Bridging the Simulation to Reality Gap in Robotics

Konstantinos Vasios

School of Computer Science and Electronic Engineering

University of Essex

This dissertation is submitted for the degree of

Doctor of Philosophy

November 2024

I dedicate this thesis to all the beings with whom I have ever interacted, directly or indirectly, for both the good and the bad.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 80,000 words including appendices, bibliography, footnotes, tables and equations.

Konstantinos Vasios

November 2024

Acknowledgements

I would like to thank my supervisors Dr. Panos Chatzakos, Dr. Vishwanathan Mohan, and Dr. Delaram Jarchi for their vision and support throught the PhD journey.

This publication was made possible by the sponsorship and support of Lloyd's Register Foundation (LRF). The work was enabled through, and undertaken at the National Structural Integrity Research Centre (NSIRC), a postgraduate engineering facility for industry-led research into structural integrity established and managed by TWI through a network of both national and international Universities.

Abstract

Recent advances in Machine and Reinforcement Learning, particularly in visuomotor control policies for robotics, have increased reliance on simulation frameworks and physics engines. These tools generate synthetic data and create sandbox environments to meet the substantial data demands of neural network training. However, **given the inherit discrepancies between simulation and reality, the Simulation to Reality (Sim2Real) Gap in Robotics refers to all factors and specialized techniques that affect a transfer of an agent from the simulation to the real-world**.

Our literature review revealed that this field is largely empirical, fragmented across the robotics landscape, and heavily influenced by technical aspects of visuomotor policy design. To address this, our methodology covers the Sim2Real domain comprehensively, establishing performance metrics, identifying Reinforcement Learning design considerations, and developing a taxonomy of specialized Sim2Real techniques. We also create a detailed taxonomy of available simulation frameworks and physics engines for robotics.

The next phase of our research focuses on mushroom harvesting, an unsolved problem in industrial food automation. This interdisciplinary challenge involves complex kinodynamic task and motion planning under constraints and environment uncertainties related to deformable bodies and material failure modes.We develop a practical Sim2Real pipeline for mushroom harvesting using a robotic gripper, **allowing us to evaluate several Sim2Real techniques, including system identification with modeling approximations and explicit transferable abstractions**. **Contrary to conventional Sim2Real approaches, we show**

that the simulation framework is not just a tool for training but should be an integral component of the perception and planning system. This is a key statement of the thesis, demonstrating the predictive power of simulation in real-world applications.

Our concluding remark and future work directions, based on the experience gained during this work, point to **a holistic point-of-view for active inference, where the robotic agent actions are point towards an active, life-long, real-world model discovery.**

Table of contents

List of Figures

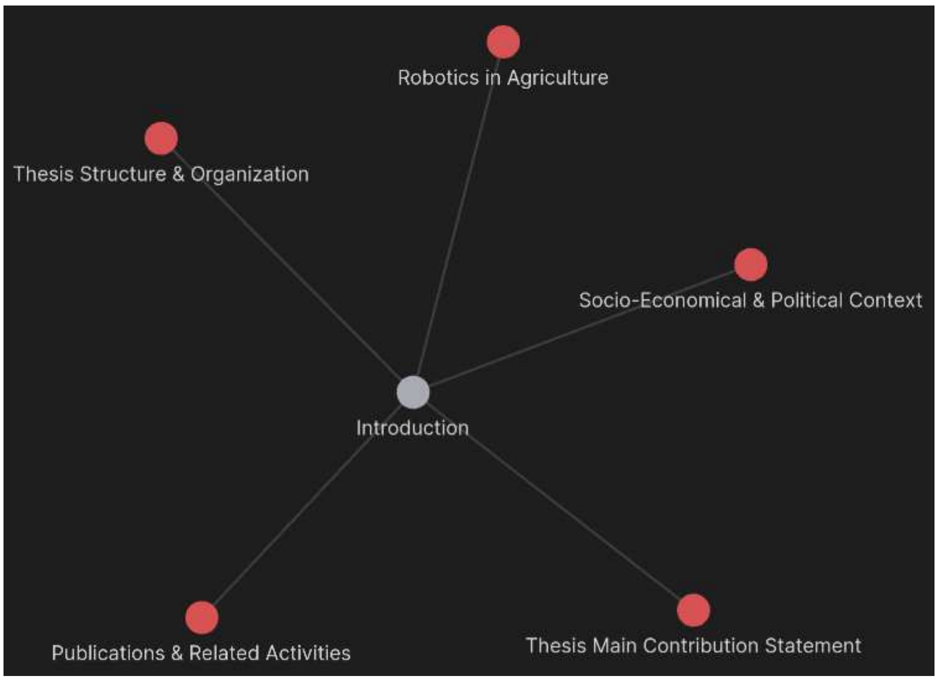
List of Tables

Chapter 1

Introduction

"Reality exists in the human mind, and nowhere else."

—George Orwell, *1984*


Robotics in Agriculture
Thesis Structure & Organization
Socio-Economical & Political Context
Introduction
Publications & Related Activities
Thesis Main Contribution Statement

1.1 Socio-Economical & Political Context

1.1.1 From Cybernetics to AI

The Sim2Real problem, as it currently viewed by the community, refers exclusively in the adaptation of an Reinforcement-Learning (RL) scheme for training an agent, most usually in the form of a Neural-Network in Simulation and a consequent deployment to a real-world equivalent robotic hardware platform.

It could be argued thought, that this problem, in a more generic formulation, is not exactly a new problem and a very extensive and advanced scientific and technological framework has already been developed within the control & systems engineering raum over the last century. Robust controller design and system identification techniques combined with the so called model-based design approaches, are very well researched and understood concepts and have been widely adopted in a wide variety of systems, ranging from space exploration vehicles to military applications, modern vehicles, telecommunications etc.

It is very interesting to observe the changing nature of "machine intelligence" over the technological eras, starting with terms like, cybernetics, regulators, "smart systems", system engineering till the current AI umbrella term.

It would be then naturally assumed that the Sim2Real problematic, would be a natural evolution and it should probably then be derived directly of the system engineering and control disciplines. The fundamental components, such as Reinforcement-Learning, Neural-Nets, mathematical optimization, stochastic-control, adaptive-control where all there.

It would seem only natural for this discipline of system engineering and control to give birth to the "modern AI", where instead of controllers we are now talking about agents, instead of robust control and system identification we talk about Sim2Real etc.

The question of how we jumped from “cybernetics” and “control” to “agents” and “generative AI” is a very interesting one and has to do with a variety of factors with the dominant being one: data.

Another important factor that needs to be added, lies in the philosophy of approaching things. Control Engineering is deeply rigorous, analytical and thoroughly heavily relies on mathematical proof. This allows for concrete definitions of robustness, stability and safety margins which in turn allows for rockets and missiles to be launched, vehicles and communications to be operable with safety. But it was known that there are limitation. Severe limitations.

Especially in the field of robotics, and some of it’s hardest fields like e.g robotic manipulation there have been prominent voices in the community raising the need for a paradigm shift as in the advent of the 21st century the robots still were missing, from physical or man-made disasters, helping in our every day lives etc.

Real-world perception through vision which should result in true autonomy in planning and control for systems that are physically interacting with the world were outside of reach, as the DARPA robotics challenge has proven. Analytical, manual-feature engineering for perception and robotic manipulation is proven to be an intractable problem to formulate.

Control theory has served perfectly well the cold-war era economy, has successfully lifted humanity to space and the world-wide telecommunication network. It has also fostered the building blocks that would later shape the “modern AI” era.

The paradigm shift has already been set in action but from a unexpected source.

1.1.2 YouTube & Cat Videos - Data Feature Shaping

The internet boom of the 2000s in combination with smartphone boom after 2007 has given rise to a new economy and a new set of players, such as Google & Facebook which dominated the new landscape.

Computation, due to developments in data-centers and the smart phone revolution became cheap and therefore accessible. This also in combination with the gaming industry gave rise to accessible powerful parallel GPU processing.

Internet seems to be acting as world-wide data accumulator and this has not been left unnoticed as phrases like “data is the new gold” started to become commonplace.

It became apparent that a great deal of value could be potentially hidden in unstructured, unlabeled data stored in data-bases, which would be really expensive and slow to manually extract.

Some sort of intelligent system should be able to automatically perform pattern precognition to these unlabeled data and extract valuable insights.

Google was at the forefront of this research front and in 2012 shook the research community by training a neural-network, in an unsupervised manner, to auto-decipher the concept of a “cat” by just “watching” thousands upon thousands of you-tube videos. This was a pivotal moment.

