CEA Comité Español de Automática

XLVI JORNADAS AUTOMÁTICA CARTAGENA 2025

UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Survey of Simulators for Deformable Objects in Robotics

Zafra-Navarro, Alberto*, Aragüés, Rosario, López-Nicolás, Gonzalo

Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, C/Mariano Esquillor s/n, 50018, Zaragoza Spain.

To cite this article: Zafra-Navarro, Alberto, Aragüés, Rosario, López-Nicolás, Gonzalo. 2025. Survey of Simula-tors for Deformable Objects in Robotics. Jornadas de Automática, 46.
https://doi.org/10.17979/ja-cea.2025.46.12171

Resumen

La simulación de objetos deformables sigue siendo un gran desafío en la robótica actual, yendo desde la necesidad de emular partes del cuerpo humano en cirujías autónomas, hasta el modelado de tareas de manipulación industrial. Esta breve revisión analiza los entornos de desarrollo actuales más usados para la simulación de deformaciones, mencionando y evaluando sus características más relevantes. En concreto se presentan y analizan ocho sistemas distintos, entre los que se encuentran NVIDIA Isaac Sim, MuJoCo y SOFA. Por otro lado, se describen en detalle tres librerías del entorno Unity mediante las cuales se pueden modelar objetos deformables. Esta guía ayudará a seleccionar las herramientas más adecuadas para sus respectivos proyectos.

Palabras clave: Robótica, Modelado, Simulación, Control de estructuras flexibles, Materiales flexibles e inteligentes.

Survey of Simulators for Deformable Objects in Robotics

Abstract

Simulating deformable objects remains a major challenge in robotics, going from the need to emulate parts of the human body in autonomous surgeries to the modelling of industrial manipulation tasks. This short survey provides a summary of the most commonly used environments for deformable object simulation, highlighting and evaluating their most relevant features. Specifically, eight different systems will be presented and analyzed, including NVIDIA Isaac Sim, MuJoCo, and SOFA. Additionally, three libraries from the Unity game engine that have been used to model deformable objects are described in detail. This paper provides a guide helping in the selection of the most suitable tools for their respective projects.

Keywords: Robotics, Modelling, Simulation, Control of flexible structures, Flexible and smart materials.

1. Introduction

Simulation plays a central role in robotics by providing safe, cost-effective, and repeatable virtual environments that accelerate development and reduce reliance on physical testing Chowdhury (2024); ROS2 Contributors (2025). Moreover, simulators generate large volumes of synthetic data critical for machine-learning approaches, such as reinforcement and imitation learning, that would be impractical to acquire from real-world systems Fernández-Fernández et al. (2024). In opposition to rigid-body frameworks, accurately modeling deformable objects, whose shape and volume vary under applied forces, remains challenging Jong et al. (2014). Thus, precise deformable-object simulation is vital, as it encompasses a wide range of applications, from robotic surgery to textile handling, e.g. Fig. 1 illustrates deformation of four shoe-sole geometries under diverse contact conditions using NVIDIA Flex in Unity.

This paper overviews the current state-of-the-art simulation platforms for deformable-object in robotics, evaluating their performance, fidelity, and integration features. By comparing their strengths and limitations, we provide guidance to researchers and developers selecting the most suitable simulation tools for their projects.

*Corresponding author: 876628@unizar.es

Figure 1: Shoe soles deformation utilizing NVIDIA Flex Unity's library. Spheres represent grasping points.

1.1. Simulation Challenges

Simulating deformable objects in robotics is intrinsically challenging due to the high dimensionality of their state space Huang et al. (2022). Rigid bodies can be modelled easily as they only require their position and orientation. In opposition, modelling flexible objects demands specific algorithms for capturing its properties, such as elasticity and plasticity Shan et al. (2024). Moreover, collisions may lead to unrealistic behaviours. Therefore, it is necessary to handle them carefully to ensure stability. However, this only can be achieved with highly demanding mathematical computations. Therefore, the computational requirements of these systems might often exceed researchers' resources Fernández-Fernández et al. (2024); Antonova et al. (2021); Va et al. (2023).

Simplifications resolve this issue, but they come along with a lower realism, increasing the "reality gap". The reality gap, is the mismatch between real and simulated world behaviour. This lack of accuracy complicates the policy transfer between simulations and physical systems ROS2 Contributors (2025); Arriola-Rios et al. (2020). Thus, developers face the compromise between accuracy and computational cost. Being necessary the creation of new techniques or hardware to solve this issue.

