IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 4, 2021, accepted March 16, 2021, date of publication March 25, 2021, date of current version April 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068769

A Review of Physics Simulators for Robotic Applications

JACK COLLINS[1,2], SHELVIN CHAND[1], ANTHONY VANDERKOP[1,2], AND DAVID HOWARD[1], (Member, IEEE)

[1]Cyber-Physical Systems Program, CSIRO, Brisbane, QLD 4069, Australia
[2]Centre for Robotics, Queensland University of Technology, Brisbane, QLD 4000, Australia

Corresponding author: David Howard (david.howard@csiro.au)

The work of Jack Collins and Anthony Vanderkop was supported in part by the Data61 Ph.D. Scholarships and in part by the Queensland University of Technology (QUT) through the Centre for Robotics. The work of Shelvin Chand was supported by the CSIRO's CERC Postdoctoral Program.

ABSTRACT The use of simulators in robotics research is widespread, underpinning the majority of recent advances in the field. There are now more options available to researchers than ever before, however navigating through the plethora of choices in search of the right simulator is often non-trivial. Depending on the field of research and the scenario to be simulated there will often be a range of suitable physics simulators from which it is difficult to ascertain the most relevant one. We have compiled a broad review of physics simulators for use within the major fields of robotics research. More specifically, we navigate through key sub-domains and discuss the features, benefits, applications and use-cases of the different simulators categorised by the respective research communities. Our review provides an extensive index of the leading physics simulators applicable to robotics researchers and aims to assist them in choosing the best simulator for their use case.

INDEX TERMS Simulation, review, robotics, field robotics, soft robotics, aerial robotics, marine robotics, manipulation, robotic learning, surgical robotics.

I. INTRODUCTION

Physics simulators enable the vast majority of robotics research. It is commonplace to test and prove theoretical methods initially or solely in a simulator as robots themselves are oftentimes expensive, fragile and scarce. Physics simulators overcome these issues as they provide an environment that is cheap and allows users access to a variety of desired robots without the potential to degrade or break the physical platform. Simulation can run faster than real time (which is especially important for learning-based approaches), is parallelisable, and does not need to be physically tended for an environment to be reset.

The landscape of commercial and open source simulation options for researchers is in a state of perpetual flux, as new simulators are added to support the latest research trends, while others are deprecated. It is naturally difficult, therefore, to select an appropriate simulator for a given robotics project, and the literature to date reports an absence of any sort of comprehensive guide to help researchers find the right simulator for their specific purpose. Existing literature has a number of studies that compare the replication of real-world physical accuracy and constraint satisfaction of physics engines, however, these studies often only compare a small subset of simulators and are usually focused on specialised tasks or sub-domains without consideration for robotics as a whole [1]. Many studies summarise information which can easily be gathered from simulator websites and forums without adding significant value from a research perspective [2]. The capabilities of these simulators in a variety of robotics-specific tasks is not explored or compared against one another.

Through this review we aim to fill the research gaps highlighted above. We focus on seven sub-domains which together capture the majority of robotics work involving simulators. In each sub-domain we discuss current challenges in the field, highlight the necessary capabilities of robotics simulators used in the field, and describe existing approaches to robot simulation in the literature of the sub-domain. Finally, we provide a summarising table in each section which details the capability of numerous robotics simulators in features we have identified to be relevant to robotics simulation in the sub-domain.

We define a robotics simulator as an end-user software application that includes at least the following functionality: (i) physics engine for realistic modelling of physical

The associate editor coordinating the review of this manuscript and approving it for publication was Guilin Yang.

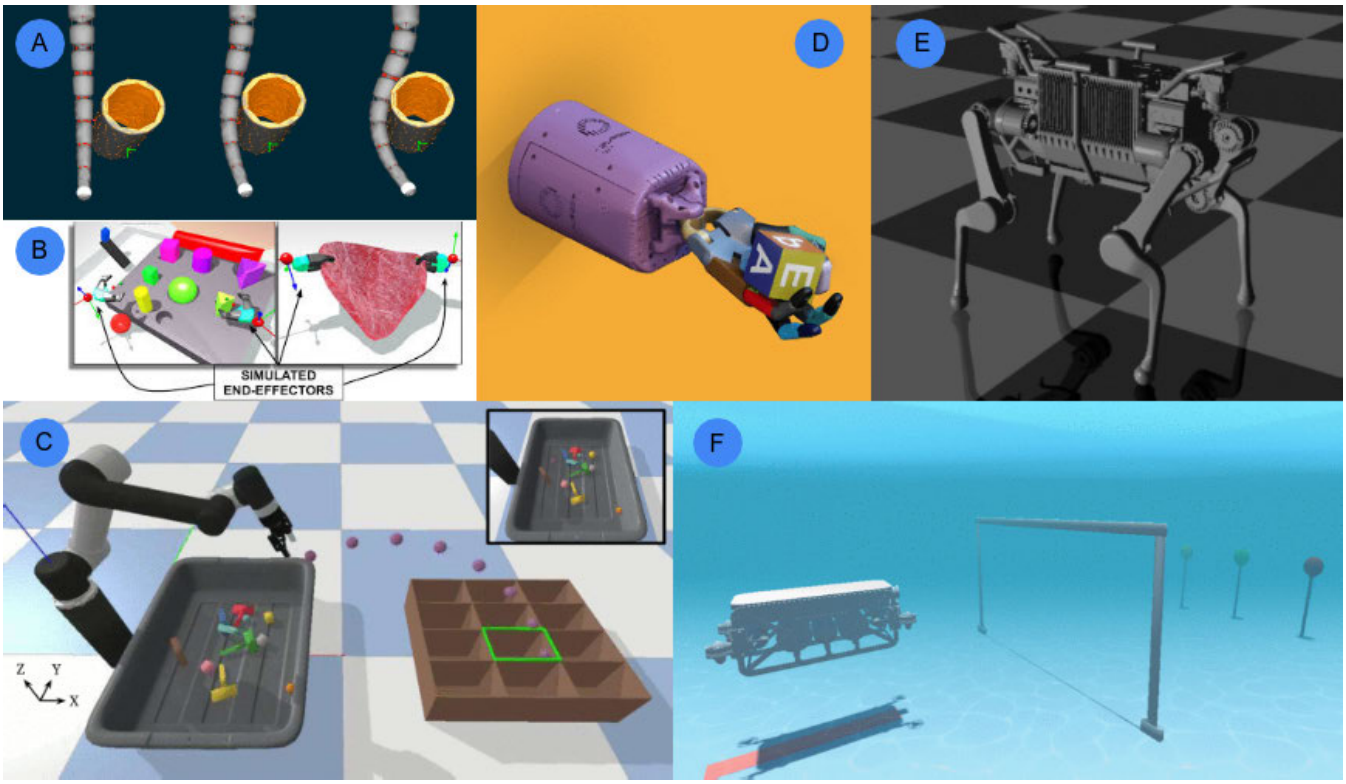

FIGURE 1. Diversity of simulation scenes and environments throughout robotics (a) soft robotics in Simulation Open Framework Architecture [3], (b) medical robotics in Asynchronous Multi-Body Framework [4], (c) manipulation in PyBullet [5], (d) dexterous manipulation in MuJoCo [6], (e) legged locomotion in RaiSim [7] and (f) underwater vehicles in URSim [8].

phenomenon, (ii) collision detection and friction models, (iii) Graphical User Interface (GUI), (iv) import capability for scenes and meshes, (v) API especially for programming language used by the robotics community (c++/Python), and (vi) models for an array of joints, actuators and sensors readily available. This review covers only simulators which are actively being developed, used, or maintained. It is our understanding that anything otherwise would be of limited importance to the research community in the long-term. Additionally, this review focuses on robotics simulators and not physics engines. Physics engines are integral to every robotics simulator, but a physics engine alone does not constitute a robotics simulator unless it satisfies all criteria we use to define what a robotics simulator is.

This review acts as a guide to assist researchers in short-listing the most relevant simulators for a given application to aid their decision making. We map out the landscape of current robotics simulators by categorising simulators that are actively maintained, and provide exemplar tasks and functionality that makes a simulator useful for a particular field or sub-domain. For each category we also provide a standardised summary table for concise communication of simulators and features. Figure 1 demonstrates the diversity of simulation environments required within robotics. The popularity of the robotics simulators discussed in sections II-VIII is portrayed visually in Figure 2.

We recommend researchers use this review as a guide for selecting a simulator for their particular research endeavour. We suggest first that a user has some idea of the particular robot platform(s) they wish to simulate (e.g. UR10, Husky, etc.), the method of actuation of the robotic platform(s), the sensors they intend to use and the physical operating environment (e.g. air, underwater, sand, city streets, etc.). The user then infers the relevant robotics research community from the denoted platform, so that, e.g., robot arms are found under manipulation. From here, we identify a subset of simulators that are likely to provide support for the required methods of actuation, sensors, and operating environment. If the planned endeavour is a crossover between fields, e.g., underwater manipulation, we suggest that all relevant sections be considered.

II. MOBILE GROUND ROBOTICS

Autonomous ground vehicle research – including legged, wheeled and tracked robots – is one of the largest studied domains in robotics. There are many fields which are incorporated in this sub-domain, including navigation, locomotion, cognition, control, perception, Simultaneous Localisation and Mapping (SLAM), and many others [9]. To motivate the use cases for simulators within mobile robotics we begin by investigating current challenges and competitions being run that represent some applications of mobile robotics research. The most prominent challenges are

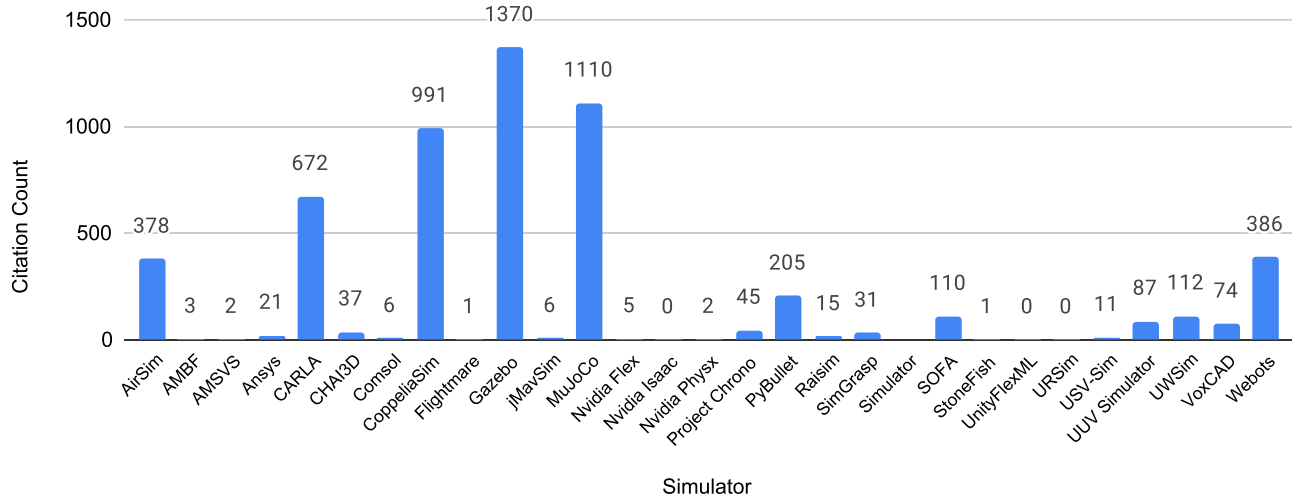

FIGURE 2. Citation count from 2016 to 2020 for reviewed simulators. Citations were gathered from Google Scholar using either one or more of a simulators' research paper, reference manual or other citation type and then filtered for robotics keyword.

