) = (\mathbf{p}_m^y \cdot \mathbf{p}_w)\|\mathbf{p}_w\|; \tag{5}$$

$$F_x\left(\mathbf{p}_m^x, \mathbf{p}_w\right) = (\mathbf{p}_m^x \cdot \mathbf{p}_w)\|\mathbf{p}_w\|; \tag{6}$$

where $\|\mathbf{p}_w\|$ is the magnitude of the wind force. Hence, if both vectors are strictly parallel and share the same direction (positive signal from the inner product), the aerial vehicle will suffer a thrust force. Otherwise it will feel a drag force. For the time being only this simple physics is implemented in the simulator.

D. Simulated Sensory Data

Kelpie provides several sensor models, such as vision cameras, underwater sonars, tilting laser scanners, inertial measurement units (IMU) and global position systems (GPS). These models generate sensory data in a compliant format ready to be published in the ROS network (e.g., *sensor_msgs::Imu, sensor_msgs::LaserScan, sensor_msgs::NavSatFix,* etc.). Therefore, changing from a simulated sensor to its real counterpart is a seamless operation. To account for sensor noise (excluding vision cameras), only a Gaussian model is currently being used. Several ROS nodes can then access and process these data in the context of high-level functionalities (e.g., surface obstacle avoidance from laser scanners, bathymetric from underwater sonars, navigation from positioning and attitude sensors). Fig. 7 illustrates the output of the tilting underwater sonar currently available in Kelpie. Sonar and laser beams are simulated using the ray casting and ray-geometry collision detection techniques available through OSG.

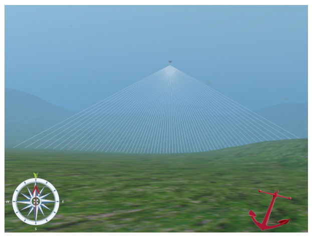

Fig. 7. Simulated underwater sonar beams.

IV. Application Case

As mentioned, Kelpie is being developed for the RIVERWATCH experiment in the ECHORD european FP7 project. This experiment aims at contributing with a multi-robot solution for the remote monitoring of riverine environments (e.g., assessing the water quality, find pollution sources, and tracking underwater biodiversity). Concretely, one of robots is a an Unmanned Surface Vehicle (USV), whereas the other is an Unmanned Aerial Vehicle (UAV), a hexacopter. These innovative multi-rotor flying platforms have significant advantages over other types of aerial robots, namely capability of vertical take-off and landing, high stability and manoeuvrability.

The two robots have complementary properties which can only be fully exploited if coordinated. While the USV is able to assess the water body, the UAV is able to visually inspect the riverbanks. While the USV is able to transport a solar panel to perform energy harvesting, the UAV is not and so must recharge itself by docking in the USV. The higher speed of the UAV means that the USV can move downriver while the former is performing its associated inspection while its energy supply allows it. Furthermore, the UAV can help the USV on path planning by providing a better vantage point with its aerial perspective. To speed up the analysis, these robotic pairs can be multiplied, which in turn requires their coordination at both USV and UAV levels.

To show the usefulness of Kelpie in the context of developing and debugging a project as RIVERWATCH, the following sections present three application cases, namely, obstacle detection and avoidance, UAV-USV cooperative environment perception.

A. *Obstacle Detection and Avoidance*

The short-range obstacle detection in the USV rely on 3-D data provided by a stereo vision head, a tilting laser scanner, and an underwater tilting sonar. The data produced by these sensors is integrated on a volumetric map, from which a bi-dimensional cost map is produced. Cost is high in the presence of objects on water's surface and in the presence of low depth bathymetry.

Fig. 8 illustrates the volumetric map generated with Kelpie's simulated tilting laser scanner. These data were gathered while the simulated USV was moving on the water's surface. The registration of range data in a common volumetric data is done using the Kelpie's simulated GPS and IMU filtered with an Extended Kalman Filter (EKF).