1.2. Previous Surveys

In this survey we analyze simulators for some tasks in robotics, such as the manipulation or transportation of non-rigid objects. The most related work, Collins et al. (2021) classifies the available frameworks based in its applications. Despite mentioning commonly known deformable objects simulators (e.g. SOFA, PyBullet, Isaac Sim, MuJoCo, Unity), its analysis in this area is more general than ours. Another relevant work is Kargar et al. (2024), which provides an impressive evaluation of simulators compatible with the ROS framework. Furthermore, it highlighs the trend of using game engines, such as Unity and Unreal Engine for more realistic emulation of real-world scenarios. Nevertheless, its focus differs from simulating flexible objects, unlike ours. Other relevant works, as Harris and Conrad (2011) and Shamshiri et al. (2018), provide a well detailed review of the diverse available systems, but are outdated. Finally, works specifically targeting deformable object simulation, Arriola-Rios et al. (2020) and Moore and Molloy (2007), rarely address the software ecosystem but focus more on a theoretical approach of the techniques used for emulating flexible objects.

Hence, it is clear that there is a need for a targeted survey of current deformable object simulation environments. By evaluating the usability, fidelity of deformable models, integration with robotic frameworks, extensibility, and performance, this survey aims to guide researchers and developers to the most suitable tools and to chart future trends in soft-robotics simulation.

1.3. Analysis Criteria

According to Afzal et al. (2020), many users of simulation software find certain characteristics burdensome, such as the reality gap, complexity of use, scenario construction, learning curve, computational costs, and reliability. Based on these issues, previous studies provide a limited overview of available simulators. Therefore, a further evaluation using the aforementioned specific criteria is necessary. Thus, each presented system would be analysed based on their general characteristics and their modelling capabilities. The general characteristics, presented in Table 1, covers non-functional, deployment and integration facets; and is characterized by the software licensing, the programming language used, its documentation quality or their minimal requirements. On the other hand, the modelling capabilities, shown in Table 2, specify the software attributes, such as the available pre-integrated 3D and robotic models, the supported model integration, the degree of realism of the system or even the mesh deformation related features.

2. Survey of Simulators

This section serves as a practical guide to the robotic engines used for simulating soft bodies. Although a wide range of systems offer this functionality, only the most commonly used ones have been listed in Tables 1 and 2. In Fig. 2, the evolution of their impact in the field of robotics is shown, by analyzing the number of papers available on the Web of Science (2025) since 2010 that use the presented frameworks.

NVIDIA Isaac Sim utilizes the Omniverse platform and GPU-based PhysX engine to deliver high-fidelity robotics simulations, including synthetic data generation, software-in-the-loop testing, and reinforcement learning using Isaac Lab Camargo et al. (2021); Fernández-Fernández et al. (2024). It natively supports deformable dynamics through PhysX soft-body Finite Element Method (FEM, see Cook (1994)) which uses volumetric tetrahedral meshes, realistic rigid–soft contacts, joint responses, and multiple sensors (vision, RADAR, LiDAR, contact, IMU) and has over 1,000 SimReady assets and extensive OpenUSD-based robot model sets.

Isaac Sim includes benchmark tasks, such as cloth folding and soft-object manipulation, with photorealistic rendering driven by NVIDIA hardware Blanco-Mulero et al. (2024); Mittal et al. (2023). Although it excels in elastic deformation, it lacks specific cloth or fluid-mesh solvers and plastic/fracture modeling NVIDIA (2025a). The simulator is available for free, but an Omniverse Enterprise license is required for a complete development and distribution. As an alternative, Isaac Lab is an open-source variant from NVIDIA, with less features and often used for robot learning and benchmarking.

Multi-Joint dynamics with Contact (MuJoCo) is a high-performance physics engine widely applied in robotics, biomechanics, computer graphics, and machine-learning research for its accurate and high-speed rigid-body dynamics. MuJoCo's flexible object simulation is not volumetric. Its

Table 1: General characteristics from the studied simulators.