Another pivotal moment was the seminal worked on computer vision, which managed to train a deep layered neural network architectures, the now extremely popular convolutional neural network (CNN). Deep layering, as a measure to increase performance is challenging to a variety of technical issues, with the most prominent the problem of vanishing gradients. But the influence of this work was monumental, as it demonstrated the ability to automatically,through training, shape a hierarchical set of features which vastly outperformed the previous-state-of-the-art, which was based in the traditional manual-feature engineering.

The departure from “manual-feature-engineering” was a tremendous breakthrough, and it’s in sharp contrast to the more traditional ways of working for the control engineering and pattern recognition communities. Allowing the data to shape the feature space completely within the context of a neural-network architecture was major shift.

1.1.3 Deep-Layers and Robots

Naturally, now that we are able to train deep-layered architectures the question is how can we apply the lessons learned in optimal-control design for optimal decision making given a set of measurements which represent the current state of the system and the world.

Optimal Control Theory and Reinforcement-Learning offer a rigorous mathematical definition and frameworks for the design of such agents. Traditionally though a complete end-to-end approach by a naive application of these methods is proven to be intractable, especially when dealing with high-dimensional inputs such as cameras and other multi modal sensors within complex environments.

The seminal DQN paper from, now Google's, Deepmind with the seminal DQN paper made the breakthrough and managed to train end-to-end an agent (or previously controller) for competitively playing Atari games, directly from pixel readings to controller discrete actions. The introduction of CNNs to process raw pixels in combination with techniques such as Experience Replay, Fixed-Q Targets and Reward Clipping was critical factor for the success.

This line of work naturally then raised the question on whether this might be the so wanted paradigm shift for robotics and attempts for adapting this end-to-end framework for continuous-control started shortly after. Early attempts focused in continuous control for a variety of kinematic structures only in simulation.

The watershed moment on continuous control for robotics with RL based on a deep network architecture came in 2018 from OpenAI and the Dactyl project which managed to train an agent in simulation for manipulating a Rubik's cube in 3D space with the help of a Shadow Robot Hand with a significant rate of success when deploying to the real-world setup. Quite extensive domain & dynamics randomization was utilized during training, e.g changing the colors, textures as well as a set of physical parameters and the application of random forces on the object forcing the agent to focus on an esoterically meaningful set of features and thus increasing the chances for a successful transfer on the real-world setup.

This is fundamentally the line of work that gave essentially birth to a modern line of work for robotics and the quest for Sim2Real. Although the Domain randomization received the attention of the community as the predominant technique for Sim2Real, only limited attempts have been made for a rigorous and analytical mathematical-statistical definition and still empirical approaches dominate the field.

The standard workflow has been set and a very extensive line of research work tried since to extend and improve this framework in robotics.

Apart from the Domain & Dynamics randomization a set of specific techniques have been introduced specifically for Sim2Real as it seems on the surface that this might be the most significant bottleneck for transferring the impressive agent performance achieved in simulation to the real-world.

Although some great research results have been presented, based on this line of work there are quite a few of severe limitations that hinder wide-adaptation and eventually broad commercialization in industrial and consumer settings.

1. The black-box nature of the agent's architecture and the consequent explainability problem.
2. Still expensive to train in simulation.
3. The simulation its self is not an easy task to develop. This is rarely discussed and usually it is taken as a given that it simply "exists." In some cases it might be more difficult to develop manually than the controller its self.
4. Lack of a hard-guarantee on safety margins. The long-tail distribution problem.

1.1.4 Robot dance is (largely) not AI

During the same time and as this revolution is taking place in continuous control for robotics with deep RL, some equally or more impressive results in robotics control are coming from the robots of Boston Dynamics, which seem able to perform impressive locomotion and acrobatics and even more impressively build an available commercial robot platform with the spot robot dog.

This is though, it it's core, not RL but rather a very advanced and very well fine-tuned of Model Predictive Control (MPC) which is derived from the more traditional control engineering line of research.

This level of locomotion and the commercialization of a usable and useful robotic platform seemed a bit far for the aforementioned RL approaches.

The Model Predictive Control approach is still a very active area of research and still receives significant attention by the research community, despite the AI dominance.

MPC has still some significant drawbacks and limitations:

1. Requires a model of the system! This might be hard to obtain and errors might be detrimental to performance.
2. Computational complexity and Real-Time constraints.
3. Tuning & Sensitivity to parameters.
4. Cost function design.
5. MPC is fundamentally a local planner. It requires a high-level planner to achieve true "intelligence".

Boston dynamics seems though to be diving deep into the realms of Reinforcement-Learning space, as the investments strategies on the AI Institute of the company reveal in combination with actual recent research results for the robot dog and it's advanced RL based robust control strategies.

1.1.5 Large Language Models (LLMs) have entered the chat.

Since the public release of ChatGPT Large Language Model by OpenAI the world of technology & business is phasing a monumental shift. LLMs trained on vast corpora of text data, possess a deep contextual understanding and can perform a wide range of language-related tasks changing the fundamental nature of human labor in almost all it's facets.

The core breakthrough of these models lies in the adaptation of the Transformer Models [1] in a very large scale which in turn allow a contextual multi-scaled semantic "understanding" of words-sentences.

Since the introduction ChatGPT a race on harnessing the predictive power of generative models in robotics has been initiated in various and creative ways, as an attempt to mitigate/alleviate well-established bottlenecks within existing working frameworks.

For example in the RL context, the reward function shaping is notoriously hard task if done manually. A great line of work tries for a direct exploitation of the existing LLMs as-they-are. Interpreting LLMs as high-level semantic planners and by exploiting their code writing capabilities they can be for example utilized directly for reward function shaping [2]. A direct generation/synthesis of code containing an action policy for a robotic task has been attempted as well [3].

Another already known and existing bottleneck, is the lack of the vast amount of data required for training potential foundation models in robotics. Auto-generation of synthetic data in simulation [4] is another potential use-case of LLMs in robotics, as the generation of well established formal and internet-scale data-sets would potentially allow for a "ChatGPT" moment in robotics [5].

As the "pixel2torques" problem in robotics is a sequence optimal decision making problem, the core LLMs model architecture could be direly repurposed for predicting actions instead words, as it is the case with Vision-Language-Action models [6].

A great topic of discussion regarding LLMs lies on the kind of internal representation and on whether they build some kind of “world-model”, as a prediction capability on the world state would allow for explicit and accurate reasoning beyond a pure statistical inference [7]. Sparks of the effectiveness of LLMs as global planners have already demonstrated [8] where a task decomposition into simpler admissible tasks is achieved.

LLMs seem to have strongly impacted the world of robotics as well, and have sparked an impressive research trend with significant results in a very short period. We only can expect this trend to exponentially grow over the following years.

1.1.6 The race for an “iPhone moment” for robotics. The era of humanoid robots.

In parallel to the currently ongoing LLM “revolution” we observe another strong trend in robotics with an unprecedented surge in interest and investment in humanoid robots in recent years. Tech companies like Tesla, Amazon, Google, Nvidia and Microsoft and research institutions worldwide are in a race to develop advanced humanoid robots capable of performing tasks that closely mimic human actions and interactions. This race is fueled by the potential of humanoid robots to revolutionize industries such as healthcare, manufacturing, customer service, and personal assistance. Major technology corporations like Tesla, Honda, and Boston Dynamics are at the forefront of this innovation wave, pouring significant resources into the development of robots that exhibit human-like dexterity, mobility, and social interaction capabilities.

This level of investment and the subsequent economies of scale, can be expected to democratize the availability of highly capable humanoid platforms to a broad range of researchers & technologist which in turn will further accelerate the pace of research in robotics intelligence. The Humanoid Robot by Unitree Robotics, currently available at $16K is a testament to this ongoing revolution in the field. The efforts of Agility Robotics, Apptronik, Sanctuary AI to explore the rise of these AI-driven humanoids are also worth mentioning.

1.1.7 Overall remarks

From the ancient "automaton" empirical machines utilized for the emperor's amusement, till Watt's centrifugal regulator, from the first PID controller application in the USA navy in late 19th century till the modern jet fighters, the space age and the modern control theory, from the Antikythera mechanism till Baggage's "difference engine"-automatic mechanical calculator, from the ENIAC in WWII till the World-Wide-Web and the Apple's iPhone, from Google till ChatGPT and from Unimate robot to Atlas from Boston Dynamics, we are standing in a point in history where we, for the first time, might be reaching a singular point in the field of robotics.

This has an ultimate disruptive potential as human-labor has been at the core of economy and civilization since the agricultural revolution. A careful human-centric sensible political approach should allow for a universe of abundance where the auto-generated wealth should be democratized.