TABLE 1. Feature comparison between popular robotics simulators.

Simulator	RGBD + LiDAR	Force Sensor	Linear + Cable Acutator	Multi-Body Import	Soft-Body Contacts	DEM Simulation	Fluid Mechanics	Headless Mode	ROS Support	HITL	Teleoperation	Realistic Rendering	Inverse Kinematics
Airsim	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓, unreal	✗
CARLA	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓, unreal	✗
CoppeliaSim	✓	✓	Linear only	✓	✗	✓	✗	✓	✓	✓	✓	✗	✓
Gazebo	✓	✓	Linear only	✓	✗	Through Fluidix	Through Fluidix	✓	✓	✓	✓	✗	✓
MuJoCo	✓	✓	✓	✓	✓	✓	Limited	✓	✗	HAPTIX only	HAPTIX only	✗	✗
PyBullet	✓	✓	Linear only	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓
SOFA	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓, Unity	✗
UWSim	RGBD only	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓, custom	✗
Chrono	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓, offline	✓
Webots	✓	✓	linear	✓	✗	✗	Limited	✓	✓	✗	✓	✗	✗

the Defense Advanced Research Projects Agency (DARPA) organised robotics challenges, starting in 2003 with an autonomous driving challenge, continuing in 2012 with a search and rescue challenge and now a subterranean competition running since 2018 [10]–[12]. The most current of these requires a team of robots to collaboratively navigate and map underground GPS denied environments to find human survivors, this event even hosts a virtual competition that is to be completed in the Gazebo simulation environment.

DJI has been running a robot competition featuring mobile robots since 2015. The task is to employ a team of robots in an arena to battle an opposing team using projectiles [13]. RoboCup is another prominent challenge that sees teams of mobile robots compete in games of football. Depending on the league, robots can either be legged or wheeled [14]. Each of these challenges require multiple robots to: coexist in the same environment and potentially interact with one another; navigate through and interact with terrain that may be geometrically uneven; and perceive the state of the robot in its environment with a suite of onboard sensors such as LiDAR, stereo-camera, GPS, or IMU.

Gazebo is a popular robotics simulator used in a wide range of mobile ground robot state-of-the-art (SOTA) research, for both legged [15], [16] and wheeled [17] robots. The Robot Operating System (ROS) interface provided by Gazebo contributes to the simulator's popularity, and simplifies the

process of testing control software in simulation and transferring it onto the physical system. Gazebo also offers a model library for many commonly used sensors such as camera, GPS, and IMU. Gazebo provides capability to import environments from digital elevation models, SDF meshes, and OpenStreetMap. It is also possible to import robot models from the Universal Robot Description Format (URDF) files. Being a rigid body simulator, the simulator runs quickly and can simulate multiple robots in real-time. Although Gazebo itself doesn't provide motion planning functionality, its tight integration with ROS allows ROS path planners to be used.

CoppeliaSim (previously V-Rep) is another popular choice for simulating ground-based mobile robots. SOTA research uses CoppeliaSim for navigation planning of bipedal robots [18], differential-drive robots [19], and visual trajectory tracking of differential-drive robots [20]. Similar to Gazebo, CoppeliaSim is a rigid-body simulator which is able to simulate multiple robots in real-time. It also supplies a large model library of common mobile robot platforms and sensors including 2D/3D laser, accelerometer, stereo-camera, camera, event camera, GPS, and gyro. CoppeliaSim offers path planning functionality through the commonly used OMPL library and supports height-fields for terrain specification.

Webots is another popular alternative, and is used in SOTA research to investigate, e.g., the performance of non-holonomic robot trajectory tracking [21], and to evolve bipedal robot gaits [22]. Webots has a large model database of mobile robots, environments and sensors. Sensors include accelerometer, camera, compass, GPS, inertial measurement unit, LIDAR, and radar. Webots also supports maps to be imported using the openDrive file format.

Raisim [7] is a rigid-body physics simulator developed by ETH Zurich, used in research into learning dynamic policies for legged platforms [23]. Raisim allows uneven terrain to be imported using height-map images. Raisim is not as full featured as other reviewed mobile robot simulators but instead is developed to provide high fidelity contact dynamics models which are needed for transferring controllers from simulation to reality.

SOTA work into transferring locomotion policies for quadrupeds from simulation to real-world platforms was conducted using the PyBullet simulator [24] along with research into visual navigation on the turtlebot platform [25]. Pybullet is applicable to most mobile robotics applications as it supports rigid-body simulation with the possibility of faster than real-time simulation of multiple robots. It also supports the import of height-fields for terrain geometry specification. The sensors supported by PyBullet are quite minimal compared to other more specific mobile robot simulators with support for cameras and several less relevant sensors.

The recent large scale investment into self-driving cars has yielded a comparably large amount of research into the field of autonomous cars with the CARLA simulator a by-product. Several SOTA papers use CARLA for learning driving policies [26] and transferring policies trained in CARLA to the real world [27]. CARLA is a simulator targeted at self-driving car research and therefore has features aligned with this goal. It is scaleable to allow for large scenes through a distributed architecture and includes ROS integration. CARLA provides support to import environments through the openDRIVE file format. A large number of sensors are available for use including GPS, IMU, LIDAR, Radar and Camera. CARLA uses PhysX to compute the vehicle physics although the settings available to the user are restricted.

Table 2 provides a comparison between the capabilities of the discussed robotics simulators in areas that are identified to be critical to the domain of mobile ground robots. Critical features identified include the ability to model sensors commonly used in the field, as well as common forms of locomotion, ability to import various environments, and inbuilt support for ROS. If the user wishes to realistically model complex environments, including sand, water, and gravel, Project Chrono provides this capability through an inbuilt Discrete Element Method (DEM) model [28].

III. MANIPULATION

The manipulation community within robotics is large and diverse, exploring everything from physical design of arms and grippers to algorithms for motion planning and control. Recent competitions and benchmarks provide insight into the current research avenues within the field and as an extension, the use cases for simulators within the manipulation research community.

One of the most prolific annual competitions is Robocup, which has a league called Robocup@home [29] for assistive robotics that compete in a domestic setting. Tasks that must be completed as part of the challenge rely upon manipulating rigid and deformable objects. The Tidy Up My Room Challenge from the International Conference on Robotics and Automation (ICRA) 2018 [30] and Fetch it! The Mobile Manipulation Challenge [31] are two additional challenges that are similar in that they all require multiple mobile manipulation platforms to complete multi-step tasks which include manipulation, often of rigid objects.

Other relevant challenges include the Amazon picking challenge, held between 2015 and 2017, which required robots to complete pick-and-place style tasks of seen and unseen objects, some of which were deformable and others which were transparent [32]. Most recently a challenge called the Real Robot Challenge looks at the dexterous manipulation of objects in both simulated and real-world environments. There are a number of benchmarks proposed within the manipulation community for benchmarking physical as well as algorithmic advances, relevant benchmarks to our review are task-based and include pick-and-place [33], assembly [34], peg-in-hole [35] and deformable objects [36].

From here we see that the field is at a state of simulating multi-task or multi-step scenarios, requiring the fine movement and contact modelling of rigid bodies. This, combined with the self-expletive need for stable physics that robustly handles contacts, reduces the field of viable simulators. To be

TABLE 2. Feature comparison between popular robotics simulators used for Mobile Ground Robotics.

Simulator	GPS	Tracks	Wheels	Legs	Mecanum / Omni Wheels	Heightmap Import	OpenDrive / OpenStreetMap	Pathplanning	ROS Support	RGBD	LiDAR	Realistic Rendering
Gazebo	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
CoppeliaSim	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
Raisim	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓, Unity
Webots	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
PyBullet	✗	✗	✓	✓	✗	✓	✗	✓	✗	✓	✓	✗
CARLA	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓, Unreal
Project Chrono	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓, POV-Ray

TABLE 3. Feature comparison for popular robotics simulators used for Manipulation.

Simulator	Pathplanning	Inverse Dynamics	Inverse Kinematics	Suction	Deformable Objects	Force/Torque Sensor	Realistic Rendering
SimGrasp	✓	✓	✓	✗	✗	✓	✗
Gazebo	✓	✓	✓	✗	✗	✓	✗
CoppeliaSim	✓	✗	✓	✓	✗	✓	✗
Pybullet	✓	✓	✓	✗	✓	✓	✗
MuJoCo	✗	✓	✗	✗	✓	✓	✗
NVidia Isaac	✓	✗	✓	✓	✓	✓	✓

useful for manipulation, research simulators must have actuator models for position control, velocity control, and torque control, as these are the most commonly used modes of control for physical arms. The simulator needs to support torque sensors as well as visual sensors, namely RGB and RGB-D. Finally, built-in features that are relevant specifically for manipulators are Inverse and Forward Kinematics solvers, and path planning. Less common – but becoming more relevant as computation becomes cheaper – is the need for modelling deformable objects as the underlying assumption that the robotics world is entirely rigid does not hold true.

Recent SOTA research has utilised a range of simulators for producing results. The capabilities of these simulators in areas relevant to the domain of robotic manipulation is summarised in Table 3. MuJoCo [37] is a simulator commonly employed within research, with notable contact stability being a reason for its popularity [38]. It was applied in an in-hand manipulation context for solving a Rubiks cube with a 24DOF robotic hand actuated with tendons [39]. SOTA research uses MuJoCO to train policies for robotic manipulators in simulation, whether just for proof of concept [40] or for later transferring onto real-world systems [41]–[43]. From the list of features a good manipulation simulation should have, MuJoCo supports most but lacks support for inverse kinematics and path planning.

Pybullet is used in studies with object collisions [44], pick and grasp dynamics [5] and for deformable object manipulation (i.e. cloth) [45]. Pybullet has a strong robotics target with functionality specifically implemented for those researching robotics. Functionality that may assist manipulation researchers includes: forward/inverse kinematics; Reinforcement Learning (RL) environments; Virtual Reality (VR) integration (for task demonstration); and deformable object and cloth simulation (Finite Element Method).

Gazebo [46] is used for robotic manipulation research [47], [48]. Although neither of these investigations rely on Gazebo to conduct dexterous manipulation, one of the studies did explicitly augment the simulation with external algorithms to deal with non-rigid bodies. Gazebo provides a simulation environment with the necessary actuators and sensors for robotic manipulation. It also provides support for ROS, which provides packages for forward and inverse kinematics, as well as path and motion planning.