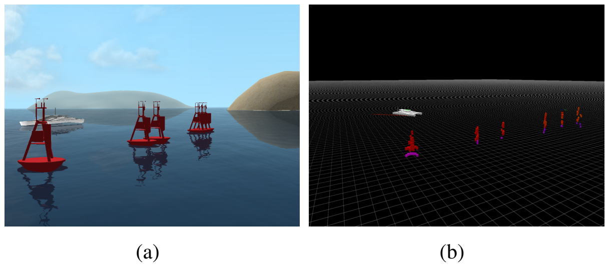

Fig. 8. Set of virtual buoys in (a) and corresponding volumetric map in (b).

Fig. 9 depicts the successful autonomous behaviour of the USV in an environment composed of natural obstacles when asked to move towards a given waypoint. This time the cost map is fed by the tilting laser scanner operating above the water surface. To navigate safely, the USV uses a path planner on the top of a local obstacle avoidance algorithm, both running on the cost map. Again, the localisation of the USV is estimated by filtering the GPS and IMU simulated sensors with an EKF.

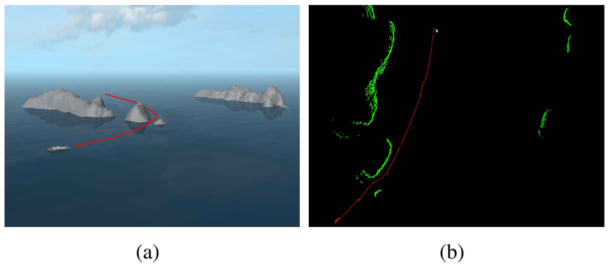

Fig. 9. Laser-based autonomous navigation among natural obstacles towards a given waypoint. (a) Perspective over the simulated environment. (b) Cost map built online with the tilting laser scanner. Green points correspond to mapped obstacle points. The red lines in (a) and (b) correspond to the path executed by the USV from its initial position to the goal waypoint.

B. *USV-UAV Cooperation*

To enable robust safe navigation in aquatic environments, long-range water/land segmentation of the environment is essential, which is considerably hampered by the low vantage point of the USV. Alternatively, we can use the UAV high vantage point to extend the perceptual range of the USV and, consequently, enable truly long-range obstacle detection. For instance, compare the perspective of the environment as seen slightly above the water level and as seen from a considerably high vantage point in Fig. 8(a) and Fig. 10, respectively. These figures are representative of both USV and UAV perspectives over the environment. This USV-UAV cooperation facilitates considerably the hard problem of detecting sand banks and distant shorelines, which are key for safe navigation and difficult to detect from the USV's perspective.

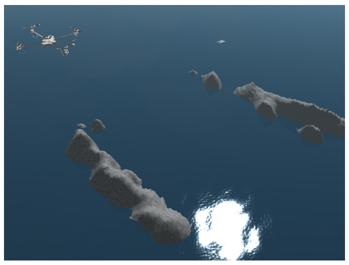

Fig. 10. Aerial perspective of the environment. The simulated UAV is represented in the upper-left corner of the figure.

A key element in this cooperative process is the ability for the UAV to register the images acquired with its downwards looking camera in the USV's frame of reference. One solution to the problem is to detect the USV in the images themselves. Fig. 11 presents a set of images acquired from the simulated UAV's onboard camera while approaching a simulated USV. These images feed a ROS node responsible for actually detecting the USV based on a computer vision algorithm tuned to detect a H-shaped helipad (see Fig. 11(d)). This experiment relies on the set of ROS nodes abstracting both USV and UAV and on ROS messages for their interactions. These protocols are exactly the same as the ones used in the real robotic platforms, thus enabling a seamless transition from simulated to real world experiments.