Simulator	Open Source	Free	Programming Language	Minimal Requirements	Documentation Quality	APIs	Parallelization
NVIDIA Isaac Sim NVIDIA (2025b)	No	Yes (limited)	Python	**OS:** Windows 10/11 or Ubuntu 20.04/22.04; **CPU:** Intel i7 / Ryzen 5; **Cores:** 4; **RAM:** 32Gb; **Storage:** 50Gb SSD; **GPU:** RTX 3070; **VRAM:** 8Gb	★★★★★	ROS	Yes
MuJoCo Todorov et al. (2012)	Yes	Yes	Python C/C++	**OS:** Windows (x86-64 only), Linux (x86-64 and AArch64), Mac OS or Google Colab; **Processor:** 64 bit; **Python:** V 3.9+	★★★★★	ROS, Matlab Unity	Yes
PyBullet Coumans and Bai (2016)	Yes	Yes	Python	**OS:** Windows, Ubuntu, MacOS or Google Collab; **Python:** V 2.7/3.5.2	★★★★★	ROS, UDP, TCP	Yes
SOFA Coevoet et al. (2017)	Yes	Yes	Python C++/C#	**OS:** Windows, Latest Ubuntu LTS, MacOS 10.13.2+ and 9.1.0+; **C++17**	★★★★★	Matlab, Unity Unreal, Godot	Yes
iGibson Li et al. (2021)	Yes	Yes	Python	**OS:** Windows 10/11, Ubuntu 16.04+ or Mac OS X; **GPU:** NVIDIA; **VRAM:** 6Gb; **GPU Driver** >= 384; **CUDA** >= 9.0; **CuDNN** >= v7; **CMake** >= 2.8.12	★★★★★	ROS	Yes
Drake Tedrake et al. (2019)	Yes	Yes	Python C++	**OS:** Ubuntu 22.04/24.04 or MacOS 14/15; **Processor:** 64 bit; **Python:** V 3.10+; **C++20**	★★★★★	ROS (unofficial)	Yes
PlasticineLab Huang et al. (2021)	Yes	Yes	Python	**OS:** Windows, Ubuntu or MacOS; **Python:** V 3.+	★⋆⋆⋆⋆	Not Specified	Yes
Unity (without libraries)	No	For personal and student accounts	C#	**OS:** Windows 7/10/11, Ubuntu 18.04/20.04, Mac OS 10.13+ or CentOS 7; **Processor:** 64 bits	★★★⋆⋆	ROS, Matlab, Python	Yes

elastic objects are represented as a association of rigid bodies attached through spring-like connectors. This achieves a mass-spring behaviour that emulates volumetric deformable models Todorov et al. (2012).

MuJoCo has been developed in C/C++ with Python bindings Choi et al. (2021). It uses MJCF XML and URDF (further information about these formats can be found at Tola and Corke (2024)), object models Zakka et al. (2022) and exploits AVX-compatible CPUs and multithreading and XLA acceleration. After becoming open-sourced in 2022, some APIs in C and Python, as well as interfaces to platforms like Unity and CoppeliaSim have been developed. Despite having a limited soft-body fidelity, it remains as the main tool for cable and fabric manipulation researchs Shan et al. (2024).

PyBullet is a popular Python interface for physics simulation robotics. It offers a rapid rigid-body dynamics, collision detection, and a flexible object modelling, which is currently under development. This last feature supports FEM-based volumetric meshes and position-based mass-spring shells. Furthermore, it allows users to load soft bodies from `.obj` files and define material properties such as Lamé coefficients and damping factors Coumans and Bai (2016); Kao et al. (2024); Coumans and Bai (2025). Cloth is modelled with mass-spring (position-based) constraints, while volumetric soft bodies use embedded tetrahedral meshes. PyBullet can connect with ROS for sensor simulation and control programming. Additionally, it supports the top Machine Learning libraries for policy prototyping, and forms the basis of several soft-object manipulation benchmarks. Hence, PyBullet is a great, user-friendly solution in research as well as teaching settings Mower et al. (2022); Chowdhury (2024); Daniel Zakaria et al. (2022).

Simulation Open Framework Architecture (SOFA) is an open-source C++ library from INRIA, designed for interactive physics and soft-robotics simulation, with applications including medical training and deformable-tissue modelling Coevoet et al. (2017). It relies on volumetric FEM meshes solved with implicit or explicit integrators, for elastic and plastic deformation, and complements them with mass-spring cloth models, particle/grid-based fluids, plasticity, fracture, and cutting through plug-ins.