CoppeliaSim is a robotics simulator with a range of user-centric features including sensor and actuator models, as well as motion planning, and forward and inverse kinematics support. PyRep was recently introduced as a python toolkit for robot learning built on CoppeliaSim, and has been shown to be capable of being used for manipulation, explicitly picking and placing cubes using a Kinova robot arm [49].

SimGrasp is used in a study for the design and simulation of tendon-driven underwater robotic hands [50]. It is a simulation package built on top of the Klampt't simulator, which markets itself as having better collision handling than the previously-mentioned manipulation simulators [51]. Klamp't lends itself to fast deployment of dexterous manipulation in robotics through the simulation of actuators and sensors with kinematics, dynamics and path planning.

Several works simulate deformable object manipulation using physics simulators that this review does not classify as robotics simulators. One such work uses Blender for cloth simulations [52], however this investigation only simulates cloth and the displacement of picked cloth coordinates. Another study uses Nvidia Flex to simulate fluids and deformable objects but abstracts away robot interactions [53]. Additionally, another work with Nvidia Flex simulates a robot completing a swinging peg-in-hole task with a 7-DoF Yumi robot. Flex is available through the ISAAC simulator [54].

IV. MEDICAL ROBOTICS

Medical robotics is a sub-domain of robotics research applying automation and robotics to e.g., surgery, therapy, rehabilitation and hospital automation [55]. Unlike competitions in other sub-fields of robotics, medical robotics does not have competitions focused on solving specific tasks, instead

TABLE 4. Feature comparison of popular robotics simulators used for Medical Robotics.

Simulator	Teleop	Haptic Feedback	Tendon Actuator	Deformable Objects	VR support
SOFA	✓	✓	✓	✓	Unity
Chai3D	✓	✓	✗	✓	✓
AMBF	✓	✓	✗	✓	✓
FLEX	✓	✓	✓	✓	Unity + Unreal
CoppeliaSim	✓	Chai3D plugin	✗	✗	✓

competitions are often judged based on the innovation of submissions. Examples of medical robotics competitions include the United Kingdom Robotics and Autonomous System Medical Robotics for Contagious Diseases Challenge [56] and the Kuka Medical Robotics Challenge [57]. The wide range of applications for medical robotics limits the number of medical robotics benchmarks to base the requirements of a medical robotics simulator on. We therefore base the requirements of medical robotics simulators on the needs of recent research being conducted in this field.

Due to the nature of therapeutic and rehabilitation interventions, studies are typically conducted in reality only. Instead, we focus on simulation for robotic surgery, including training and practice for real surgeries with a robotic surgical system, and training autonomous agents to attempt surgery in a safe environment.

The most prolific robotic surgical system is the da Vinci by Intuitive Inc., which consists of multiple arms with both rotational joints and tendon driven joints [58]. There are research robotic platforms that have been developed with similar hardware to the da Vinci, including the Raven II [59]. To realistically simulate these platforms, a simulator must be capable of simulating rotational as well as prismatic joints and should ideally simulate the tendons in both these robots for accuracy.

Surgeries performed with a robotic surgical system are often teleoperated by the surgeon. State of the art teleoperation controls have haptic feedback which gives the user force/torque feedback directly from the robot [60]. Robotics simulators for research into medical robotics benefit from having the tools to provide simulated force/torque feedback for haptic devices. Simulators which take in user input for teleoperation must run in real-time, otherwise user input would result in a delayed action in the environment. Robotic simulators for medical robotics also require deformable object simulation. As humans consist primarily of non-rigid tissue, realistic simulations require the ability to simulate deformable objects.

There are several simulators that offer some or all of the features required of a simulator in medical robotics. SOTA research in the domain of medical robotics uses robotics simulators such as the Simulation Open Framework Architecture (SOFA), CHAI3D [61], Asynchronous Multi-Body Framework (AMBF), CoppeliaSim, and UnityFlexML. The capabilities of the identified simulators in areas relevant to the domain of surgical robotics is summarised in Table 4.

SOFA is a medical simulator used to study force and thermal feedback methods for minimally invasive surgery [62]. SOFA offers a plugin for teleoperation and haptic control, supports real-time deformable object simulation and through a robotics plugin supports tendon driven joints. These features paired with the strong focus on medical applications makes SOFA a good candidate for medical robotics research requiring simulation.

CHAI3D is another simulation framework used in medical robotics research to learn a neural network for autonomous tissue manipulation in simulation [63]. CHAI3D supports multi-DOF teleoperation controllers and haptic feedback systems, real-time simulation, and deformable object simulation.

AMBF is a simulator for medical applications developed at the Automation and Interventional Medicine Robotics Research Laboratory at Worcester Polytechnic Institute, built around CHAI3D and the bullet physics engine [4], [64]. AMBF enables fast-running simulation which uses CHAI3D for teleoperation support and haptic feedback, and bullet for soft body simulation and prismatic joints.

UnityFlexML is a simulator developed for use in machine learning applications [65]. It is based on Unity which has a large network of supported plugins including Nvidia flex for deformable object simulation and teleoperation control with haptics.

The remaining simulators used for research in medical robotics offer a limited subset of the necessary features identified. For example CoppeliaSim is used as a deep reinforcement learning environment to train both pick and reach policies on a surgical robot where deformable objects are not required [66]. CoppeliaSim does however support teleoperation and haptic feedback through a CHAI3D plugin, and supports prismatic joints for realistic simulation of surgical robot joints.

Another simulation environment that was used for medical research was a custom implementation using Open Dynamics Engine (ODE). ODE was used in the presentation of a framework for training users for robotics surgery [67]. Although ODE is a physics-engine and not a robotics simulator, the users added additional support for teleoperation and haptic feedback for use with the Raven II surgical robot.

V. MARINE ROBOTICS

Simulators for marine robotics can be divided into two categories, namely those which are designed for underwater vehicles (AUVs, ROVs, etc) and those which are suitable

TABLE 5. Feature comparison of popular simulators used for marine robotics.

Simulator	Hydrodynamics	Hydrostatics	Thruster	Fins	Pressure Sensor	GPS	DVL	Realistic Rendering	Contact Dynamics	Wind	Waves	Water Currents
UWSim	✓	✗	✓	✗	✓	✓	✓	✓, osgOcean	✓	✗	✓	✗
UUV	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗
Stonefish	✓	✓	✓	✗	✓	✓	✓	✓, custom	✓	✓	✓	✓
URSim	✓	✓	✓	✗	✓	✗	✗	✓, Unity	✗	✗	✓	✗
USVSim	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓

for surface vehicles (USVs, Ships, boats, etc). Competitions such as the Singapore AUV Challenge (SAUVC) [68], RoboSub [69], RobotX [70] and MATE ROV [71] focus on practical, challenging missions. The interested reader is directed towards [72] which provides an excellent review of such competitions in the field of marine robotics and the summary in Table 5 that summarises the capabilities of the reviewed simulators. The design of marine robots is greatly aided by high fidelity simulation in areas such as navigation, waypoint following, seabed mapping, and sensor-based control [73]. A good simulator should support different types of controllable vehicles, manipulators, sensors, and complex environments with accurate representation of hydrodynamic/hydrostatic forces. UWSim [74] and UUV Simulator [75] are the two most widely used options for underwater simulation.

Unmanned Underwater Vehicle (UUV) Simulator [75] is an extension for Gazebo which supports multiple underwater vehicles (ROVs and AUVs) and robotic manipulators with high fidelity representation of hydrostatic and hydrodynamic forces. A number of commonly used sensors are included eg. underwater camera, pressure sensor, IMU, Magnetometer, Doppler Velocity Log (DVL), etc. Models for fins and thrusters are also included for actuation. UUV allows researchers to create complex underwater environments with models already included for seabeds, lakes, ship-wrecks, etc. UUV simulator has been used for applications such as mapping [76] and path following [77].

Other notable gazebo extensions/packages include Rock-Gazebo [78] and *freefloating − gazebo* [73]. ROCK-Gazebo is the integration between Gazebo and the Robot Construction Kit (ROCK) framework to allow for real-time simulation. This involved extending the ROCK visualisation tool using OpenSceneGraph(OSG) for rendering underwater environments while Gazebo was used for physics simulation [75]. Rock-Gazebo has a number of limitations including not supporting multi-robot simulations [75]. *freefloating − gazebo* combines the dynamic simulation capabilities of Gazebo with the realistic underwater rendering of UWSim [74]. This allows it to model hydrodynamic forces. *freefloating−gazebo* lacks in that, due to stability concerns, it does not include the computation of added-mass forces [75]. Rock-Gazebo and *freefloating − gazebo* both have limited sensor support.

UWSim [74] is another open source option, which was developed at the Interactive and Robotic Systems Lab at the Jaume-I University. It utilises Bullet and OpenSceneGraph(OSG) for contact physics and supports a wide range of simulated sensors such as pressure sensor, force sensor, GPS, range camera/sensor, IMU, DVL, and much more. Multiple vehicles can be loaded and managed simultaneously while complex environments can be modelled using OSG and other 3rd party tools such as Blender. The underwater rendering is highly realistic and it already includes a default model for girona500 and ARM5E manipulator. UWSim has been used in a number of applications including controller design [79], path planning [80], 3D mapping [81], etc. It does, however, lag behind in terms of accuracy of simulation for dynamics and hydrodynamics of vehicles [82]. It also does not support simulation of manipulator dynamics (only kinematics) [82].

The most credible alternative to UWSim and UUV Simulator is the newly proposed StoneFish Library [82], a wrapper for bullet which supports standard sensors such as camera, pressure, DVL, multi-beam, etc. All hydrodynamic computations are based on the actual geometry of the body, which allows for better approximation of hydrodynamic forces. The simulation effects include added mass, buoyancy and drag. Underwater thrusters and vehicle manipulator systems are available for modelling more complex setups. The simulator supports advanced rendering of underwater scenes, including scattering and light absorption. The advanced rendering is computationally expensive though and requires a recent GPU.

Unity ROS Simulator (URSim) [8] uses ROS and the Unity 3D game engine. It has sensor models for camera, IMU and pressure together with noise models for sensor input. The simulator is capable of modelling environments used in competitions such as SAUVC and RoboSub. Unity allows for the modelling of hydrodyanmic forces such as buoyancy and drag. ROS provides functionality needed for control, communication, vision and sensing, with target applications in sensing, mapping, path planning, localization, obstacle avoidance and target acquisition. URSim is being actively developed, with new sensors (DVL, side scan SONAR, etc) and a robotic manipulator planned.

Surface vehicle simulation is relatively rare [83], primarily due to the complexity associated with modelling environmental factors such as waves, wind and water currents [83], [84]. Unmanned Surface Vehicle simulator (USVSim) [83] is a dedicated simulator for this application, which is an extension of Gazebo [46]. The *freefloating* plugin [73] supports USV simulations by improving the hydrodynamics and buoyancy effects. The lift-drag plugin was used for calculating foil dynamics. UWSim [74] offers accurate modelling of wave and water visual effects. Re-using and improving elements from the above tools allowed the authors to come up with a robust simulator, which has been used for path planning [85].

TABLE 6. Feature comparison of popular simulators used for aerial robotics.