V. Conclusions and Future Work

A ROS-based multi-robot simulator, entitled Kelpie, specially tuned to enable fast development and debugging of systems composed of cooperating unmanned surface and aerial vehicles was presented. A set of practical cases in remote environmental monitoring were provided in order to demonstrate the usefulness and reach of the simulator. These cases were extracted from the requirements of the RIVERWATCH experiment included in the EU-FP7 ECHORD project.

Kelpie being freely available and open sourced, is expected to contribute towards affordable development of field robotics, in particular in the context of environmental monitoring in aquatic environments. Kelpie is deeply inspired by Gazebo simulator, through standing out from it by being able to render

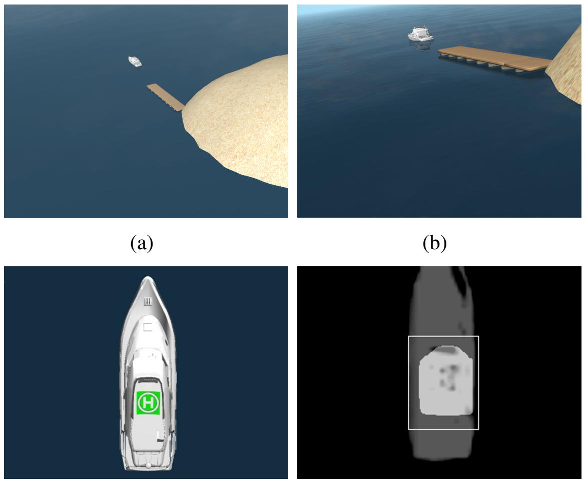

Fig. 11. (a)-(c) Images acquired from the simulated UAV's onboard camera as it approaches the USV. (d) Output of the USV detection and tracking ROS node, given the input image in (c).

marine environments and providing context-related dynamics, such as buoyancy and feedback from environmental forces (e.g., winds and water currents).

Kelpie is under active development. Bugs are being fixed and improved functionality being included. For instance, Kelpie is being extended to include day-night cycles and dynamic weather changes, which will help assessing the robustness of the control systems in extreme situations (e.g., assess the robot's behaviour in context when vision-based obstacle detection is hampered due to the presence of heavy fog).

An interesting extension would be to include improved wind models so that the simulation of sailing vessels would be enabled, as well as the effects caused by wind in the appearance of water waves. In particular, the addition of visual effects in the water (e.g., density of white foam crests and airborne spray) according to the Beaufort scale.

To increase the simulator efficiency in the simulation of marine environments, a migration from the osgOcean library to the faster and novel rendering method based on Graphics Processing Units (GPUs) proposed in [13] is planned.

Finally, improvements to the graphical user interface will be carried out. The interface will provide several features to the user, such as: drag and drop mechanism for on-line materialisation of ready-to-use models (e.g., deployment of buoys into the virtual environment as aquatic obstacles); point-and-click way-point creation for navigation; mouse-based selection of models for easy on-screen access of internal variables; and, notifications of simulation events using the heads-up display (HUD) system (e.g., successful docking procedure, collision detected, etc.). This opens the door to Kelpie as front-end of multi-robot teleoperation and diagnostic systems.

ACKNOWLEDGMENTS

This work was developed within the scope of the RIVERWATCH experiment from the European FP7 Large Scale Integrating Project ECHORD and it was also partially supported by CTS multi-annual funding, through the PIDDAC Program funds.