SOFA's modular framework enables interaction among rigid, elastic, plastic, fluid bodies, and user-defined actuators like cables and inverse-kinematics components. Cross-platform support is available through XML scene descriptions and Python bindings, and multithreading and CUDA-accelerated solvers enable scalable performance through a GPU plug-in Faure et al. (2025). Maintained by a lively community and large documentation, SOFA remains a top tool for high-fidelity soft-body simulation on medical applications, while also being a valuable framework for other applications.

iGibson is a Stanford Vision and Learning Lab open-source, Python simulation platform. iGibson uses the Bullet engine Coumans and Bai (2025) for high-speed visual rendering and rigid-body physics. It has been widely used to train and test robotic agents in realistic reconstruction-based 3D environments (e.g., offices, houses) Shen et al. (2021), iGibson Team (2025). Even though its physic engine is currently limited to rigid bodies and joints, a soft-body functionality is currently being actively developed. Moreover, iGibson 2.0 incorporates dynamic object attributes, temperature, wetness, cleanliness, and a VR interface for imitation-learning data gathering Li et al. (2021). Finally, iGibson can connect to NVIDIA Omniverse via OmniGibson to leverage top-of-the-line deformable-body and fluid solvers. This connection serves as the basis of the BEHAVIOR-1K benchmark of labeled, interactive domestic scenes NVIDIA (2025a); Li et al. (2024).

Drake is an MIT's Robot Locomotion Group open-source software C++ toolbox for optimization-based control and multibody dynamics. It is supported on Linux, Windows, and macOS with Python bindings (PyDrake) Tedrake et al. (2019). It integrates a rigid-body engine, system-composition framework, and a mathematical-programming toolkit, enabling fast prototyping and deployment on simulation as well as on hardware. While its deformable-body support is yet experimental, Drake applies FEM to tetrahedral meshes for volumetric elastic deformations, where rigid objects are mod-

elled with its well-established rigid-body solver. Drake main features include: a deep optimization toolkit (with trajectory optimization, model-predicted control and user-specified math programs), parallelism, a modular structure and an extensive documentation. Because this wide variety of capabilities, Drake is expected to be a general-purpose platform.

PlasticineLab. It is known that purely reinforcement-learning methods are highly dependent on the user defined rewards. This, sometimes can lead to trained less efficient agents, that focus on receiving the highest reward possible instead on having a high performance. On the other hand, some benchmarks have determined that gradient-based methods excel in short time-horizon problems. Therefore, there has been a trend on implementing hybrid methods, which combine both policies. These hybrid methods have proved that are optimal for complex problems. However, only differentiable simulators, such as DiffTaichi Hu et al. (2020), provide the most favourable environment to train hybrid agents. This is where PlasticineLab's stands out. PlasticineLab is an open-source, Python soft-body benchmark implemented on DiffTaichi and featuring support for differentiable physics, by offering exact gradient computation for planning and control optimization Dickson (2022); Huang et al. (2021)

PlasticineLab is compatible with Linux, Windows, macOS and with ML frameworks through its modular API, which comes with a set of predefined tasks (rolling, pinching, chopping, molding) centered on volumetric plastic deformation. Nevertheless, PlasticineLab is highly consuming in computational terms. Moreover, as it is still underdevelopment, its reduced community can limit its accessibility and support.

Game Engines, while not traditionally considered robotic simulators, are acknowledged for their high-quality graphics and realistic environments Kargar et al. (2024). Moreover, their adoption by researchers of the field is drastically growing. For instance, Unreal Engine serves as the foundation for leading autonomous vehicle simulators CARLA and AirSim. Unity's latest official ROS support has been revolutionary, to the point that some of the previously mentioned teams, as the ones from MuJoCo or SOFA, have implemented APIs to connect the innovative capabilities of Unity with their software.

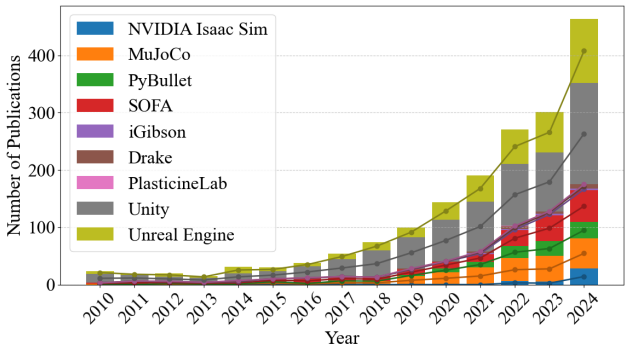


Figure 2: Evolution of the deformable simulators' popularity over time based on the number of papers referencing them.