Lastly, and probably most importantly, a great opportunity is presented, where in our attempt of building a an artificial human, we can vastly expand our understanding of the actual human being. At the heart of Large Language Models, World-Models, Representation Learning, Vision, Planning and control, namely the "pixel2torques" problem we can draw parallels and find the right sets of questions on the greatest mystery of them all, that of human consciousness (qualia).

1.2 Robotics in Agriculture

Agricultural robotics is a rapidly advancing field in academia-research and commerce due to multiple synergistic effects on recent scientific and technological breakthroughs, most notably in perception and planning, in combination with hardware cost and availability.

The continued advancement of robotics in agriculture offers solutions to pressing issues, including population growth, increasing urbanization, and labor shortages. Robotics are being explored as a method of increasing agricultural productivity and meeting the increasing global demand for food. The integration of robotic systems in agriculture aims to enhance efficiency, productivity, and sustainability by automating various tasks and minimizing reliance on manual labor, which has been especially impacted by recent global health crises such as the COVID-19 pandemic. This transition towards automation is driven by the need to adopt advanced technologies that improve working conditions for farmers, addressing issues like musculoskeletal disorders linked to traditional farming practices. Precision agriculture, a growing trend that promotes sustainable practices and reduces the carbon footprint of farming, is a key beneficiary of these advancements. [9, 10]

1.2.1 The Low Hanging Fruit for Robotics Adaptation in Scale

Harvesting robots, particularly those designed for crops like strawberries and tomatoes, are moving closer to commercialization. These robotic solutions often combine a mobile platform, a robotic arm for picking, and sophisticated vision systems for identifying and locating ripe produce. This progress is driven by the labor-intensive and time-sensitive nature of harvesting, making automation highly desirable to address labor shortages and boost productivity. However, challenges remain in adapting these systems for crops with more complex shapes and growth patterns, like grapes, which still lack widely available robotic harvesting solutions. Despite this, research into vineyard robots is ongoing, focusing on tasks like monitoring, pruning, and spraying. The development of standardized and modular

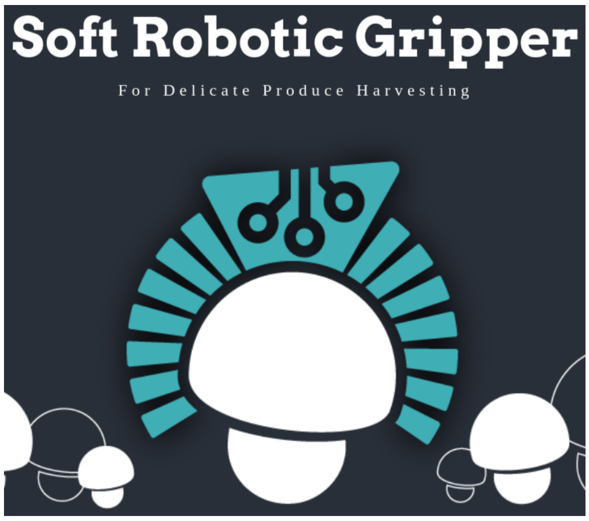


Figure 1.1 The SoftGrip Project logo.

robotic platforms is further driving commercial prospects by allowing for adaptation to different agricultural tasks beyond just harvesting, such as UV treatment and weeding [9].

1.2.2 Robotic Mushroom Harvesting

The task of mushroom harvesting is an exemplary case for demonstrating advanced perception, long-horizon planning, and dexterous robotic manipulation capabilities, as it simultaneously involves scene decluttering based on long-horizon Task and Motion Planning (TAMP) with forceful manipulation and grasp synthesis for purposeful fracturing, given a set of cost objectives and operational constraints necessary for an optimal crop yield. As is usually the case to avoid bruises in the case of grasping and manipulation of delicate produce, soft grippers are necessary, which further increases the complexities around perception, modeling, and control.

1.2.3 The SoftGrip EU Research Project

The SoftGrip Project [11] is an EU Research Project in which the University of Essex is a partner responsible with the key task of developing an imitation learning pipeline for autonomous mushroom harvesting.

Overall the project aims to create a soft robotic gripper for the delicate harvesting of produce, with a particular focus on the mushroom farming sector. The gripper is designed to address the need for automation in this industry, aiming to increase production and reduce labor costs for European SMEs. The project focuses on developing a gripper that is economically viable, scalable, and environmentally friendly. The gripper design evolves around soft, self-repairing, food-safe, and recyclable materials, ensuring both safety and sustainability. Intrinsic sensing will be embedded directly into the soft material of the gripper, enabling it to interact with delicate produce without causing damage.

The SoftGrip project has been of paramount importance to the creation of this thesis, as it has given us the opportunity to work in the forefront of research, among high-profile institutes and some of the most prominent researches in the field. Lastly it has allowed us to experiment and test against a real-world, high-stakes industrial scale problem which in turn allows for some greatly valuable insights on what kind of state-of-the-art research and methodology is practically applicable and with a potential for commercialization.

1.3 Thesis Structure & Organization

1.3.1 Chapter 2

From early on we fundamentally establish that Sim2Real in robotics is quite empirically practiced, with a quite scattered methodological approach, with sometimes seemingly contradictory research outcomes, and finally without a concrete, well framed and established ontological foundation.

We do therefore attempt in **Chapter 2** through a rigorous literature survey to:

1. establish a complete taxonomy of all the Sim2Real methodologies already practiced in robotics to date.
2. establish a coherent Sim2Real framework based on the compiled taxonomy, which should utilized as a road-map in the field.
3. identify the key open questions and the limitations on the current methodologies.
4. establish connections between preexisting and coexisting concepts, like the traditional Control Theory, Model-Based Control and the more modern Generative AI, Diffusion Models, LLMs etc.
5. as the Simulation Environments and the Physics Engines lie at the core of Sim2Real, based on most often used RL workflows, for synthetic data generation, training and testing, we attempt a complete listing of all the existing simulation frameworks for robotics.
6. lastly attempt a take on future research work and how the field could progress and evolve in the years to come.

1.3.2 Chapter 3

At the core of Sim2Real workflow lies the development of the Simulation Environment as the "Digital Twin" of real-world robotics task at hand. The development of of simulation environments comes with it's own unique set of challenges, which are quite often overlooked within the context of Reinforcement Learning methodologies for robotics control.

As already mentioned, our selected task at hand, as a demonstrator and a test-bench against the techniques and methodologies developed within the context of this dissertation, is the one of automated mushroom picking with a robotic gripper.

At **Chapter 3** we do therefore proceed with analytically explore the development of a simulation framework for mushroom harvesting with a robotic gripper, with the pose of the research question on whether such approximations in simulation can suffice for such a complex robotic manipulation task.

The key elements of this work are:

1. Utilizing a rigid multibody physics engine (pybullet) for approximating the mushroom-root system elastic deformations and failure mode with a first-order approximation of continuous mechanics and the Von Mises Failure Mode adapted to the anisotropic properties of the mushroom-root material.
2. We proceed to an experimental methodology for determining the simulation parameters through system identification as a methodology for Real2Sim.
3. We showcase a real-world equivalent setup for deploying and testing to the real-world.
4. Based on the same principles we develop a solution for the simulation of a soft robotic gripper.
5. We demonstrate the limitations of a hand-crafted approach to the development of "Digital-Twins" and we propose a road-map for future solutions.

1.3.3 Chapter 4

A core statement of our Thesis is that the Simulation Environment is an asset that traditionally is under-utilized for agent training. We believe that the predictive power of the simulation engine should be further utilized for solving the Sim2Real gap as a core component of the perception and control pipeline.

We do therefore utilize the concept of the Model Predictive Path Integral (MPPI) control for harnessing the "digital twin" within the perception-planning-control pipeline.

Specifically for the task of mushroom harvesting with a robotic gripper we experimentally prove that this method is able to autonomously generate optimal and identifiable picking strategies, which are then able to reliably transfer to the real-world though selected transferable abstractions.

1.3.4 Chapter 5

Lastly in the closing chapter we present a summary with the potential extensions of the current line of work, as well as a possible research direction for Sim2Real based on latest research trends.

1.4 Thesis Main Contribution Statement

As mentioned in the previous section, the Sim2Real field is applied empirically without a concrete framework as a reference guide for the researchers and engineers.

Our literature survey on Sim2Real contributes therefore to the following main aspects:

- A complete taxonomy on the Sim2Real techniques already utilized in the field.
- A complete taxonomy on the simulation frameworks and physics engines available for the development of digital twins.
- A core statement on the case of Real2Sim2Real and the case of lifelong learning as a solution to the automatic 3D world model reconstruction problem, as an equivalent reformulation of the Sim2Real problem.