REFERENCES

[1] Arrichiello, F., et al., "Cooperative caging using autonomous aquatic surface vehicles". In *Proc. of Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, 2010.
[2] Bertram, V., "Unmanned surface vehicles a survey", Skibsteknisk Selskab, Copenhagen, Denmark, 2008.
[3] Bhattacharya, S., et al., "Cooperative control of autonomous surface vehicles for oil skimming and cleanup". In *Proc. of Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, 2011.
[4] Boeing, A. and Brunl, T., "SubSim: An autonomous underwater vehicle simulation package". In *Proc. of the Intl. Symp. on Autonomous Minirobots for Research and Edutainment*, Springer, 2006.
[5] Caccia, M., Bono, R., Bruzzone, G., Spirandelli, E., Veruggio, G., Stortini, A., and Capodaglio, G., "Sampling sea surface with SESAMO". *IEEE Robotics and Automation Magazine*, vol. 12, no. 3, pp. 95-105, 2005.
[6] Choi, S., Yuh, J., Takashige, G. Y., "Development of the Omni Directional Intelligent Navigator". *Robotics and Automation Magazine IEEE*, vol. 2, no. 1, pp. 44-53, 1995.
[7] Craighead, J., et al., "A survey of commercial and open source unmanned vehicle simulators". In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, 2007.
[8] "DeepWorks" [Online]. Available: http://www.fugrogrl.com/software/
[9] Koenig, N. and Howard, A., "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In *Proc. of the Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149-2154, 2004.
[10] Kuroda, Y., Ura, T., Aramaki, K., "Vehicle control architecture for operating multiple vehicles". In *Proc. of the Symp. on Autonomous Underwater Vehicle Technology*, pp.323-329, 1994.
[11] Leedekerken, J., Fallon, M., and Leonard, J., "Mapping complex marine environments with autonomous surface craft". In *Proc. of the Intl. Symp. on Experimental Robotics (ISER)*, 2010.
[12] Lindemuth, M., Murphy, R., Steimle, E., Armitage, W., Dreger, K., Elliot, T., Hall, M., Kalyadin, D., Kramer, J., Palankar, M., Pratt, K., Griffin, C., "Sea Robot-Assisted Inspection". *IEEE Robotics and Automation Magazine*, vol. 18, no. 2, pp. 96-107, 2011.
[13] Liu, S., Xiong, Y., "Fast and stable simulation of virtual water scenes with interactions". *Virtual Reality*, vol. 17, no. 1, pp. 77-88, Springer-Verlag, 2013.
[14] Matsebe, O., Kumile, C., and Tlale, N., "A review of virtual simulators for autonomous underwater vehicles (AUVs)". NGCUV, Killaloe, Ireland, 2008.
[15] Merino, L., et al., "Cooperative fire detection using unmanned aerial vehicles". In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, 2005.
[16] Murphy, R., et al., "Cooperative use of unmanned sea surface and micro aerial vehicles at Hurricane Wilma". *Journal of Field Robotics*, 25(3), pp. 164-180, 2008.
[17] Pascoal, A., Oliveira, P., Silvestre, C., Bjerrum, A., Ishoy, A., Pignon, J., Ayela, G., and Petzelt, C., "MARIUS: an autonomous underwater vehicle for coastal oceanography". *IEEE Robotics and Automation Magazine*, pp. 46-59, 1997.
[18] Pinto, E., Santana, P., Barata, J., "On Collaborative Aerial and Surface Robots for Environmental Monitoring of Water Bodies". In *Technological Innovation for the Internet of Things*, 183–191, Springer-Verlag, 2013.
[19] Rafi, F., et al., "Autonomous target following by unmanned aerial vehicles". *Defense and Security Symposium*, International Society for Optics and Photonics, 2006.
[20] Ridao, P., et al., "NEPTUNE: A HIL simulator for multiple UUVs". In *Proc. of OCEANS*, vol. 1, pp. 524-531, 2004.
[21] Santos, A., "WaVeSim-Ambiente de simulação para veículos aquáticos", *Phd Dissertation*, Universidade do Porto, 2006.
[22] Spry, S., Girard, A., and Hedrick, J., "Convoy protection using multiple unmanned aerial vehicles: organization and coordination". In *Proc. of the American Control Conference*, IEEE, 2005.
[23] Thomas, P. and Djapic, V., "Improving autonomy and control of autonomous surface vehicles in port protection and mine countermeasure scenarios". *Journal of Field Robotics*, 27(6), pp. 903-914, 2010.