3. Unity Engine

The use of game-engines, originally designed for entertainment, has gained relevance in robotics, offering high-performance and visually realistic simulation capabilities that tackle some of the traditional frameworks' limitations Jong et al. (2014). Unity, in particular, has exhibited more rendering fidelity and performance than conventional simulators Melo et al. (2019). Furthermore, the addition of ROS#, has facilitated its integration with robotics Siemens (2018). Nonetheless, the immense number of altenatives available further complicates the selection of the adequate library, and, thus motivates a review of the most promising Unity libraries for emulating non-rigid bodies.

Therefore in this section, we present the three most relevant (and freely available) Unity's libraries for simulating deformable object. Each library is briefly described, highlighting its main features and providing relevant references. Furthermore, they have been tested by stretching the mattress shown in Fig. 3(a) by applying an arbitrary and opposite force from the four object corners.

Unity Soft Body Deformation implements As-Rigid-As-Possible (ARAP) deformations, see Igarashi et al. (2005), in Unity via an anchor–manipulator framework Zgeb (2021). Here, a custom surface shader computes a transformation matrix, capturing translation, rotation, and scale between anchor and manipulator. Then, it displaces vertices within a sphere, while the normals are updated by transforming tangent and binormal vectors to preserve lighting accuracy. When coupled with physics-based springs, this setup yields a tunable, semi-realistic soft-body simulation. Moreover, as shown in Fig. 3(b), the deformation can be controlled by using a custom script that defines four anchors to stretch the mattress.

Deform is a CPU-multithreaded Unity library that offers a component-based system of modular "deformers" (e.g., bending, color masking, spherify) for both editor and runtime mesh manipulation. Deform has been optimized via Burst-friendly types (float3, float4×4) for high-efficiency vertex processing Woodall (2019, 2021). Deform is a highly customizable and accurate library, which allows implementing specific behaviours without altering the core code and supports GPU execution. In this case, the built-in magnet deformer was used to accomplish the aforementioned deformation task shown in Fig. 3(c). This deformer stretches the target mesh based on three properties: the distance from the anchor, the falloff, and the deformation strength.

NVIDIA Flex is a particle-based real-time library, which has been discontinued and integrated within NVIDIA PhysX engine. It employs a unified particle representation to simulate rigid, soft, and fluid interactions seamlessly Macklin et al. (2014); Macklin and Müller (2013). Despite its deprecation, Flex remains popular for deformable-object prototyping in Unity Tagliabue et al. (2020).

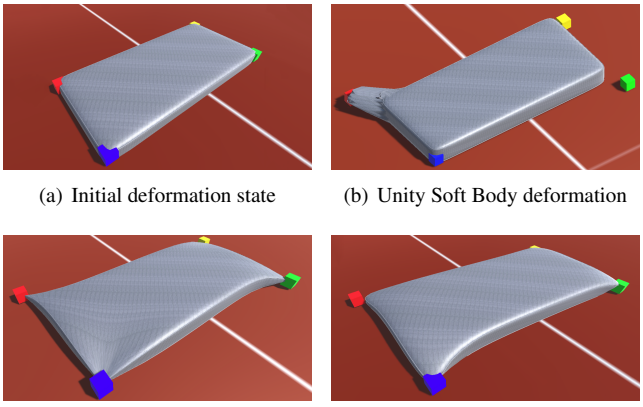
The library comprises three main modules: the *Flex Container*, which allocates and updates the particles via the solver; the *Flex Asset*, which organizes particles into weighted clusters (akin to skinning) to model materials by tuning spacing,

Table 2: Modelling capabilities of the studied simulators.