Fundamentally our Thesis challenges the standard assumptions regarding the typical Sim2Real workflow with Reinforcement-Learning for Robotics Control. We argue that the simulation-modeling environment is a valuable asset that holds some strong predictive value on the fundamental dynamics of the scene and it should be therefore further harnessed and utilized as an integral part of the perception-planning-control stack.

Based on that core statement our contribution is based around the following main elements:

- A dynamic simulation framework for mushroom crop harvesting with a soft or rigid robotic gripper with first-order continuum mechanics approximations, with parameter identification based on real-world mushroom characterization (SysId).

- A Real2Sim 3D scene reconstruction with embedded physics based on mushroom pose estimation and the aforementioned simulation framework.
- An offline planner based on the Volumetric Grasping Network (VGN) for mushroom selection and MPPI for generating optimal mushroom picking primitives, which despite being a local planner subject to potentially stuck in local minima, is able to generate composite picking actions that correspond well to empirical evidence.

1.5 Publications & Related Activities

The following conference papers have been **accepted** and they will **presented** at the respective conference.

- **K. Vasios**, A. Porichis, V. Mohan and P. Chatzakos "**Robotic Mushroom Harvesting with Real2Sim2Real and Model Predictive Path Integral (MPPI) based Planning**" ICRA 2025

The following conference papers have been published as part of the current PhD Thesis.

- A. Porichis, **K. Vasios**, M. Iglezou, V. Mohan and P. Chatzakos, "**Visual Imitation Learning for robotic fresh mushroom harvesting**," 2023 31st Mediterranean Conference on Control and Automation (MED), Limassol, Cyprus, 2023, pp. 535-540, doi: 10.1109/MED59994.2023.10185745.

The following journal papers have been submitted and are currently under review as of the writing of this Thesis.

- **K. Vasios**, V. Mohan and P. Chatzakos "**Real2Sim2Real in Robotics: A literature review on active perception**".

A presentation & a panel discussion have been delivered in

- agROBOfood conference in the Agricultural University of Athens, GR

representing the University of Essex as partners in the EU SoftGrip project.
Lastly a visit in

- ICRA23' London, UK conference

has been paid in it's full duration with a participation in the following workshops:

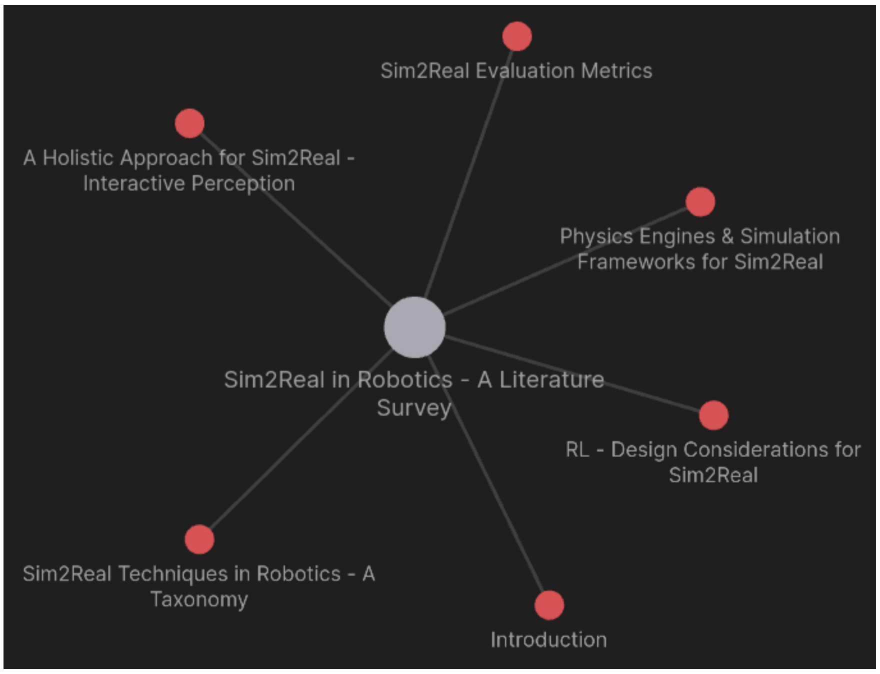
- 3rd Workshop on Representing and Manipulating Deformable Objects.
- Embracing contacts. Making robots physically interact with our world.

Chapter 2

Sim2Real in Robotics - A Literature Survey

"How could they see anything but the shadows if they were never allowed to move their heads?"

—Plato, *The Republic, Allegory of the Cave*


Sim2Real Evaluation Metrics
A Holistic Approach for Sim2Real - Interactive Perception
Physics Engines & Simulation Frameworks for Sim2Real
Sim2Real in Robotics - A Literature Survey
RL - Design Considerations for Sim2Real
Sim2Real Techniques in Robotics - A Taxonomy
Introduction

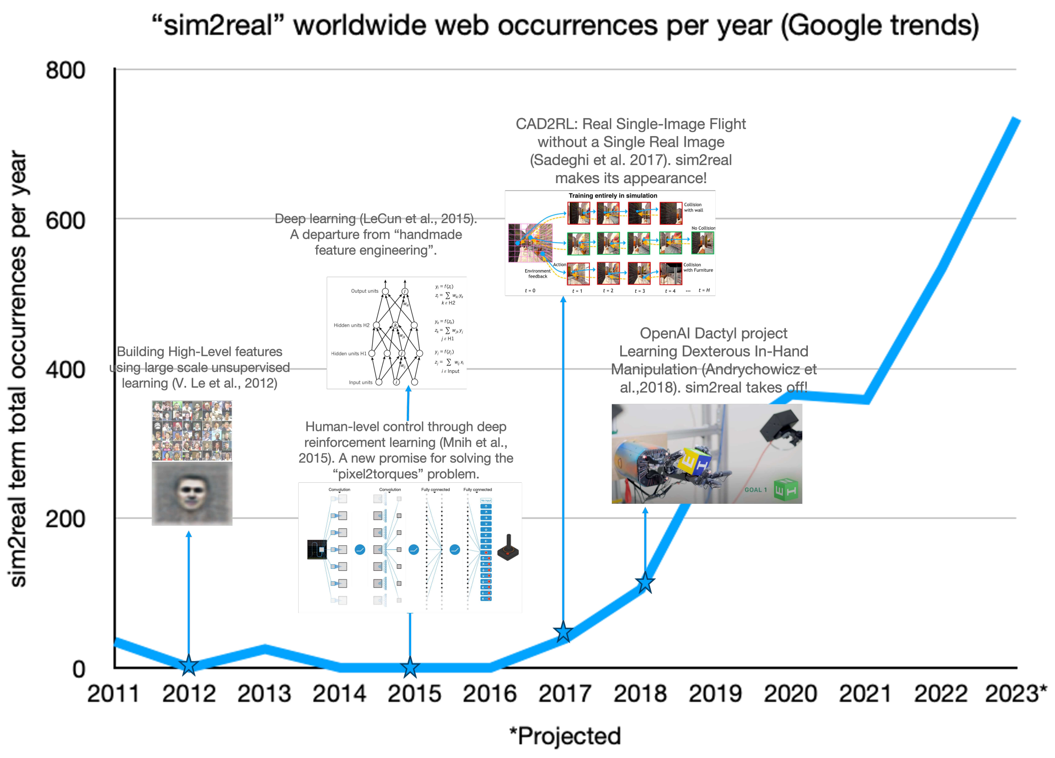


Figure 2.1 "sim2real" term occurrences in Google Trends with the associated standout contributions in the field [12–16]

2.1 Introduction

2.1.1 Methodological Approach

We start our work with a literature survey on the topic of "bridging the simulation to reality (Sim2Real) gap in robotics" by first identifying a key set of relevant, already preexisting, literature reviews and surveys. We end up with a set of 4 main lines of work [17–20] that directly elaborate on the topic of Sim2Real.