Simulator	Realistic Rendering	GPS	Barometer	Sonar	Radar	PX4	ArduPilot	HITL	RGBD + LiDAR	ROS Support	VR Support
AirSim	✓, Unreal + Unity	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓
Flightmare	✓, Unity	✗	✗	✗	✗	✗	✗	✗	RGBD only	✓	✓
Gazebo	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Webots	✗	✓	✗	✓	✓	✗	✓	✗	✓	✓	✓

VI. AERIAL ROBOTICS

In this section we focus on unmanned aerial vehicles (UAVs) which are the most popular field of research within aerial robotics.

Competitions such as the UAV challenge [86] and the International Aerial Robotics Competition (IARC) [87] are open competitions in the field. In the UAV challenge, the goal is to demonstrate the utility of UAVs on real-world missions such as medical rescue or delivery of essential items. IARC is the longest running collegiate competition for aerial robots, focusing on missions relating to human-robot interactions, robot-robot interactions and interactions of robots with complex environments. The NASA SAND (Safeguard with Autonomous Navigation Demonstration) [88] competition aims to address safety-critical risks associated with flying UAVs in US airspace.

Modern UAV simulators allow researchers to replicate complex real-world environments by modelling turbulence, air density, wind shear, clouds, precipitation and other fluid mechanics constraints [89]. They also support various sensors – eg. Lidars, GPS, camera, etc. Digital elevation models or height maps are also used to simulate the terrain underneath the UAV. Aerial robotics simulators include Gazebo, AirSim, Flightmare, jMAVSim, and Webots all of which are included in Table 6 along with a comparison of important features required for aerial robotic research.

Gazebo [46] is a popular simulator for both indoor and outdoor applications [90]. Gazebo relies on the LiftDrag Plugin to simulate aerodynamic properties, and supports many common sensors such as stereo-cameras and LIDAR. The Hector plugin [91] adds UAV-specific sensors such as barometers, GPS receivers and sonar rangers. Gazebo supports a comprehensive list of UAV models [46], and open-source hardware controllers such as Ardupilot and PX4 which can be integrated for hardware-in-the-loop simulations. Gazebo, however, features limited rendering capability compared to Unity and Unreal Engine [92]. Gazebo has found application in, e.g., autonomous navigation [93], landing on moving platforms [94], multi-UAV simulation [95], and visual servoing [96].

Microsoft's AirSim [92] is based on Unreal engine, and supports IMU, magnetometer, GPS, barometer, and camera sensors. AirSim provides a built-in controller called simple_flight, and also supports open-source controllers such as PX4. AirSim is resource-intensive and hence requires large computing power to run when compared with other simulators [89]. It has been used in drone racing [97], wildlife conservation [98], and depth perception from visual images [99].

Flightmare [100] combines a flexible physics engine with the Unity rendering engine into a powerful simulator. Flightmare simulates high-fidelity environments including warehouses and forests. Sensor models are available for IMU and RGB cameras with ground-truth depth and semantic segmentation. The simulator is well suited for applications in deep/reinforcement learning.

jMAVSim [101] is another widely used simulator, mainly due to its tight coupling with the open-source PX4 controller owing to the initial goal of testing PX4 firmware and devices [92]. jMavSim supports basic sensing and rendering [92].

Webots [102] is an open source simulator with an extensive set of supported sensors, including cameras, LIDARs, GPS, etc. Users can add custom physics to simulate things such as wind and integrate data from OpenStreetMap to create more realistic environments. Integration with the Adrupilot flight controller is supported. Webots has been used in multi-agent simulations [103], mitigation of bird strikes [104] and landing applications [105].

VII. SOFT ROBOTICS

Soft Robotics is generally a harder simulation problem than other robotics domains which often assume that the robot and world it operates in are mechanically rigid. Soft robotics requires simulating deformable objects and support for unconventional modes of actuation, including tendon or cable, pneumatic, and heat transference. Simulators must also support contact dynamics between the soft robot and soft/solid materials or fluids. As an emerging research field, competitions are relatively recent in their inception. The 2016 Robosoft Grand Challenge consisted of three team challenges: manipulation, terrestrial locomotion, and underwater locomotion [106]. The Annual Soft Robotics Competition ran annually between 2015 and 2018. It consisted of several categories, with a panel of judges awarding prizes based on contribution and design [107]. Table 7 provides a comparison between the capabilities of different robotics simulators used to simulate soft robots in areas relevant to the domain.

Soft robotics typically employs Multiphysics packages such as COMSOL, ANSYS, and Abaqus, which solve through Finite Element Method (FEM), as well as simulating aspects including heat transfer, electric conduction, magnetism, and fluid flow. They typically lack sensing, however they are fully capable of modelling soft actuation mechanisms and are used for more fundamental studies (e.g., not including environmental modelling). They have a range of modules

TABLE 7. Feature comparison of popular simulators used for soft robotics.

Simulator	FEA	Spring-Mass	Heat Conduction	Electrical Conduction	Magnetism	Pneumatics	Tendon
Evosoro	✗	✓	✗	✗	✗	✓	✗
SOFA	✓	✗	✓	✓	✗	✓	✓
COMSOL	✓	✗	✓	✓	✓	✓	✓
ABAQUS	✓	✗	✓	✓	✓	✓	✓
ANSYS	✓	✗	✓	✓	✓	✓	✓

available to support different physics, however tend to be expensive to purchase.

Abaqus, for example, is used to model laminar jamming structures [108], 3D locomotion of soft robots with electrostatic actuators [109], deflection of a soft robot produced by thermal conduction [110] and a soft robotic grippers [111]. Abaqus models non-linear behaviour well, and supports a large range of material properties with a material model library. The Abaqus FEM simulation is considered to be the industry standard. ANSYS is another modelling package used in soft robotics research, which simulates electrical, thermal and structural properties in simulation [112]. ANSYS Fluent is a well-developed package for fluid simulation, which is popular for e.g., underwater soft robotic modelling [113]. COMSOL has more user-definable material properties than Abaqus, lending itself to research methods which use materials with unique properties. It is also considered more user-friendly than ANSYS. In SOTA research, Comsol is used to simulate a flexible inchworm with actuation through magnetic fields [114], a caterpillar-like robot actuated by light [115].

SOFA is a popular open-source simulator that has been used to simulate cable driven soft robots [3], and for FEM simulation of four-legged soft robots [116]. SOFA has several useful features for robotics, including a ROS bridge and a Soft Robot plugin for modelling and actuation. Actuators from the soft robot plugin include tendon-driven and pneumatic actuators. SOFA is supported by an active open source community that regularly adds new modules and features alongside its internal development.

Evosoro is a soft robot simulator based on the Voxelyze physics engine. It uses Spring-Mass modelling for simulation of voxel-based soft robots and includes variable-volume actuation but no sensing. Evosoro is a comparatively fast soft robot simulator, which has been coupled with evolutionary algorithms to design robot morphologies [117], and as a design tool for real deployments of soft robots [118]. The simulator was found to have a significant gap when solutions were transferred to reality however, owing to the (fast, relatively inaccurate) Spring-Mass modelling.

VIII. LEARNING FOR ROBOTICS

Learning for robotics has been an important topic of research over the last decade. Due to the sample inefficiency of current Reinforcement Learning (RL) algorithms as well as the need to explore the state-action space that may lead to robot failure or damage during training, the majority of works on deep RL are first learned in simulation before being deployed on hardware. Due to the relative recency of robotic deep learning, there are relatively few competitions in the domain of learning for robotics. The Real Robot Challenge [119] is a notable exception in which participants learn dexterous manipulation of objects with a parallel manipulator, and their learned policies are compared both in simulation – in phase 1 of the competition – and hardware in a later phase. Tasks that must be completed as part of this challenge include pushing an object to a target location, lifting it to a specified height, and moving it to a target position and orientation.

Though there are few challenges targeted towards learning for robotics, learning methods have been applied in a number of other robotics challenges. A learned locomotion controller for a quadrupedal robot was recently deployed in the DARPA Subterranean challenge [123], for example.

Learning for robotics differs from the other sections covered in this work because it is concerned with implementation on a robot rather than the type of robot or the environment that a robot is deployed in. Due to the emerging popularity of robotic learning, a guide for simulator selection is included in this work. Learning methods can be applied in a wide range of robotics fields, and so the features pertinent to those fields should be considered alongside the features required for learning itself. For instance, applying learning methods to soft robotics requires support for soft contacts and materials as well as the ability to perform rapid iterative policy learning.

OpenAI Gym is a popular toolkit for training and evaluating RL algorithms, and provides environments in MuJoCo (Fig. 3) that are commonly used as baselines to evaluate new RL algorithms and methods in the literature [124]–[127]. OpenAI Gym is used in a number of simulators to train and evaluate learned policies, demonstrating the efficacy of these simulators for learning methods. Simulators used in conjunction with OpenAIGym include: PyBullet (Fig. 3) [128], Webots [129], Nvidia Flex [130], [131], Nvidia Isaac [132], CARLA [133], Project Chrono [134], Raisim [7], and Gazebo (Fig. 3) [135].

One common application of deep learning is to learn manipulation and grasping policies. For these tasks, the fidelity of rigid or soft body contact dynamics is important, as well as having sensors to support policies for such tasks. Another application is in path planning or locomotion over rough terrain with a mobile robot, and so researchers may require complex terrains to be modelled. Many

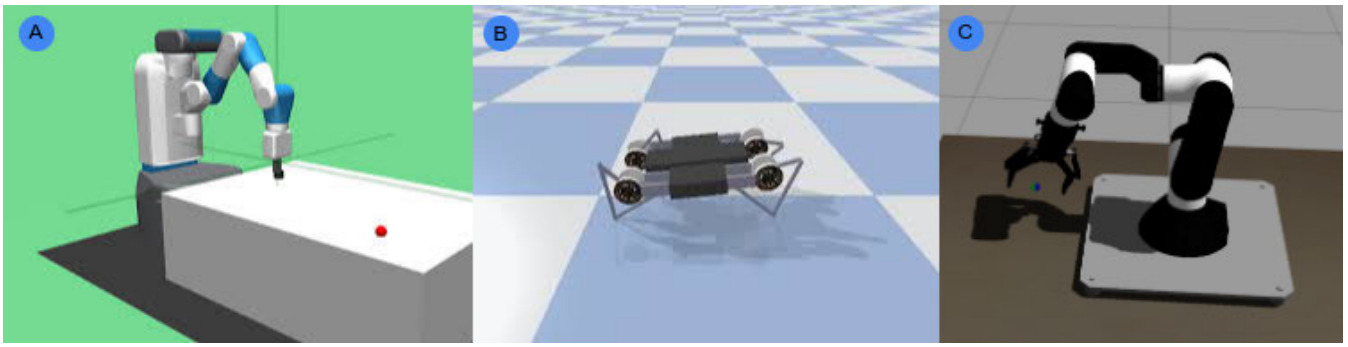

FIGURE 3. Robotics research within reinforcement learning relies heavily upon simulation environments with the ones pictures in A) MuJoCo [120], B) PyBullet [121], and C) Gazebo [122] being popular choices.

simulators such as Gazebo, Raisim, MuJoco, and PyBullet allow non-flat rigid terrains to be imported from heightmap images or mesh files, but do not model soft or granular materials at a large enough scale to mimic soils, gravels, or fluid terrains. Project Chrono is one alternative with in-built support for deformable terrain, granular terrain, and fluid simulations, as well as being parallelisable. Locomotion and path-planning policies for aerial robots such as quad-rotors has been achieved in Raisim [136], Flightmare [100], and Gazebo [137], [138].