Simulator	Robot Models	3D Models	Model Import Formats	Desired Tasks	Degree of Realism	Soft-Bodies Represented	Mesh Deformation Type	Deformation Technique
NVIDIA Isaac Sim	Humanoids Manipulators Quadrupeds AMRs	Over 1,000 SimReady 3D assets	OnShape, URDF, MJCF, ShapeNet	Synthetic data generation AI training	★★★★★	Elastic	Volumetric	FEM
MuJoCo	Manipulators Dual Arms Drones End-effectors Mobile Arms Humanoids Quadrupeds	Not specified	URDF, MJCF, MJB, XML	Model-based control Machine Learning Biomechanichs Virtual Reality	★★★⋆⋆	Tendons, Ropes, Pseudo-Soft, Cloth	Superficial	Rigid-Bodies Linking (akin to mass-spring)
PyBullet	Humanoids Manipulators Quadrupeds	Not Specified	URDF, MJCF SDF, VTK, OBJ	Machine Learning Virtual Reality	★★⋆⋆⋆	Elastic, Cloth	Volumetric & Superficial	FEM & Mass-spring
SOFA	Not Specified	A set of deformable body parts between others	VTK, OBJ, GMESH, STL, OFF, DICOM	Soft Robotics Control Virtual Reality Medical Robotics	★★★★★	Elastic, Plastic, Fractures, Cloth, Fluids	Volumetric & Superficial	Multi-physics (FEM & Mass-pring)
iGibson	Humanoid Manipulators Quadrupeds Mobile Robots Quadrotor	A wide range of datasets including articulated objects, realistic scenarios, pedestrians and soft objects	URDF	Machine Learning Virtual Reality	★★★★★	Elastic, Cloth	Volumetric & Superficial	Bullet Engine
Drake	Manipulators	Not Specified, but some manipulation scenes are included	URDF, SDF, YAML	Multibody Dynamics Underactuated Robotics Trajectory Optimization Task & Motion Planning	★★★★★	Linear-elastic	Volumetric	FEM
PlasticineLab	None	Basic plasticine and a few objects	None	Machine Learning	⋆⋆⋆⋆⋆	Plastic	Volumetric	FEM
Unity	None	Basic Elements	URDF, OBJ, STL, FBX, DAE, DXF	Task & Motion Planning	★★★★⋆	Elastic, Cloth, Fluids	Volumetric & Superficial	Depends on the library

cluster radius, and link stiffness Coenen (2021); and the *Flex Actor*, which instantiates these assets in the scene, enabling either deformable or rigid behaviours.

Finally, by using the system in Tagliabue et al. (2020), a deformation mechanism was implemented to selectively manipulating particles within an object achieving realistic deformations, as shown in Fig. 3(d).



(a) Initial deformation state (b) Unity Soft Body deformation

(c) Deform deformation (d) NVIDIA Flex deformation

Figure 3: Examples of Unity's libraries capabilities showcasing a deformation of a elastic mattress-like object on the four object corners. Constant force is applied.

4. Conclusions

Based on the previous analysis presented in Section 2, it can be determined that NVIDIA Isaac Sim is a solution for GPU-accelerated and high-fidelity simulations, particularly suited for users with access to high computational resources. On the other hand, MuJoCo and PyBullet excel in reinforcement learning applications, especially due to their compatibility with platforms like Google Colab. Pybullet is an excellent alternative for cost-sensitive projects, as it offers a balance between accuracy and accessibility, while MuJoCo provides a higher degree of realism and excellent scalability. SOFA stands out as a choice for developers in the medical field, given its specialization in biomechanical simulations. In contrast, iGibson and Drake are better for users aiming for maximum realism, helping to bridge the sim-to-real gap. In fact, iGibson is particularly well-suited for machine learning projects, whereas Drake is better aligned with standard motion planning and control tasks. The recently developed differentiable simulator PlasticineLab is a strong option for users who prioritize efficient agent training over physical realism thanks to its built-in gradient computation capabilities. Lastly, game engines like Unity, while requiring additional plugins, offer unparalleled visual realism and are becoming increasingly viable for sim-to-real transfer. A future research line is to define a list of objective criteria to evaluate and compare the presented simulators.

Acknowledgment

This work was supported by the Government of Aragón, group T45_23R and via projects PID2021-124137OB-I00 and TED2021-130224B-I00 funded by MCIN/AEI/10.13039/501100011033, by ERDF A way of making Europe and by the European Union NextGenerationEU/PRTR, and via project REMAIN S1/1.1/E0111 (Interreg Sudoe Programme, ERDF).

References

Afzal, A., Katz, D.S., Goues, C.L., Timperley, C.S., 2020. A study on the challenges of using robotics simulators for testing `arXiv:2004.07368`.

Antonova, R., Shi, P., Yin, H., Weng, Z., Kragic, D., 2021. Dynamic environments with deformable objects, in: Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track.

Arriola-Rios, V.E., Guler, P., Ficuciello, F., Kragic, D., Siciliano, B., Wyatt, J.L., 2020. Modeling of deformable objects for robotic manipulation: A tutorial and review. Frontiers in Robotics and AI 7, 534750. doi:10.3389/FROBT.2020.00082/XML/NLM.