Based on these aforementioned lines of work, as well as our overall broader study on on the topic, it became quickly apparent that the field is still mostly empirically applied, as the Sim2Real problem seem to be treated as a subproblem in the general Reinforcement Learning for control in robotics. It therefore still lacks an established, commonly accepted, well understood and formalised epistemic definition in conjunction to the broader categories and

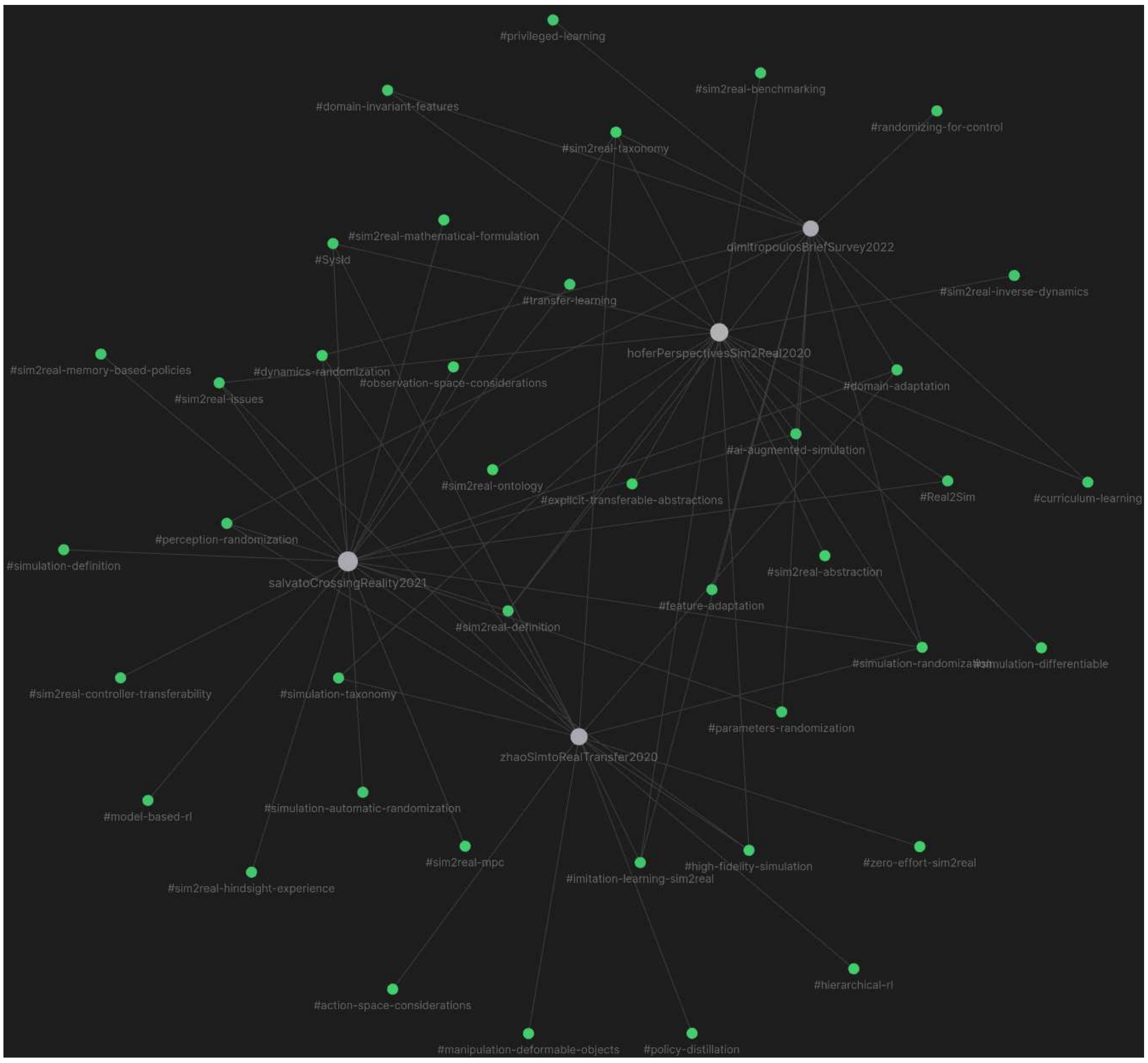

#privileged-learning
#sim2real-benchmarking
#domain-invariant-features
#randomizing-for-control
#sim2real-taxonomy
#sim2real-mathematical-formulation
#SysId
dimitropoulosBriefSurvey2022
#sim2real-inverse-dynamics
#transfer-learning
hoferPerspectivesSim2Real2020
#sim2real-memory-based-policies
#dynamics-randomization
#observation-space-considerations
#domain-adaptation
#sim2real-issues
#ai-augmented-simulation
#sim2real-ontology
#explicit-transferable-abstractions
#Real2Sim
#curriculum-learning
#perception-randomization
#simulation-definition
salvatoCrossingReality2021
#sim2real-abstraction
#feature-adaptation
#sim2real-definition
#simulation-randomization
#simulation-differentiable
#sim2real-controller-transferability
#simulation-taxonomy
#parameters-randomization
zhaoSimtoRealTransfer2020
#model-based-rl
#simulation-automatic-randomization
#sim2real-mpc
#zero-effort-sim2real
#high-fidelity-simulation
#imitation-learning-sim2real
#sim2real-hindsight-experience
#hierarchical-rl
#action-space-considerations
#manipulation-deformable-objects
#policy-distillation

Figure 2.2 Sim2Real survey papers [17–20] knowledge graph.

subsequent methodologies for the Sim2Real techniques, in a way that not only encompasses the existing ones but it would potentially allow for new ones to be readily identified and added in a scalable and expandable way.

We attempt to attack this problem by firstly identifying the overall span of the knowledge map covered by the survey papers [17–20]. By using a set of keywords we can extract a graphical representation on the existing knowledge on the field (See fig. 2.2).

Based on that in combination with a further, fairly exhaustive, research in the robotics literature, we propose a set of the main thematic entities on Sim2Real upon which we try to further analyse and expand respectively with a strong focus in a taxonomical study on Sim2Real.

Lastly, it is important to point out that as the spectrum of robotic fields and application is quite vast and although ideally Sim2Real should be treated in a unified way [18] for e.g navigation to manipulation, control, planning, perception etc. our literature survey steers in the direction of the problem of robotic manipulation with a (potentially soft) gripper. This particular problem has the unique element of really hard to model phenomena in simulation, such contact and friction which are NP-Hard [19], elastic deformations and material failure modes which allow to challenge against established Sim2Real assumption and commonly applied practices.

2.1.2 Sim2Real Definition

2.1.2.1 Sim2Real Definition in Literature.

As expected the typical definition in literature of Sim2Real refers to the attempt of transferring behaviours learning in simulation to the real world [17] where the gap lies on the modeling imperfections of the simulator [17].

Hofer et al. [19] raise the debate question on whether Sim2Real is really a new concept and not just an unecessary rebranded old concept such as model-based Reinforcement Learning, domain-randomization or system identification. The opposing side to this arguments that even Sim2Real is not new this potential re-branding could be beneficial to the field as a whole by connecting interdisciplinary research paths.

Salvato et al. [18] restricts the Sim2Real definition within the Reinforcement Learning framework and specifically the transferability of controllers. More specifically, the "Reality Gap" (RG), per author, is defined as the "degradation" in performance of a controller learned in simulation when deployed to the real-world as a result of modeling inaccuracies in the simulator side.

Muratore et al. [21] apart from mentioning the unmodeled dynamics, they also mention the numerical errors introduced by the simulator as a root cause for the Sim2Real gap.

2.1.2.2 The Historical Frame. From controller to agent.

Modeling and Simulation have always been an integral part of the software development of robotic systems and control/automation systems in general. A direct transfer of the software, which is only evaluated in a simulated environment, to the real-world has never been so far neither expected nor required as it is commonly accepted by the community that due to the, always omnipresent, modeling discrepancies a complete behavior transfer is not a system requirement.

Modeling and Simulation though remained crucial especially for early stage development in the content of the Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL), Hardware-in-the-Loop (HiL) model-based software development for control and automation systems, where a significant number of issues can be caught early on, in a safe, measurable and controllable, way saving significantly in time, effort and cost. After this model-based development pipeline a deployment and testing to the real hardware can eventually take place, where ideally with only some minor adjustment and fine-tuning, the final desired requirements can be fulfilled.

With the advent of the recent research breakthroughs in end-to-end architectures for continuous control in robotics a significant shift in culture seems to be taking place. The dynamic simulation environments, imperfect as they are, in the Reinforcement-Learning content, allow the agents to learn behaviors that can be considered "smart" and "intelligent" in ways that are simply extremely difficult to manually embed with a more "traditional" model-based control design approach. Only if we could reliably and seamlessly "transfer" this learned behavior to the real-world...

2.1.2.3 Sim2Real as a System Design Requirement.

As the Simulation to Reality gap (Sim2Real) or simply the Reality Gap (RG), as referenced by Salvato et al. [18] is not a newly introduced concept we attempt to define it in terms of a cultural shift, namely from the cybernetics era and the "smart systems", from the traditional controllers to now mostly called agents, from the "strict" model-based to data-driven etc.

We propose a definition based on the following criterion:

"Sim2Real defines **the expectation** of a direct transfer of a functional quality achieved in Simulation to the Real-World."