In each of the described applications, sensor support is an important consideration for researchers to consider when selecting a suitable simulator. Force-torque sensors and vision sensors are common requirements and are supported in simulators such as Gazebo, PyBullet and CoppeliaSim. Gazebo provides support for noise models which can be applied to sensor outputs. Because simulation is an abstraction of real-world conditions, policies learned in simulation typically degrade when transferred onto hardware. Overcoming this *reality gap* is one of the most important considerations for researchers in the learning community to address when selecting a simulator. It is also important that simulation environments vary between episodes, commonly using a technique called domain randomisation [42], to diversify the training data and allow the robot to properly explore the shared state-action space. Many simulators have in-built support, e.g., the ability to reset a simulation environment without shutting down the entire simulator- and to vary initial positions and orientations of robots, cameras, and objects within the simulation- is common to many robotic simulators. The ability to randomise the textures of rendered objects in simulation and characteristics of the camera used to render them is built into MuJoCo and demonstrated in [42]. This functionality is not innately supported in Gazebo, though an external plugin has been created to do so [139]. Randomising object mass and inertia, as well as friction coefficients, is another method of performing domain randomisation common to many of the simulators considered. Applying small random forces to robots also aids in overcoming the reality gap but is not possible in all simulators. Supporting multiple physics engines is another domain randomisation technique that prevents learned policies over-fitting to the simulation environment [140]. Gazebo, V-Rep, and Pybullet support multiple back-end physics engines, whereas some other simulators such as Raisim and MuJoCo do not. The quality of rendering is also an important factor for learned policies that rely on visual data.

Due to the large amount of data needed to train neural network parameters and properly explore the state-action space it is also important that the chosen simulator facilitates collecting this data in a timely manner. There are a number of simulator features that can facilitate deep learning in a timely fashion, including: supporting parallel simulation – either through simulating multiple robots in one environment or running multiple simulations in parallel with multi-threading or multiprocessing – the ability to run in headless mode, and rapid dynamics solvers that allow simulations to run faster than real-time. Due to the GPU-based physics engine of Nvidia Flex – which is available to use as a physics engine in the NVidia Isaac robotic simulator – a walking policy for humanoid robots could be learned in 16 minutes on a single CPU and GPU. Flex also supports distributed GPU simulations which can further reduce training times by up to eight times on some tasks [130]. Flightmare is able to maintain 200,000 steps per second while simulating 150 quadrotors in parallel, allowing it to train locomotion policies for the quadrotors much faster than in real-time [100]. Running similar simulations of a humanoid robot in Gazebo, V-Rep, and Webots [141] showed that the Gazebo was more CPU-intensive than the other two simulators and Webots was the least intensive of the three. Computational load is relevant to researchers considering simulators for learning because it is important to perform either as many simulations in parallel as possible, or to run a simulation as quickly as possible so that training time can be reduced.

Evolutionary robotics is a subset of learning that is distinct from the majority of deep RL methods, though many of the challenges with simulating environments for deep RL are shared in the field of evolutionary robotics. Improving the reality gap is just as important for locomotion policies and part designs developed through evolutionary techniques as for those developed with deep RL, and evolutionary methods

TABLE 8. Feature comparison of popular simulators used in learning for robotics.

Simulator	Random External Forces	RGBD + LiDAR	Force Sensor	Multiple Physics Engines	Realistic Rendering
Raisim	✓	✓	✗	✗	✓, Unity
Gazebo	✓	✓	✓	✓	✗
Nvidia Isaac	✓	✓	✗	✗	✓, Unity + Unreal
MuJoCo	✓	✓	✓	✗	✗
PyBullet	✓	✓	✓	✗	✗
CARLA	✗	✓	✗	✗	✓, Unreal
Webots	✓	✓	✓	✗	✗
CoppeliaSim	✓	✓	✓	✓	✗

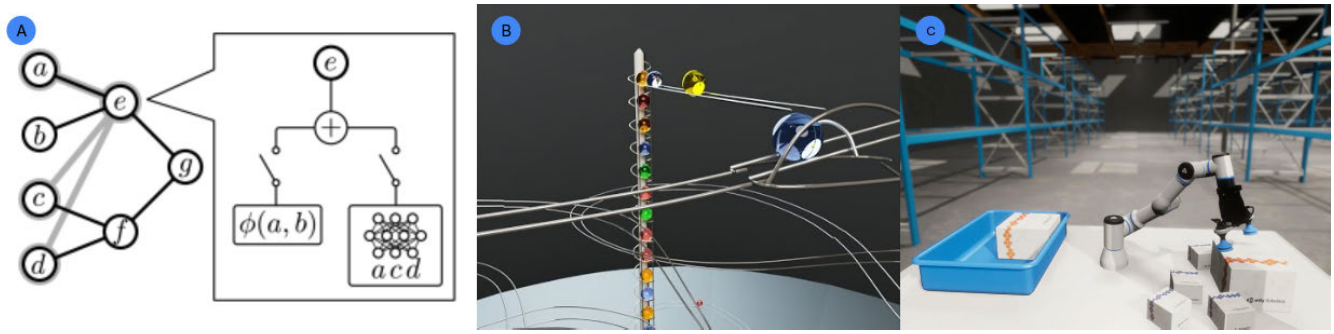

FIGURE 4. The future of robotic simulators is predicted to see advancements with A) widespread use of differentiable physics [151], B) increased stability and speed of simulation [155], and C) increased rendering capabilities within simulation [156].

also require a large number of time steps or simulations to be run. Simulators that have been used for evolutionary methods in the literature include ODE [142], Nvidia Physx [143], Bullet [144], [145], V-Rep [146], Gazebo [147], and Webots [148], voxcad [117], and Project Chrono [149].

Table 8 compares relevant features of common simulators used for learning in robotics. Features important to this field include: those that enable domain randomization, such as the ability to apply random external forces to the robot and employ multiple back-end physics engines; common sensors required by learned policies such as RGBD, LiDAR, and force sensors; and realistic rendering capabilities for learned policies that rely on visual data.

IX. FUTURE

Physical simulation is tightly intertwined with continued advances in robotics research. It is increasingly important, especially in fields such as robotic deep learning.

In a recent debate style workshop for sim-to-real, debaters proposed the progression of simulator accuracy as an important step in progressing simulator technology (Fig. 4) [150]. Improved accuracy can be attempted in a multitude of approaches as simulators abstract away real phenomenon, making a coarser representation of the world. The most prolific phenomenon to model well is contacts with large improvements likely to be seen with improved methods for collision detection and resolution. Collision detection is very resource intensive and is often a source of instability within simulators. One option is to replace phenomena that are difficult to model in simulation with a neural network that can be trained to replicate the properties of that phenomena with high accuracy, and integrate into the simulator [151].

Differentiable simulators are a fast growing area of research which is tightly coupled with robotics. The availability of automatic differentiation libraries contributes to the large number of new publications in this domain. The primary benefit of differentiable simulation — the ability to use gradient based rather than black-box optimisation approaches — promises a leap in efficiency and opens up previously intractable problems to learning-based optimisation. Several papers have proven examples which show the applicability of such simulators for system ID [152], policy creation [153] and embedding physics in neural networks (Fig. 4) [154].

Plugins and tools which are currently supported in some simulators will likely become even more prolific and ubiquitous. Features that are most likely to be adopted by a wider range of simulators include support for the ROS middleware, and integration with external renderers such as Unity or UnrealEngine for more realistic camera streams (Fig. 4). It is likely we will see more robotic simulators also integrate baseline tools for domain randomisation, system ID, and black-box optimisation.

We are also likely to see further integration in benchmarking and algorithmic frameworks. Examples of this are RL frameworks like OpenAI Gym [157], spinningup [158], and robosuite [159]. Benchmarks and algorithmic implementations will likely become embedded within the simulator framework much like path planners and kinematic solvers already are. This will make it easier to benchmark algorithms against the SOTA.

Finally, we predict that we will see further research into estimating and modelling uncertainty of simulators. Having a metric that encapsulates when a simulator is accurately projecting the real world is immensely advantageous. It provides researchers with an estimation of how likely a solution created in simulation will transfer to the real world, and where additional modelling is required [160].

X. CONCLUSION

Simulators aid robotics research in a multitude of ways. The benefits include reduction in cost, better management of time, and an added level of safety when dealing with complex environments. This review article provides a detailed summary on the type of simulators available for researchers in seven different, prominent, domains of robotics research. Each section covers a range of aspects including competitions, simulator support for features needed in each domain – sensors, actuators, environments – and the current SOTA. Section IX also provides a discussion on developments that we can expect to see in the not so distant future.

To the best of our knowledge, this is the first review article on robotic simulators covering such a diverse range in domains of robotics research. It is an excellent starting point for new researchers and a useful reference guide for experienced researchers. Hence, we hope that more studies like these are published over the coming years as new simulators enter the field and as some seasoned ones become obsolete.

ACKNOWLEDGMENT

(Jack Collins and Shelvin Chand contributed equally to the work.)

REFERENCES

[1] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, Havok, MuJoCo, ODE and PhysX," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 4397–4404.

[2] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, *Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics: Lecture Notes on Artificial Intellience), vol. 10965. Cham, Switzerland: Springer, Jul. 2018, pp. 357–368.

[3] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1413–1419, Jul. 2017.

[4] A. Munawar, Y. Wang, R. Gondokaryono, and G. S. Fischer, "A real-time dynamic simulator and an associated front-end representation format for simulating complex robots and environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1875–1882.

[5] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to throw arbitrary objects with residual physics," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1307–1319, Aug. 2020.

[6] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, Jan. 2020, doi: 10.1177/0278364919887447.

[7] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 895–902, Apr. 2018. [Online]. Available: www.raisim.com

[8] P. Katara, M. Khanna, H. Nagar, and A. Panaiyappan, "Open source simulator for unmanned underwater vehicles using ROS and Unity3D," in *Proc. IEEE Underwater Technol. (UT)*, Apr. 2019, pp. 1–7.

[9] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *Int. J. Adv. Robotic Syst.*, vol. 16, no. 2, Mar. 2019, Art. no. 1729881419839596, doi: 10.1177/1729881419839596.

[10] S. Thrun *et al.*, "Stanley: The robot that won the DARPA grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, Sep. 2006, doi: 10.1002/rob.20147.

[11] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski, "The DARPA robotics challenge finals: Results and perspectives," *J. Field Robot.*, vol. 34, no. 2, pp. 229–240, Mar. 2017, doi: 10.1002/rob.21683.

[12] I. D. Miller, F. Cladera, A. Cowley, S. S. Shivakumar, E. S. Lee, L. Jarin-Lipschitz, A. Bhat, N. Rodrigues, A. Zhou, A. Cohen, A. Kulkarni, J. Laney, C. J. Taylor, and V. Kumar, "Mine tunnel exploration using multiple quadrupedal robots," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2840–2847, Apr. 2020.