Blanco-Mulero, D., Barbany, O., Alcan, G., Colome, A., Torras, C., Kyrki, V., 2024. Benchmarking the sim-to-real gap in cloth manipulation. IEEE RA-L 9, 2981–2988. doi:10.1109/LRA.2024.3360814.

Camargo, C., Gonçalves, J., Conde, M., Rodríguez-Sedano, F.J., Costa, P., García-Peñalvo, F.J., 2021. Systematic literature review of realistic simulators applied in educational robotics context. Sensors 21, 4031. doi:10.3390/S21124031.

Choi, H.S., et al., 2021. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. Proc. Natl. Acad. Sci. U.S.A. 118, e1907856118. doi:10.1073/pnas.1907856118.

Chowdhury, G., 2024. Leveraging robotics simulation for safer, more efficient deployments. URL: https://www.abiresearch.com/blog/robotics-simulation-overview.

Coenen, S., 2021. Nvidia flex soft bodies graduation work digital arts and entertainment. URL: https://simoncoenen.com/downloads/flex_paper.pdf.

Coevoet, E., et al., 2017. Software toolkit for modeling, simulation and control of soft robots. Advanced Robotics 31, 1208–1224. doi:10.1080/01691864.2017.1395362.

Collins, J., Chand, S., Vanderkop, A., Howard, D., 2021. A review of physics simulators for robotic applications. IEEE Access 9, 51416–51431. doi:10.1109/ACCESS.2021.3068769.

Cook, R.D., 1994. Finite Element Modeling for Stress Analysis. 1st ed., John Wiley & Sons, Inc., USA.

Coumans, E., Bai, Y., 2016. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Coumans, E., Bai, Y., 2025. Bullet collision detection and physics library: Bullet documentation. URL: https://pybullet.org/Bullet/BulletFull/index.html.

Daniel Zakaria, M.H., et al., 2022. Robotic control of the deformation of soft linear objects using deep reinforcement learning, in: IEEE Int. Conf. on Automation Science and Engineering, pp. 1516–1522. doi:10.1109/CASE49997.2022.9926667.

Dickson, B., 2022. This deep learning technique solves one of the tough challenges of robotics. URL: https://bdtechtalks.com/2022/05/09/diffskill-robotics-deformable-object-manipulation.

Faure, F., et al., 2025. SOFA documentation. URL: https://sofa-framework.github.io/doc/.

Fernández-Fernández, J.A., Lange, R., Laible, S., Arras, K.O., Bender, J., 2024. Stark: A unified framework for strongly coupled simulation of rigid and deformable bodies with frictional contact, in: IEEE Int. Conf. on Robotics and Automation, pp. 16888–16894. doi:10.1109/ICRA57147.2024.10610574.

Harris, A., Conrad, J.M., 2011. Survey of popular robotics simulators, frameworks, and toolkits, in: 2011 Proceedings of IEEE Southeastcon, pp. 243–249. doi:10.1109/SECON.2011.5752942.

Hu, Y., et al., 2020. DiffTaichi: Differentiable programming for physical simulation. arXiv:1910.00935.

Huang, I., et al., 2022. DefGraspSim: Physics-based simulation of grasp outcomes for 3D deformable objects. IEEE RA-L 7, 6274–6281. doi:10.1109/LRA.2022.3158725.

Huang, Z., et al., 2021. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. arXiv:2104.03311.

Igarashi, T., Moscovich, T., Hughes, J.F., 2005. As-rigid-as-possible shape manipulation. ACM Transactions on Graphics 24, 1134–1141. doi:10.1145/1073204.1073323/SUPPL_FILE/PPS088.MP4.

iGibson Team, 2025. iGibson documentation. URL: https://stanfordvl.github.io/iGibson/index.html.

Jong, J.H.D., Wormnes, K., Tiso, P., 2014. Simulating rigid-bodies, strings and nets for engineering applications using gaming industry physics simulators. Int. Sym. on Artificial Intell., Robotics and Aut. in Space , 17–19.

Kao, F.C., Chen, Z.R., Shih, C.S., Lu, S.H., Lin, P.C., 2024. Bridging mechanical behavior differences of deformable soft objects in simulation and experiments using a data-driven model, in: IEEE Int. Conf. on Advanced Intelligent Mechatronics, pp. 1297–1302. doi:10.1109/AIM55361.2024.10637182.