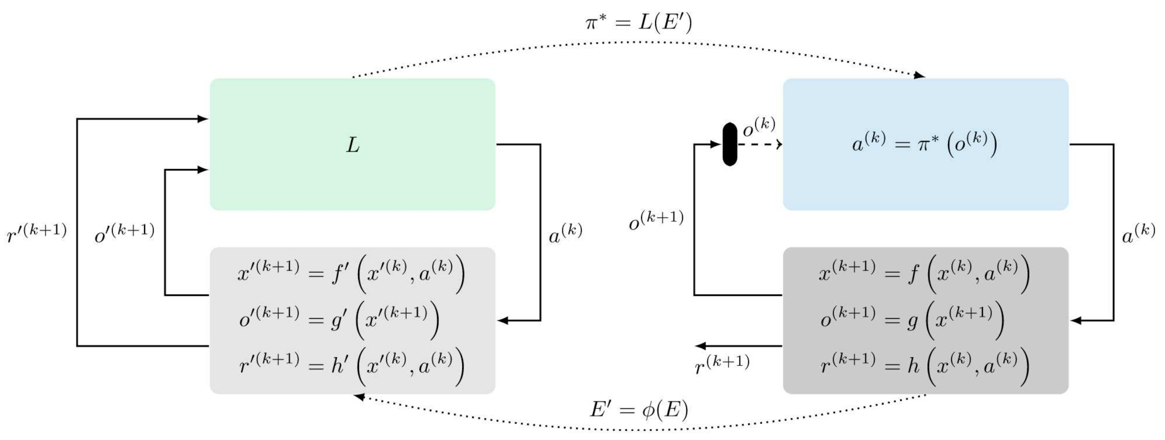
This definition is agnostic the the development methodologies utilised, it is therefore not restricted to a Reinforcement Learning (RL) framework only.

Additionally this criterion is agnostic to whether explicit Sim2Real techniques are utilised or just simply the agent is directly transferable due to intrinsic properties in a "zero effort" manner.

Lastly it is necessary to define the "transfer" term.

We propose the following definition:

"A Sim2Real transfer, in a robotics setting context, is fulfilled when it can be proven that the agent can perform the functional quality learned in simulation, well enough to solve the long-tail distribution problem in the real-world (99.99% success rate)."



$\pi^* = L(E')$
L
$a^{(k)} = \pi^*\left(o^{(k)}\right)$
$o^{(k)}$
$r'^{(k+1)}$
$o'^{(k+1)}$
$a^{(k)}$
$o^{(k+1)}$
$a^{(k)}$
$x'^{(k+1)} = f'\left(x'^{(k)}, a^{(k)}\right)$
$o'^{(k+1)} = g'\left(x'^{(k+1)}\right)$
$r'^{(k+1)} = h'\left(x'^{(k)}, a^{(k)}\right)$
$x^{(k+1)} = f\left(x^{(k)}, a^{(k)}\right)$
$o^{(k+1)} = g\left(x^{(k+1)}\right)$
$r^{(k+1)} = h\left(x^{(k)}, a^{(k)}\right)$
$r^{(k+1)}$
$E' = \phi(E)$

Figure 2.3 Sim2Real Mathematical Sim2Real Definition [18]

The later is closely related to the “Learning Reality Gap” as defined by Paull nad Courchesne [22].

2.1.3 Sim2Real Mathematical Formulation

Usually Sim2Real is treated empirically by researchers and a well defined, commonly and unifiedly, standardised mathematical formalisation and definition is still largely lacking.

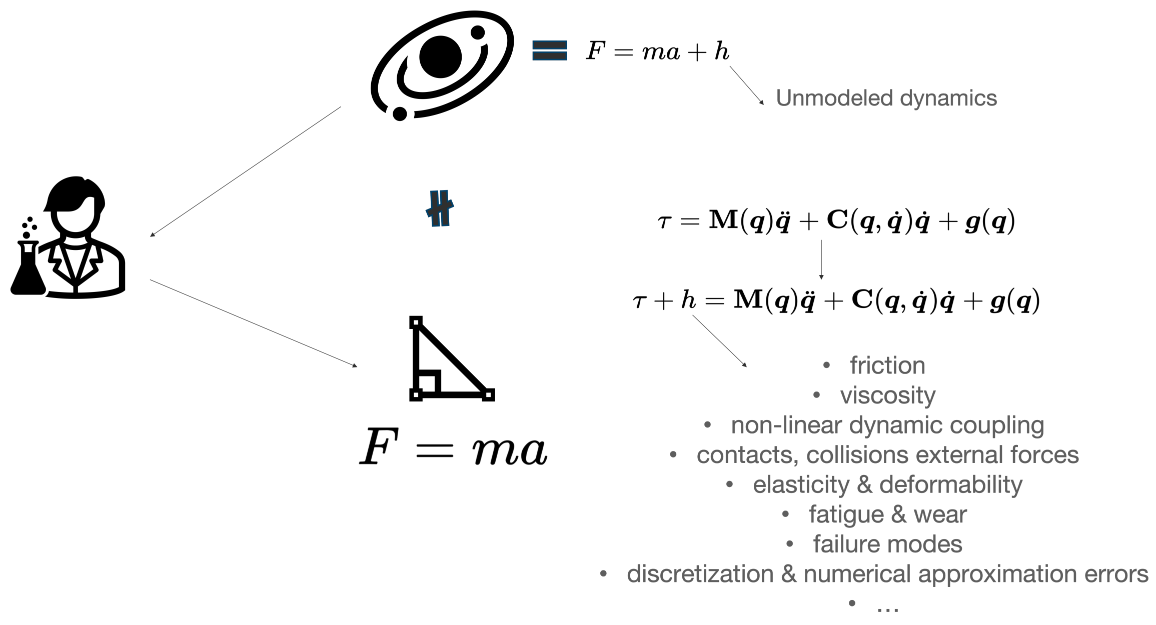
Salvato et. al [18] have a attempted a mathematical formalisation and definition for Sim2Real (see fig. 2.3).

Where E' is the simulator which models the environment E using the transformation ϕ. L is the agent and π^* the policy learned in simulation.

We cohere to this definition throughout our current work. Pure mathematical theoretical considerations on the Sim2Real, although definitely necessary for the advancement of the field, are outside of the scope of this line of work.

2.1.4 Why not just a Realistic Simulation for Sim2Real?

An obvious question arises after a first encounter with the field, namely why not just build more realistic simulation as a mitigation strategy? Although this is certainly one of the



$F = ma + h$
Unmodeled dynamics
$\tau = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$
$\tau + h = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$
• friction
• viscosity
• non-linear dynamic coupling
• contacts, collisions external forces
• elasticity & deformability
• fatigue & wear
• failure modes
• discretization & numerical approximation errors
• ...
$F = ma$

Figure 2.4 Sim2Real is not fundamentally a problem of unknown physics

already applied techniques for Sim2Real, as we explain in the next section, the existence of a generic, task agnostic, extremely high fidelity simulator for robotics is a lucid goal. There will aways be a Sim2Real gap of some sort, larger or smaller.

The history of science could be crudely summed up as an attempt to formally model the phenomena of the natural world in order to gain understanding and therefore predictive power. Fundamentally, Sim2Real for robotics, is not a problem of lack of understanding regarding the underlying physics involved, as the physics and phenomena involved in scale of robotics application are well defined and understood in the sciences, and lie well beyond the current envelope, probably with the exception of the mechanics of “Intelligence” in a biological frame.

The constraints in narrowing the Sim2Real gap lie in the following:

- Computational complexity e.g for contact-friction among objects (NP-Hard), deformable materials, failure modes which require expensive FEM analysis.

- Law of diminishing returns. After a certain point it becomes increasingly expensive in effort, time and complexity to identify, model and tune high-order phenomena of questionable value and effect to the robotics task at hand, making it infeasible for robotic teams to maintain.

This raises immediately some questions:

1. How do we define a good-enough simulator for the robotic task at hand?
2. What explicit Sim2Real techniques, such as domain-randomization, and pipeline architectures can compensate for the gap to reality and can there be well-defined guarantees on performance?
3. Can we utilise, at-least-some, data from the real-world setup for agent fine-tuning or/and residual un-modelled phenomena capturing?

These are the essential main questions with concert to the Sim2Real problem.

2.1.5 Sim2Real Ontological Question

An interesting question that rises regarding a study on Sim2Real, as Hofer et al. [19] brought up, is whether Sim2Real deserves to be standalone scientific and technological standalone entity as a separate and distinct field of study.

The rapid raise in popularity of the term “Sim2Real” in combination with the inter-disciplinary nature of Sim2Real combining fields of robotics, control, computer vision, mechatronics etc. can be extremely beneficial and fruitful in the generation of paradigm shifts and radical new standpoints.