[13] Shenzhen DJI Sciences and Technologies Ltd. *RoboMaster Robotics Competition | Overview*. Accessed: Dec. 9, 2020. [Online]. Available: https://www.robomaster.com/en-US/robo/overview

[14] H. Kitano, M. Asada, I. Noda, and H. Matsubara, "RoboCup: Robot world cup," *IEEE Robot. Autom. Mag.*, vol. 5, no. 3, pp. 30–36, Sep. 1998.

[15] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1560–1567, Jul. 2018.

[16] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2261–2268, Jul. 2018.

[17] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ROS and Gazebo," in *Proc. 20th Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Oct. 2016, pp. 96–101.

[18] A. K. Rath, D. R. Parhi, H. C. Das, M. K. Muni, and P. B. Kumar, "Analysis and use of fuzzy intelligent technique for navigation of humanoid robot in obstacle prone zone," *Defence Technol.*, vol. 14, no. 6, pp. 677–682, Dec. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214914718300229

[19] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 31–36.

[20] J. Chen, B. Jia, and K. Zhang, "Trifocal tensor-based adaptive visual trajectory tracking control of mobile robots," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3784–3798, Nov. 2017.

[21] X. Wang, G. Zhang, F. Neri, T. Jiang, J. Zhao, M. Gheorghe, F. Ipate, and R. Lefticaru, "Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots," *Integr. Computer-Aided Eng.*, vol. 23, no. 1, pp. 15–30, Dec. 2015.

[22] C.-F. Juang and Y.-T. Yeh, "Multiobjective evolution of biped robot gaits using advanced continuous ant-colony optimized recurrent neural networks," *IEEE Trans. Cybern.*, vol. 48, no. 6, pp. 1910–1922, Jun. 2018.

[23] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, Jan. 2019, Art. no. eaau5872. [Online]. Available: http://robotics.sciencemag.org/content/4/26/eaau5872.abstract

[24] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," Apr. 2018, *arXiv:1804.10332*. [Online]. Available: http://arxiv.org/abs/1804.10332

[25] K. Chen, J. P. de Vicente, G. Sepulveda, F. Xia, A. Soto, M. Vazquez, and S. Savarese, "A behavioral approach to visual navigation with graph localization networks," 2019, *arXiv:1903.00445*. [Online]. Available: https://arxiv.org/abs/1903.00445

[26] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4693–4700.

[27] J. Zhang, L. Tai, P. Yun, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard, "VR-goggles for robots: Real-to-sim domain adaptation for visual control," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1148–1155, Apr. 2019.

[28] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *High Performance Computing in Science and Engineering* (Lecture Notes in Computer Science), T. Kozubek, R. Blaheta, J. Šístek, M. Rozložník, and M. Čermák, Eds. Cham, Switzerland: Springer, 2016, pp. 19–49.
[29] F. Jumel, "Advancing research at the RoboCupHome competition [competitions]," *IEEE Robot. Autom. Mag.*, vol. 26, no. 2, pp. 7–9, Jun. 2019.
[30] J. Leitner. *Tidy Up My Room Challenge*. Accessed: Aug. 10, 2020. [Online]. Available: http://juxi.net/challenge/tidy-up-my-room/
[31] *Fetchit! The Mobile Manipulation Challenge*. Accessed: Aug. 11, 2020. [Online]. Available: https://opensource.fetchrobotics.com/competition
[32] D. Morrison *et al.*, "Cartman: The low-cost Cartesian manipulator that won the Amazon robotics challenge," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7757–7764.
[33] A. S. Morgan, K. Hang, W. G. Bircher, F. M. Alladkani, A. Gandhi, B. Calli, and A. M. Dollar, "Benchmarking cluttered robot pick-and-place manipulation with the box and blocks test," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 454–461, Apr. 2020.
[34] K. Chatzilygeroudis, B. Fichera, I. Lauzana, F. Bu, K. Yao, F. Khadivar, and A. Billard, "Benchmark for bimanual robotic manipulation of semi-deformable objects," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2443–2450, Apr. 2020.
[35] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set," *IEEE Robot. Autom. Mag.*, vol. 22, no. 3, pp. 36–52, Sep. 2015.
[36] I. Garcia-Camacho, M. Lippi, M. C. Welle, H. Yin, R. Antonova, A. Varava, J. Borras, C. Torras, A. Marino, G. Alenyà, and D. Kragic, "Benchmarking bimanual cloth manipulation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1111–1118, Apr. 2020.
[37] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033. [Online]. Available: http://ieeexplore.ieee.org/document/6386109/
[38] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Proc. Robot., Sci. Syst.*, Pittsburgh, PA, USA, Jun. 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/index.html
[39] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving Rubik's cube with a robot hand," Oct. 2019, *arXiv:1910.07113*. [Online]. Available: http://arxiv.org/abs/1910.07113
[40] D. Pathak, D. Gandhi, and A. Gupta, "Self-supervised exploration via disagreement," in *Proceedings of Machine Learning Research*, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds., Long Beach, CA, USA, Aug. 2019, pp. 5062–5071. [Online]. Available: http://proceedings.mlr.press/v97/pathak19a.html
[41] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," Oct. 2016, *arXiv:1610.03518*. [Online]. Available: http://arxiv.org/abs/1610.03518
[42] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.
[43] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Proceedings of Machine Learning Research*, vol. 78, S. Levine, V. Vanhoucke and K. Goldberg, Eds., Aug. 2017, pp. 262–270. [Online]. Available: http://proceedings.mlr.press/v78/rusu17a.html
[44] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Proceedings of Machine Learning Research*, vol. 78, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., Aug. 2017, pp. 515–524. [Online]. Available: http://proceedings.mlr.press/v78/mahler17a.html
[45] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," pp. 734–743, Aug. 2018, *arXiv:1806.07851*. [Online]. Available: http://arxiv.org/abs/1806.07851
[46] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Sep. 2004, pp. 2149–2154. [Online]. Available: http://ieeexplore.ieee.org/document/1389727/
[47] L. Kunze and M. Beetz, "Envisioning the qualitative effects of robot manipulation actions using simulation-based projections," *Artif. Intell.*, vol. 247, pp. 352–380, Jun. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370214001544
[48] H. Jin, Q. Chen, Z. Chen, Y. Hu, and J. Zhang, "Multi-LeapMotion sensor based demonstration for robotic refine tabletop object manipulation task," *CAAI Trans. Intell. Technol.*, vol. 1, no. 1, pp. 104–113, Jan. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2468232216000111
[49] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Proceedings of Machine Learning Research*, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78, Aug. 2017, pp. 334–343. [Online]. Available: http://proceedings.mlr.press/v78/james17a.html
[50] H. Stuart, S. Wang, O. Khatib, and M. R. Cutkosky, "The ocean one hands: An adaptive design for robust marine manipulation," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 150–166, Feb. 2017, doi: 10.1177/0278364917694723.
[51] K. Hauser, "Robust contact generation for robot simulation with unstructured meshes," in *Proc. 16th Int. Symp. Robot. Res. (ISRR)*, M. Inaba and P. Corke, Eds. Cham, Switzerland: Springer, 2016, pp. 357–373.
[52] D. Tanaka, S. Arnold, and K. Yamazaki, "EMD net: An encode–manipulate–decode network for cloth manipulation," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1771–1778, Jan. 2018.
[53] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," in *Proc. Int. Conf. Learn. Represent.*, 2019. [Online]. Available: https://openreview.net/forum?id=rJgbSn09Ym
[54] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8973–8979.
[55] J. Troccaz, G. Dagnino, and G.-Z. Yang, "Frontiers of medical robotics: From concept to systems to clinical translation," *Annu. Rev. Biomed. Eng.*, vol. 21, no. 1, pp. 193–218, Jun. 2019, doi: 10.1146/annurev-bioeng-060418-052502.
[56] *Medical Robotics for Contagious Diseases Challenge 2020—UK-RAS Network*. Accessed: Dec. 9, 2020. [Online]. Available: https://www.ukras.org/robotics-week/challenges/medical-robotics-for-contagious-diseases-challenge-2020/
[57] *KUKA Innovation Award 2020|KUKA AG*. Accessed: Dec. 9, 2020. [Online]. Available: https://www.kuka.com/en-au/future-production/research-and-development/kuka-innovation-award/kuka-innovation-award-2020
[58] C. Freschi, V. Ferrari, F. Melfi, M. Ferrari, F. Mosca, and A. Cuschieri, "Technical review of the da vinci surgical telemanipulator," *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 9, no. 4, pp. 396–406, Dec. 2013, doi: 10.1002/rcs.1468.
[59] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, and L. White, "Raven-II: An open platform for surgical robotics research," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 4, pp. 954–959, Apr. 2013.
[60] N. Enayati, E. De Momi, and G. Ferrigno, "Haptics in robot-assisted surgery: Challenges and benefits," *IEEE Rev. Biomed. Eng.*, vol. 9, pp. 49–65, 2016.
[61] F. Conti, F. Barbagli, R. Balaniuk, M. Halg, C. Lu, D. Morris, L. Sentis, J. Warren, O. Khatib, and K. Salisbury, "The CHAI libraries," in *Proc. Eurohaptics*, Dublin, Ireland, 2003, pp. 496–500.
[62] M. Guiatni, V. Riboulet, C. Duriez, A. Kheddar, and S. Cotin, "A combined force and thermal feedback interface for minimally invasive procedures simulation," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 3, pp. 1170–1181, Jun. 2013.
[63] C. Shin, P. W. Ferguson, S. A. Pedram, J. Ma, E. P. Dutson, and J. Rosen, "Autonomous tissue manipulation via surgical robot using learning based model predictive control," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3875–3881.
[64] E. Coumans, "Bullet physics simulation," in *Proc. ACM SIGGRAPH Courses (SIGGRAPH)*, New York, NY, USA, 2015. Accessed: Aug. 24, 2020, doi: 10.1145/2776880.2792704.
[65] E. Tagliabue, A. Pore, D. Alba, E. Magnabosco, M. Piccinelli, and P. Fiorini, "Soft tissue simulation environment to learn manipulation tasks in autonomous robotic surgery," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Las Vegas, NV, USA, Jan. 2020, pp. 3261–3266. Accessed: Nov. 26, 2020.