Kargar, S.M., Yordanov, B., Harvey, C., Asadipour, A., 2024. Emerging trends in realistic robotic simulations: A comprehensive systematic literature review. IEEE Access 12, 191264–191287. doi:10.1109/ACCESS.2024.3404881.

Li, C., et al., 2021. iGibson 2.0: Object-centric simulation for robot learning of everyday household tasks. Proceedings of Machine Learning Research 164, 455–465.

Li, C., et al., 2024. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. Proceedings of Machine Learning Research 205, 80–93.

Macklin, M., Müller, M., 2013. Position based fluids. ACM Trans. Graph. 32. doi:10.1145/2461912.2461984.

Macklin, M., Müller, M., Chentanez, N., Kim, T.Y., 2014. Unified particle physics for real-time applications. ACM Trans. Graph. 33. doi:10.1145/2601097.2601152.

Melo, M.S.P.D., et al., 2019. Analysis and comparison of robotics 3D simulators, in: 21st Symposium on Virtual and Augmented Reality (SVR), pp. 242–251. doi:10.1109/SVR.2019.00049.

Mittal, M., et al., 2023. Orbit: A unified simulation framework for interactive robot learning environments. IEEE RA-L 8, 3740–3747. doi:10.1109/LRA.2023.3270034.

Moore, P., Molloy, D., 2007. A survey of computer-based deformable models, in: Int. Machine Vision and Image Processing Conf., pp. 55–66. doi:10.1109/IMVIP.2007.31.

Mower, C.E., et al., 2022. ROS-pybullet interface: A framework for reliable contact simulation and human-robot interaction. Proceedings of Machine Learning Research 205, 1411–1423.

NVIDIA, 2025a. Deformable-body simulation: Omniverse extensions. URL: https://docs.omniverse.nvidia.com/extensions/latest/ext_physics/deformable-bodies.html.

NVIDIA, 2025b. Isaac Sim: Robotics simulation and synthetic data generation. URL: https://developer.nvidia.com/isaac/sim.

ROS2 Contributors, 2025. Programming multiple robots with ROS 2. URL: https://osrf.github.io/ros2multirobotbook/.

Shamshiri, R.R., et al., 2018. Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison. Int J Agric & Biol Eng 11, 15–31. doi:10.25165/J.IJABE.20181104.4032.

Shan, J., et al., 2024. Soft contact simulation and manipulation learning of deformable objects with vision-based tactile sensor. arXiv:2405.07237.

Shen, B., et al., 2021. iGibson 1.0: A simulation environment for interactive tasks in large realistic scenes, in: IEEE/RSJ IROS, pp. 7520–7527. doi:10.1109/IROS51168.2021.9636667.

Siemens, 2018. ROS#. URL: https://assetstore.unity.com/packages/tools/physics/ros-107085#description.

Tagliabue, E., et al., 2020. Soft tissue simulation environment to learn manipulation tasks in autonomous robotic surgery, in: IEEE/RSJ IROS, pp. 3261–3266. doi:10.1109/IROS45743.2020.9341710.

Tedrake, R., et al., 2019. Drake: Model-based design and verification for robotics. URL: https://drake.mit.edu.

Todorov, E., Erez, T., Tassa, Y., 2012. Mujoco: A physics engine for model-based control, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 5026–5033. doi:10.1109/IROS.2012.6386109.

Tola, D., Corke, P., 2024. Understanding URDF: A dataset and analysis. IEEE RA-L 9, 4479–4486. doi:10.1109/LRA.2024.3381482.

Va, H., Choi, M.H., Hong, M., 2023. Efficient simulation of volumetric deformable objects in Unity3D: GPU-accelerated position-based dynamics. Electronics 12, 2229. doi:10.3390/ELECTRONICS12102229.

Web of Science, 2025. URL: https://www.webofscience.com.

Woodall, K., 2019. Deform. URL: https://assetstore.unity.com/packages/tools/modeling/deform-148425.

Woodall, K., 2021. Deform: Documentation. URL: https://github.com/keenanwoodall/Deform/wiki.

Zakka, K., Tassa, Y., Contributors, M.M., 2022. Mujoco menagerie: A collection of high-quality simulation models for mujoco. URL: http://github.com/google-deepmind/mujoco_menagerie.

Zgeb, B., 2021. Unity softbody deformation: An example of mesh deformation in Unity. URL: https://bronsonzgeb.com/index.php/2021/07/10/mesh-deformation-in-unity/.