This approach though should be careful handled and in good coordination with the co-jointed scientific and technological field in order to avoid an unnecessary fragmentation among the community [19].

2.1.6 Is Sim2Real really necessary?

Thought our line of argumentation for Sim2Real so far the simulation is assumed to be an integral part for solving the quest of a truly smart agent operating in the real world. But is the simulation part really a fundamental part for achieving intelligence? Despite the fundamental issues when dealing exclusively with data and hardware in the real world, namely, cost, time effort and safety, there are cases where this might be preferable and more effective.

A particularly successful case of a Real2Real approach is the one applied in imitation learning schemes, where the agent is trained directly on data received by an expert trajectory applied in a real-world setup like in the seminal case of the work presented by Shi et. al [23].

An effective Real2Real workflow would pose the benefit of eliminating the need for developing a simulated "digital twin" environment which depending on the task at hand can be a very costly and complicated endeavour.

At the moment the usage of simulation is highly incentivised by the need of a large amount of data for training the existing deep learning architectures. Although the research field of "few-shot" learning is very active the need for a high number of trial-error tries in the RL context is still the predominant method.

2.1.7 Sim2Real as a multidisciplinary "Meta" field

A crucial factor in any approach to this field is the realization that Sim2Real is a multidisciplinary field which requires an optimal combinatoric distillation of a number of fields.

We identify the following adjacent scientific and engineering disciplines:

Table 2.1 Engineering & Scientific Principles connected to Sim2Real

Fields	Connection to Sim2Real
Reinforcement Learning	The “birth” of Sim2Real stems from breakthroughs in deep end-to-end reinforcement learning for continuous control. Any advancement in the RL field, especially in data efficiency during training in the direction of “few shot” learning would be pivotal in any sim2real approach and could even lead potentially to its obsolescence.
Machine Learning	Advancements in Machine Learning are directly affecting the Sim2Real pipeline especially which concern to core computational architectures such as deep layered neural networks, auto-encoders, and attention mechanism in combination with advancements in optimization techniques for training. Meta-Learning architectures [24] are also pivotal in Sim2Real.
Representation Learning	**We believe this is the field most closely connected with Sim2Real. Simulation and modeling as computational physics are essentially our best effort in human intuitive and mathematically comprehensible terms to build representations about our surrounding cosmos. Sim2Real workflows essentially are essentially assisting the agent to see the world through our own man-made representations (physics) first before engaging in the real-world. This line of argumentation regarding representation learning and Sim2Real, their connections and parallels, are at the core of this Thesis.**

Fields	Connection to Sim2Real
Computer Vi- sion	As computer vision advances, along with the increasing levels of feature abstraction hierarchies in image processing, there could be significant implications for Sim2Real robustness.
Robotics	As the available robotic hardware platforms increase together with the number of field applications the need for respective Sim2Real is proportionally following those trends.
Physics En- gines & Sim- ula- tion Frame- works	Every advancement and performance increase in the available physics engines and simulation frameworks, as well as the software libraries and tools available, has huge consequences in utilising the proper technique for Sim2Real.
Computer Sci- ence	Advancements in computational complexity that would make computations of hard to model phenomena faster and more accurate, especially for friction, contacts, deformations and failure modes would also be pivotal for bridging the Sim2Real gap and enabling robotic applications.

Fields	Connection to Sim2Real
Control Engineering	Control Theory is a well-established field with strong mathematical foundation and rich tradition in building advanced and complex military, space and commercial applications. Establishing bridges between and allowing for a convergence between concepts such as adaptive and optimal control, model-based design, Model-Predictive-Control (MPC) and representation learning etc. would significantly impact the Sim2Real workflow.

Any advancements in Sim2Real lie in the overlap of all the aforementioned disciplines and their current state-of-the-art, which automatically constitutes Sim2Real a meta-scientific and technological field. This brings automatically some inherit challenges and opportunities, as keeping-up with the progress in each adjacent field is a haunting task but at the same time synergies and potential bridges between fields can accelerate the overall scientific progress in ways not possible when studying each field independently.

2.1.8 Methodological Gaps and Thesis Contributions

Domain Randomization (DR) has emerged as a critical technique for addressing the *reality gap* in robotics, enabling the transfer of policies learned in simulation to real-world environments. While DR has shown significant promise, several gaps and limitations in current methodologies persist, which my thesis explicitly addresses through novel approaches.

2.1.8.1 Identified Gaps in Current Methodologies

1. **Excessive Parameter Randomization**:

- Many DR methods rely on uniformly randomizing a wide range of simulation parameters, such as friction coefficients, sensor noise, and visual properties. However, this often leads to increased task complexity and suboptimal policies due to the exponential growth in sample complexity with the number of randomized parameters [25–27].
- Additionally, inappropriate or excessive randomization can result in overgeneralized models that fail to exploit specific environmental constraints for optimal performance [28, 27].

2. **Limited Contextual Adaptation**:
   - Most DR approaches assume static parameter distributions throughout training, which neglects the evolving dynamics of real-world environments. This rigidity limits the ability to adapt policies dynamically when faced with unforeseen conditions [27, 29].
3. **Dependency on High-Fidelity Simulations**:
   - Despite advances, existing DR techniques are constrained by the fidelity of the underlying physics engines. Simulators often fail to capture nuanced real-world phenomena such as deformable object interactions or complex contact dynamics, leading to persistent reality gaps [25, 30].
4. **Inefficient Real2Sim2Real Integration**:
   - While Real2Sim2Real workflows have been explored, they often lack seamless integration of real-world data into simulation updates. This results in inefficient feedback loops and suboptimal exploitation of real-world observations during policy refinement [30, 31].

2.1.8.2 Unique Contributions of This Thesis

To address these gaps, my thesis introduces an innovative framework that combines analytical modeling with domain randomization in a Real2Sim2Real pipeline. Key contributions include:

1. **First-Order Analytical World Models**:
   - By leveraging first-order approximations within a rigid multibody physics engine, my approach reduces reliance on high-fidelity simulations while retaining sufficient accuracy for robotic manipulation tasks. This analytical model serves as a robust foundation for simulating diverse scenarios without overwhelming computational costs.
2. **Dynamic Scene Reconstruction**:
   - A novel 3D scene reconstruction module integrates real-world observations into simulation updates. This Real2Sim component enhances the simulator's fidelity by iteratively refining key environmental parameters based on observed discrepancies between simulated and real-world outcomes.
3. **Model Predictive Path Integral (MPPI) Planner Integration**:
   - The incorporation of an MPPI planner showcases the practical and optimal harnessing of the predictive power of our model as it allows to readily synthesize smart manipulation policies without the need for training and in a fully interpretable and explainable manner.
4. **Real2Sim2Real Pipeline**:
   - My framework establishes a perception-planner-action pipeline where real-world data informs simulation updates (Real2Sim), and improved simulations refine policy learning for real-world deployment (Sim2Real). This process reduces

reliance on static parameter distributions and enhances adaptability to dynamic environments.

2.1.8.3 Conclusion

By addressing critical gaps in domain randomization methodologies—such as excessive parameter randomization, limited adaptability, and inefficient Real2Sim2Real integration—this thesis provides a comprehensive framework for bridging the simulation-to-reality gap in robotics. The combination of analytical modeling and iterative feedback loops represents a significant step forward in achieving robust and efficient sim-to-real transfer for robotic systems.

2.2 Sim2Real Evaluation Metrics

Before any attempt on listing any technique for Sim2Real it is necessary to define a set of metrics and benchmarks in order for any methodological and systematic evaluation approach to take place.

2.2.1 Simulation Optimization Bias (SOB)

Muratore et al. [32] introduce a concrete mathematical definition of the Simulation Optimization Bias (SOB) which is critical in Sim2Real and can be a useful indication of policies that tend to overfit and exploit simulation imperfections which in turn hinder a deployment to reality.

As previously defined, the RL setting with domain randomization in place for parameter distribution ξ over a set of environments the goal is the maximization of the expected return over all environment instances

$$J(\theta) = \mathbb{E}_{\xi}\left[J(\theta, \xi)\right]$$

θ is the parametrization of the policy $a_t \sim \pi(a_t \mid s_t; \theta)$

Formulating the RL problem as a Stochastic Program (SP) with $\theta^* = \arg\max_{\theta \in \Theta} J(\theta)$ we get:

$$J(\theta^*) = \max_{\theta \in \Theta} \mathbb{E}_\xi \left[J(\theta, \xi) \right] = \max_{\theta \in \Theta} J(\theta)$$

An approximation of the above can be produced with n domains:

$$\hat{J}_n(\theta_n^*) = \max_{\theta \in \Theta} \hat{J}_n(\theta) = \max_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} J(\theta, \xi_i)$$

Ultimately the Simulation Optimization Bias (SOB) can be defined as:

$$b\left[\hat{J}_n(\theta_n^*)\right] = \mathbb{E}_\xi \left[\max_{\theta \in \Theta} \hat{J}_n(\hat{\theta}) \right] - \max_{\theta \in \Theta} \mathbb{E}_\xi \left[J(\theta, \xi) \right] \geq 0.$$

with the property that it is always positive, namely **the policy's performance in the target domain is systematically overestimated**.