[66] F. Richter, R. K. Orosco, and M. C. Yip, "Open-sourced reinforcement learning environments for surgical robotics," 2019, *arXiv:1903.02090*. [Online]. Available: http://arxiv.org/abs/1903.02090
[67] X. Li, H. Alemzadeh, D. Chen, Z. Kalbarczyk, R. K. Iyer, and T. Kesavadas, "Surgeon training in telerobotic surgery via a hardware-in-the-loop simulator," *J. Healthcare Eng.*, vol. 2017, Jan. 2017, Art. no. 6702919, doi: 10.1155/2017/6702919.
[68] IEEE-OES. *Singapore AUV Challenge*. Accessed: Dec. 15, 2020. [Online]. Available: https://sauvc.org/
[69] RoboNation. *RoboSub*. Accessed: Dec. 15, 2020. [Online]. Available: https://robosub.org/
[70] *RobotX*. Accessed: Dec. 15, 2020. [Online]. Available: https://robotx.org/
[71] *MATE ROV Competition*. Accessed: Dec. 15, 2020. [Online]. Available: https://materovcompetition.org/
[72] F. Ferreira and G. Ferri, "Marine robotics competitions: A survey," *Current Robot. Rep.*, vol. 1, no. 4, pp. 169–178, Dec. 2020, doi: 10.1007/s43154-020-00022-5.
[73] O. Kermorgant, "A dynamic simulator for underwater vehicle-manipulators," in *Simulation, Modeling, and Programming for Autonomous Robots* (Lecture Notes in Computer Science), D. Brugali, J. F. Broenink, T. Kroeger, and B. A. MacDonald, Eds. Cham, Switzerland: Springer, 2014, pp. 25–36.
[74] M. Prats, J. Perez, J. J. Fernandez, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 2577–2582.
[75] M. M. M. Manhaes, 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 *Proc. OCEANS MTS/IEEE Monterey*, Sep. 2016, pp. 1–8.
[76] B.-J. Ho, P. Sodhi, P. Teixeira, M. Hsiao, T. Kusnur, and M. Kaess, "Virtual occupancy grid map for submap-based pose graph SLAM and planning in 3D environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 2175–2182.
[77] Q. Zhang, J. Lin, Q. Sha, B. He, and G. Li, "Deep interactive reinforcement learning for path following of autonomous underwater vehicle," *IEEE Access*, vol. 8, pp. 24258–24268, 2020.
[78] 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 *Proc. 12th Latin Amer. Robot. Symp. 3rd Brazilian Symp. Robot. (LARS-SBR)*, Oct. 2015, pp. 132–138.
[79] P. Kormushev and D. G. Caldwell, "Towards improved AUV control through learning of periodic signals," in *Proc. OCEANS*, San Diego, CA, USA, Sep. 2013, pp. 1–4.
[80] M. Carreras, J. D. Hernández, E. Vidal, N. Palomeras, D. Ribas, and P. Ridao, "Sparus II AUV—A hovering vehicle for seabed inspection," *IEEE J. Ocean. Eng.*, vol. 43, no. 2, pp. 344–355, Apr. 2018.
[81] Z. Shen, J. Song, K. Mittal, and S. Gupta, "Autonomous 3-D mapping and safe-path planning for underwater terrain reconstruction using multi-level coverage trees," in *Proc. OCEANS*, Anchorage, AK, USA, Sep. 2017, pp. 1–6.
[82] P. Cieślak, "Stonefish: An advanced open-source simulation tool designed for marine robotics, with a ROS interface," in *Proc. OCEANS*, Marseille, France, Jun. 2019, pp. 1–6.
[83] M. Paravisi, D. H. Santos, V. Jorge, G. Heck, L. Gonçalves, and A. Amory, "Unmanned surface vehicle simulator with realistic environmental disturbances," *Sensors*, vol. 19, no. 5, p. 1068, Mar. 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6427536/
[84] Y. Shi, C. Shen, H. Fang, and H. Li, "Advanced control in marine mechatronic systems: A survey," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 3, pp. 1121–1131, Jun. 2017.
[85] A. G. D. S. Silva Junior, D. H. D. Santos, A. P. F. D. Negreiros, J. M. V. B. D. S. Silva, and L. M. G. Gonçalves, "High-level path planning for an autonomous sailboat robot using Q-Learning," *Sensors*, vol. 20, no. 6, p. 1550, Mar. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/6/1550
[86] *UAV Challenge*. Accessed: Dec. 15, 2020. [Online]. Available: https://uavchallenge.org/
[87] AUVSI-Foundation. *International Aerial Robotics Competition (IARC)*. Accessed: Dec. 15, 2020. [Online]. Available: http://www.aerialroboticscompetition.org/
[88] NASA. *NASA SAND Challenge*. Accessed: Dec. 15, 2020. [Online]. Available: https://www.nasa.gov/sand
[89] A. I. Hentati, L. Krichen, M. Fourati, and L. C. Fourati, "Simulation tools, environments and frameworks for UAV systems performance analysis," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2018, pp. 1495–1500.
[90] M. Schmittle, A. Lukina, L. Vacek, J. Das, C. P. Buskirk, S. Rees, J. Sztipanovits, R. Grosu, and V. Kumar, "OpenUAV: A UAV testbed for the CPS and robotics community," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2018, pp. 130–139.
[91] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and Gazebo," in *Simulation, Modeling, and Programming for Autonomous Robots*, I. Noda, N. Ando, D. Brugali, and J. J. Kuffner, Eds. Berlin, Germany: Springer, 2012, pp. 400–411.
[92] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," Jul. 2017, *arXiv:1705.05065*. [Online]. Available: http://arxiv.org/abs/1705.05065
[93] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, "Autonomous navigation of UAV by using real-time model-based reinforcement learning," in *Proc. 14th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2016, pp. 1–6.
[94] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, P. de la Puente, and P. Campoy, "A deep reinforcement learning strategy for UAV autonomous landing on a moving platform," *J. Intell. Robotic Syst.*, vol. 93, nos. 1–2, pp. 351–366, Feb. 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10846-018-0891-8, doi: 10.1007/s10846-018-0891-8.
[95] N. Mahdoui, V. Frémont, and E. Natalizio, "Communicating multi-UAV system for cooperative SLAM-based exploration," *J. Intell. Robotic Syst.*, vol. 98, no. 2, pp. 325–343, May 2020, doi: 10.1007/s10846-019-01062-6.
[96] H. Shi, X. Li, K.-S. Hwang, W. Pan, and G. Xu, "Decoupled visual servoing with fuzzy Q-learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 241–252, Jan. 2018.
[97] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "AirSim drone racing lab," 2020, *arXiv:2003.05654*. [Online]. Available: http://arxiv.org/abs/2003.05654
[98] E. Bondi, D. Dey, A. Kapoor, J. Piavis, S. Shah, F. Fang, B. Dilkina, R. Hannaford, A. Iyer, L. Joppa, and M. Tambe, "AirSim-W: A simulation environment for wildlife conservation with UAVs," in *Proc. 1st ACM SIGCAS Conf. Comput. Sustain. Societies*, New York, NY, USA, Jun. 2018, pp. 1–12, doi: 10.1145/3209811.3209880.
[99] K. Julian, J. Mern, and R. Tompa, "UAV depth perception from visual images using a deep convolutional neural network," Stanford Univ., Stanford, CA, USA, Tech. Rep., 2017.
[100] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," 2020, *arXiv:2009.00563*. [Online]. Available: http://arxiv.org/abs/2009.00563
[101] A. Babushkin. *jMavSim*. Accessed: Dec. 15, 2020. [Online]. Available: https://github.com/PX4/jMAVSim
[102] O. Michel, "Cyberbotics Ltd. Webots: Professional mobile robot simulation," *Int. J. Adv. Robotic Syst.*, vol. 1, no. 1, pp. 39–42, Mar. 2004. [Online]. Available: http://journals.sagepub.com/doi/10.5772/5618
[103] Z. Obdržálek, "Software environment for simulation of UAV multi-agent system," in *Proc. 21st Int. Conf. Methods Models Autom. Robot. (MMAR)*, Aug. 2016, pp. 720–725.
[104] P. Aliasghari, K. Dautenhahn, and C. L. Nehaniv, "Simulations on herding a flock of birds away from an aircraft using an unmanned aerial vehicle," in *Proc. Conf. Artif. Life*, 2020, pp. 626–635.
[105] V. Stary, R. Doskocil, V. Krivanek, P. Kutilek, and A. Stefek, "Missile guidance systems for UAS landing application," in *Proc. 17th Int. Conf. Mechatronics-Mechatronika (ME)*, Dec. 2016, pp. 1–5.
[106] M. Calisti, M. Cianchetti, M. Manti, F. Corucci, and C. Laschi. (2016). *Contest-Driven Soft-Robotics Boost: The RoboSoft Grand Challenge*. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2016.00055
[107] D. P. Holland, C. Abah, M. Velasco-Enriquez, M. Herman, G. J. Bennett, E. A. Vela, and C. J. Walsh, "The soft robotics toolkit: Strategies for overcoming obstacles to the wide dissemination of soft-robotic hardware," *IEEE Robot. Autom. Mag.*, vol. 24, no. 1, pp. 57–64, Mar. 2017.
[108] Y. S. Narang, J. J. Vlassak, and R. D. Howe, "Mechanically versatile soft machines through laminar jamming," *Adv. Funct. Mater.*, vol. 28, no. 17, Apr. 2018, Art. no. 1707136, doi: 10.1002/adfm.201707136.
[109] J. Cao, L. Qin, J. Liu, Q. Ren, C. C. Foo, H. Wang, H. P. Lee, and J. Zhu, "Untethered soft robot capable of stable locomotion using soft electrostatic actuators," *Extreme Mech. Lett.*, vol. 21, pp. 9–16, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352431617302250

[110] C. Ahn, X. Liang, and S. Cai, "Bioinspired design of light-powered crawling, squeezing, and jumping untethered soft robot," *Adv. Mater. Technol.*, vol. 4, no. 7, Jul. 2019, Art. no. 1900185, doi: 10.1002/admt.201900185.
[111] Z. Wang, R. Kanegae, and S. Hirai, "Circular shell gripper for handling food products," *Soft Robot.*, Aug. 2020, doi: 10.1089/soro.2019.0140.
[112] M. Al-Rubaiai, T. Pinto, C. Qian, and X. Tan, "Soft actuators with stiffness and shape modulation using 3D-printed conductive polylactic acid material," *Soft Robot.*, vol. 6, no. 3, pp. 318–332, Jun. 2019, doi: 10.1089/soro.2018.0056.
[113] C. Yue, S. Guo, and M. Li, "ANSYS Fluent-based modeling and hydrodynamic analysis for a spherical underwater robot," in *Proc. IEEE Int. Conf. Mechatronics Autom.*, Aug. 2013, pp. 1577–1581.
[114] E. B. Joyee and Y. Pan, "A fully three-dimensional printed inchworm-inspired soft robot with magnetic actuation," *Soft Robot.*, vol. 6, no. 3, pp. 333–345, Jun. 2019, doi: 10.1089/soro.2018.0082.
[115] M. Rogó , H. Zeng, C. Xuan, D. S. Wiersma, and P. Wasylczyk, "Light-driven soft robot mimics caterpillar locomotion in natural scale," *Adv. Opt. Mater.*, vol. 4, no. 11, pp. 1689–1694, Nov. 2016, doi: 10.1002/adom.201600503.
[116] Z. Zhang, J. Dequidt, A. Kruszewski, F. Largilliere, and C. Duriez, "Kinematic modeling and observer based control of soft robot using real-time finite element method," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 5509–5514.
[117] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding," in *Proc. 15th Annu. Conf. Genet. Evol. Comput. (GECCO)*, Amsterdam, The Netherlands. New York, NY, USA: Association for Computing Machinery, 2013, pp. 167–174, doi: 10.1145/2463372.2463404.
[118] S. Kriegman, A. M. Nasab, D. Shah, H. Steele, G. Branin, M. Levin, J. Bongard, and R. Kramer-Bottiglio, "Scalable sim-to-real transfer of soft robot designs," in *Proc. 3rd IEEE Int. Conf. Soft Robot. (RoboSoft)*, May 2020, pp. 359–366.
[119] Max Planck Institute for Intelligent Systems. *Real Robot Challenge*. Accessed: Oct. 30, 2020. [Online]. Available: https://real-robot-challenge.com
[120] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," 2018, *arXiv:1802.09464*. [Online]. Available: http://arxiv.org/abs/1802.09464
[121] B. Delhaisse, L. Rozo, and D. G. Caldwell, "PyRoboLearn: A python framework for robot learning practitioners," in *Proc. Conf. Robot Learn. (PMLR)*, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds. vol. 100, 2020, pp. 1348–1358. [Online]. Available: http://proceedings.mlr.press/v100/delhaisse20a.html
[122] N. G. Lopez, Y. L. E. Nuin, E. B. Moral, L. U. S. Juan, A. S. Rueda, V. M. Vilches, and R. Kojcev, "Gym-Gazebo2, a toolkit for reinforcement learning using ROS 2 and Gazebo," 2019, *arXiv:1903.06278*. [Online]. Available: http://arxiv.org/abs/1903.06278
[123] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. (2020). *Animal Robots Learning Quadrupedal Locomotion Over Challenging Terrain*. [Online]. Available: http://robotics.sciencemag.org/
[124] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 4111–4117.
[125] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," Mar. 2017, *arXiv:1703.03864*. [Online]. Available: http://arxiv.org/abs/1703.03864
[126] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Long Beach, Ca, USA. Red Hook, NY, USA: Curran Associates, 2017, pp. 5285–5294.
[127] H. Zheng, P. Wei, J. Jiang, G. Long, and C. Zhang, "Cooperative heterogeneous deep reinforcement learning," Tech. Rep. [Online]. Available: https://arxiv.org/pdf/2011.00791.pdf
[128] E. Coumans and Y. Bai. (2017). *Pybullet, a Python Module for Physics Simulation in Robotics, Games and Machine Learning*. [Online]. Available: https://pybullet.org
[129] M. Kirtas, K. Tsampazis, N. Passalis, and A. Tefas, "Deepbots: A webots-based deep reinforcement learning framework for robotics," in *Proc. IFIP Int. Conf. Artif. Intell. Appl. Innov.* Cham, Switzerland: Springer, 2020, pp. 64–75.
[130] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "GPU-accelerated robotic simulation for distributed reinforcement learning," 2018, *arXiv:1810.05762*. [Online]. Available: http://arxiv.org/abs/1810.05762
[131] X. Lin, Y. Wang, J. Olkin, and D. Held, "SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation," 2020, *arXiv:2011.07215*. [Online]. Available: http://arxiv.org/abs/2011.07215
[132] P. Klink, C. D'Eramo, J. Peters, and J. Pajarinen, "Self-paced deep reinforcement learning," 2020, *arXiv:2004.11812*. [Online]. Available: http://arxiv.org/abs/2004.11812
[133] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," 2020, *arXiv:2001.08726*. [Online]. Available: http://arxiv.org/abs/2001.08726
[134] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *Proc. Int. Conf. High Perform. Comput. Sci. Eng.*, 2015, pp. 19–49.
[135] N. G. Lopez, Y. L. E. Nuin, E. B. Moral, L. U. S. Juan, A. S. Rueda, V. M. Vilches, and R. Kojcev, "Gym-Gazebo2, a toolkit for reinforcement learning using ROS 2 and Gazebo," 2019, *arXiv:1903.06278*. [Online]. Available: http://arxiv.org/abs/1903.06278
[136] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2096–2103, Oct. 2017.
[137] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the OpenAI gym for robotics: A toolkit for reinforcement learning using ROS and Gazebo," Aug. 2016, *arXiv:1608.05742*. [Online]. Available: http://arxiv.org/abs/1608.05742
[138] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for UAV attitude control," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 2, pp. 1–21, Mar. 2019, doi: 10.1145/3301273.
[139] J. Borrego, R. Figueiredo, A. Dehban, P. Moreno, A. Bernardino, and J. Santos-Victor, "A generic visual perception domain randomisation framework for Gazebo," [Online]. Available: http://www.ros.org/
[140] D. Ferigo, S. Traversaro, G. Metta, and D. Pucci, "Gym-ignition: Reproducible robotic simulations for reinforcement learning," Nov. 2019, *arXiv:1911.01715*. [Online]. Available: http://arxiv.org/abs/1911.01715
[141] A. Ayala, F. Cruz, D. Campos, R. Rubio, B. Fernandes, and R. Dazeley, "A comparison of humanoid robot simulators: A quantitative approach," 2020, *arXiv:2008.04627*. [Online]. Available: http://arxiv.org/abs/2008.04627
[142] J. E. Auerbach and J. C. Bongard, "Environmental influence on the evolution of morphological complexity in machines," *PLoS Comput. Biol.*, vol. 10, no. 1, Jan. 2014, Art. no. e1003399.
[143] J. Rieffel, D. Knox, S. Smith, and B. Trimmer, "Growing and evolving soft robots," *Artif. Life*, vol. 20, no. 1, pp. 143–162, Jan. 2014.
[144] C. J. Pretorius, M. C. du Plessis, and J. W. Gonsalves, "Evolutionary robotics applied to hexapod locomotion: A comparative study of simulation techniques," *J. Intell. Robotic Syst.*, vol. 96, nos. 3–4, pp. 363–385, Dec. 2019.
[145] M. van Diepen and K. Shea, "A spatial grammar method for the computational design synthesis of virtual soft locomotion robots," *J. Mech. Des.*, vol. 141, no. 10, Oct. 2019, Art. no. 101402.
[146] M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen, "Evolution of repertoire-based control for robots with complex locomotor systems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 314–328, Apr. 2018.
[147] J. Nordmoen, K. O. Ellefsen, and K. Glette, "Combining MAP-elites and incremental evolution to generate gaits for a mammalian quadruped robot," in *Applications of Evolutionary Computation*, K. Sim and P. Kaufmann, Eds. Cham, Switzerland: Springer, 2018, pp. 719–733.
[148] C.-F. Juang, Y.-H. Jhan, Y.-M. Chen, and C.-M. Hsu, "Evolutionary wall-following hexapod robot using advanced multiobjective continuous ant colony optimized fuzzy controller," *IEEE Trans. Cognit. Develop. Syst.*, vol. 10, no. 3, pp. 585–594, Sep. 2018.
[149] J. Collins, W. Geles, D. Howard, and F. Maire, "Towards the targeted environment-specific evolution of robot components," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2018, pp. 61–68.
[150] S. Höfer, K. Bekris, A. Handa, J. Camilo Gamboa, F. Golemo, M. Mozifian, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. K. Liu, J. Peters, S. Song, P. Welinder, and M. White, "Perspectives on Sim2Real transfer for robotics: A summary of the R:SS 2020 workshop," Dec. 2020, *arXiv:2012.03806*. [Online]. Available: http://arxiv.org/abs/2012.03806
[151] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "NeuralSim: Augmenting differentiable simulators with neural networks," Nov. 2020, *arXiv:2011.04217*. [Online]. Available: http://arxiv.org/abs/2011.04217

[152] C. Song and A. Boularias, "Learning to slide unknown objects with differentiable physics simulations," 2020, *arXiv:2005.05456*. [Online]. Available: https://arxiv.org/abs/2005.05456
[153] J. Degrave, M. Hermans, J. Dambre, and F. Wyffels, "A differentiable physics engine for deep learning in robotics," *Frontiers Neurorobot.*, vol. 13, p. 9, Mar. 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2019.00006
[154] E. Heiden, D. Millard, H. Zhang, and G. S. Sukhatme, "Interactive differentiable simulation," 2019, *arXiv:1905.10706*. [Online]. Available: http://arxiv.org/abs/1905.10706
[155] M. Müller, M. Macklin, N. Chentanez, S. Jeschke, and T. Kim, "Detailed rigid body simulation with extended position based dynamics," *Comput. Graph. Forum*, vol. 39, no. 8, pp. 101–112, Dec. 2020, doi: 10.1111/cgf.14105.
[156] *Robotic Simulation | Unity*. Accessed: Feb. 24, 2021. [Online]. Available: https://unity.com/solutions/automotive-transportation-manufacturing/robotics
[157] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*. [Online]. Available: https://arxiv.org/abs/1606.01540
[158] J. Achiam. (2018). *Spinning Up in Deep Reinforcement Learning*. [Online]. Available: https://github.com/openai/spinningu
[159] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei, "SURREAL: Open-source reinforcement learning framework and robot manipulation benchmark," in *Proceedings of Machine Learning Research*, vol. 87, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., Aug. 2018, pp. 767–782. [Online]. Available: http://proceedings.mlr.press/v87/fan18a.html
[160] J. B. Mouret and K. Chatzilygeroudis, "20 Years of reality gap: A few thoughts about simulators in evolutionary robotics," in *Proc. GECCO Genetic Evol. Comput. Conf. Companion*, 2017, pp. 1121–1124.

JACK COLLINS was born in Brisbane, Australia, in 1995. He received the B.S. degree in mechatronic engineering from the Queensland University of Technology, Brisbane, in 2017, where he is currently pursuing the Ph.D. degree in robotics.

From 2016, he has been a Student with the Commonwealth Scientific and Industrial Research Organization, Brisbane. His research interests include the reality gap and methods to circumvent its effect, and evolutionary robotics.

SHELVIN CHAND was born in Lautoka, Fiji, in 1991. He received the B.Sc. and M.Sc. degrees from the University of the South Pacific, Suva, Fiji, in 2013 and 2014, respectively, and the Ph.D. degree in computer science from the University of New South Wales, Canberra, Australia, in 2018.

He is currently a Postdoctoral Fellow with the Robotics and Autonomous Systems Group, Commonwealth Scientific and Industrial Research Organization (CSIRO). His current research interests include evolutionary robotics and computational creativity.

ANTHONY VANDERKOP received the B.Sc. degree in physics and the B.E. degree in mechatronics engineering from the University of Queensland, in 2019. He is currently pursuing the Ph.D. degree in robotics with the Queensland University of Technology, Brisbane, Australia.

He has been a Student with the Commonwealth Scientific and Industrial Research Organization, Brisbane, since 2018. His research interests include improving the capabilities of mobile legged robots in rugged natural environments, particular those involving difficult terrain such as sand and snow.

DAVID HOWARD (Member, IEEE) was born in Grimsby, U.K., in 1984. He received the B.S. and M.Sc. degrees from the University of Leeds, U.K., in 2005 and 2006, respectively, and the Ph.D. degree from the University of the West of England, U.K., in 2011.

He has been with the Commonwealth Scientific and Industrial Research Organization, Brisbane, Australia, since 2013. His research interests include embodied cognition, the reality gap, and soft robotics.