2.2.2 Optimality Gap (OG)

Consequently a minimization of the SOB should be an objective for successful Sim2Real transfer. Muratore et al. [32] suggest an approximation of the Optimality Gap (OG) which relates to SOB since a direct computation of SOB is intractable.

The Optimality Gap at the candidate solution θ^c is:

$$G(\theta^c) = J(\theta^*) - J(\theta^c) \geq 0,$$

Muratore et al. [32] recommend a sample based estimation of the Optimality Gap, show a positive correlation between the sample size and the decrease of the OG. They also establish the connection between the SOB and OG, where a reduction in OG leads in a reduction in SOB.

Based on the aforementioned key metrics, namely the SOB and OG Muratore et al. [32] also introduce an upper bound, the Upper Confidence Bound on the Optimality Gap (UCBOG) during training based on a transferability assessment which in turn results in convergence with less steps.

2.2.3 The question of Sim2Real Predictivity

2.2.3.1 Sim-vs-Real Correlation Coefficient (SRCC)

Kadian et al. [33] raise the important question of “Sim2Real predictivity”, namely how can we evaluate whether performance increase in simulation between different agents translates in performance difference in reality as well. In an attempt to quantify this Sim2Real discrepancy they introduce the “Sim-vs-Real Correlation Coefficient (SRCC)”. By evaluating the SRCC metric in an autonomous navigation problem the predictor was found able to estimate potential failure when deployed in real. Performance differences in simulation do not seem to translate in the real-world setup as the agent seems to be exploiting the simulation imperfections for performance gains.

Sim-vs-Real Correlation Coefficient is defined as the Pearson correlation coefficient (bivariate or other e.g rank correlation can be used) on the set of pairs $\{(s_1, r_1), \ldots, (s_n, r_n)\}$ where s_i and r_i are the success rate of an n_{th} agent in simulation and reality respectively. A coefficient close to 1 would suggest a simulator with high predictive power and good transferability chances to reality.

Although this metric is a potentially useful indicator, as Paull & Courchesne [22] point out, it can be “myopic”, does not asses underlying dependencies and ultimately it does not suffice for predicting the effectiveness of a simulator for transferring in real.

2.2.4 Sim2Real gap conditioned over the robotic task

Paull & Courchesne [22] make the crucial statement that the Sim2Real gap should not be examined naively as a standalone independent problem but it should be conditioned over the robotic task at hand. In essence a metric on Sim2Real gap alone, from a pure physics perspective, could indicate potentially a large discrepancy but if the task “transfers” successfully to real then the Sim2Real should be effectively zero.

2.2.4.1 Predictive Reality Gap (PRG)

Paull & Courchesne [22] introduce the Predictive Reality Gap (PRG) for a given task T defined by evaluation metrics M and an A that generates trajectory x_{sim} in simulation and x_{real} in real as:

$$\mathrm{PRG} \triangleq \sum_{i=1}^{m} \beta_i \left| M_i(X_{\mathrm{sim}}^{1:N}) - M_i(X_{\mathrm{real}}^{1:N}) \right|$$

A *PRG* factor of zero would indicate a “perfectly faithful” simulator. This metric is not that different from the aforementioned SRCC with the difference of the task conditioning.

2.2.4.2 Relative PRG (RPRG)

Paull & Courchesne [22] introduce the Relativ PRG (RPRG) as follows:

Given K agents, the relative predictive ability of a simulator is defined by its ability to accurate predict the binary relations between agents. $A^{\mathrm{sim}} = \left[\alpha_{ij}\right]_{i,j=1..K}$ is a matrix with it’s elements defined as:

$$\alpha_{ij}^{\mathrm{sim}} = \begin{cases} 1 & A_i^{\mathrm{sim}} \geq A_j^{\mathrm{sim}} \\ 0.5 & A_i^{\mathrm{sim}} \neq A_j^{\mathrm{sim}} \text{ and } A_j^{\mathrm{sim}} \neq A_i^{\mathrm{sim}} \\ 0 & A_i^{\mathrm{sim}} \leq A_j^{\mathrm{sim}} \end{cases}$$

The A^{real} is constructed the same way. The *RPRG* is defined as:

$$RPRG(A_{1:K}) = \sum_{i,j=1}^{K} \left| \alpha_{ij}^{\text{sim}} - \alpha_{ij}^{\text{real}} \right|$$

A “perfectly faithful” simulator given the *RPRG* metric would mean that it produces “the identical partial order over agents that would be produced if the agents were run on the real robot.”

2.2.4.3 Learning Reality Gap (LRG)

Paull & Courchesne [22] introduce the Learning Reality Gap (LRG) in an attempt to evaluate the simulator this time as a teacher instead of a predictor.

“Learning Reality Gap (LRG) is **the number of trials on the real robot needed for an agent trained in simulation and transferred to the real robot**, $A^{sim2real}$ to achieve equivalent or better performance”

2.2.5 Sim2Real Observation Discrepancy

Chebotar et al. [34] incorporate the observation discrepancy function D for measuring the Sim2Real Gap for real world observation trajectories $\tau_{\text{real}}^{ob} = \left(o_{0,\text{ real}} \ldots, o_{T,\text{ real}}\right)$ and simulated observation trajectories $\tau_{\xi}^{ob} = \left(o_{0,\xi} \ldots, o_{T,\xi}\right)$ as follows:

$$D\left(\tau_{\xi}^{ob}, \tau_{\text{real}}^{ob}\right) = w_{\ell_1} \sum_{i=0}^{T} \left|W\left(o_{i,\xi} - o_{i,\text{ real}}\right)\right| + w_{\ell_2} \sum_{i=0}^{T} \left\|W\left(o_{i,\xi} - o_{i,\text{ real}}\right)\right\|_2^2$$

Where w_{ℓ_1} and w_{ℓ_2} are the weights of the ℓ_1 and ℓ_2 norms, and W are the importance weights for each observation dimension. The authors also mention the application of a Gaussian filter on the D measurement as a smoothness operation to compensate for potential misalignments.

2.2.6 Offline Trajectory Error

Aljalbout et al. [35] introduce the “Offline Trajectory Error (OTE)” as a metric in order to study the effects of different action space formulations for Sim2Real.

$$\mathrm{OTE}(\pi) = E_{a,q_{\text{real}} \sim \mathscr{D}_{\text{real}}} \left| q_{\text{sim}} - q_{\text{real}} \right|$$

a and q_{real} are the actions and joint configurations respectively from dataset D_{real} . The respective simulation joint configurations q_{sim} are collected by applying a in simulation in an open-loop fashion.

2.2.7 Sim2Real Fitness Objective

Collins et al. [36] for evaluating their experiments for simulation fine-tuning with real-world data they introduce the following fitness factor for measuring the Sim2Real discrepancy.

$$f = \frac{\sum_{\text{points}} \sqrt{\sum_{i=x,y,z} \left(W_i^d - W_i^s\right)^2}}{n_{\text{points}}}$$

which is the normalized Euclidean distance between the wrist joint of the robotic manipulator in simulation $W^s_{x,y,z}$ and the “dataset” $W^d_{x,y,z}$.

2.2.8 Sim2Real Gap Metrics for Point-Clouds

In [37] Blanco-Mulero et al. use the Chamfer Distance (CD) and the Hausdorff Distance (HD) distance metrics for measuring the discrepancy between real and simulated cloths as an attempt to evaluate different simulation frameworks for deformable object manipulation.

$$CD(\mathscr{V}_t, \mathscr{P}_t) := \frac{1}{|\mathscr{V}_t|} \sum_{v \in \mathscr{V}_t} \min_{p \in \mathscr{P}_t} \|v - p\|_1$$

$$HD(\mathcal{V}_t, \mathcal{P}_t) := \max_{v \in \mathcal{V}_t} \min_{p \in \mathcal{P}_t} \|v - p\|_1$$

For a point cloud $\mathcal{P}_t$ and a simulated mesh with vertices $\mathcal{V}_t$. The authors recommend the Manhattan distance for computing the norm instead the Euclidean one.

2.3 RL - Design Considerations for Sim2Real

2.3.1 Learning Algorithm and Policy Architecture

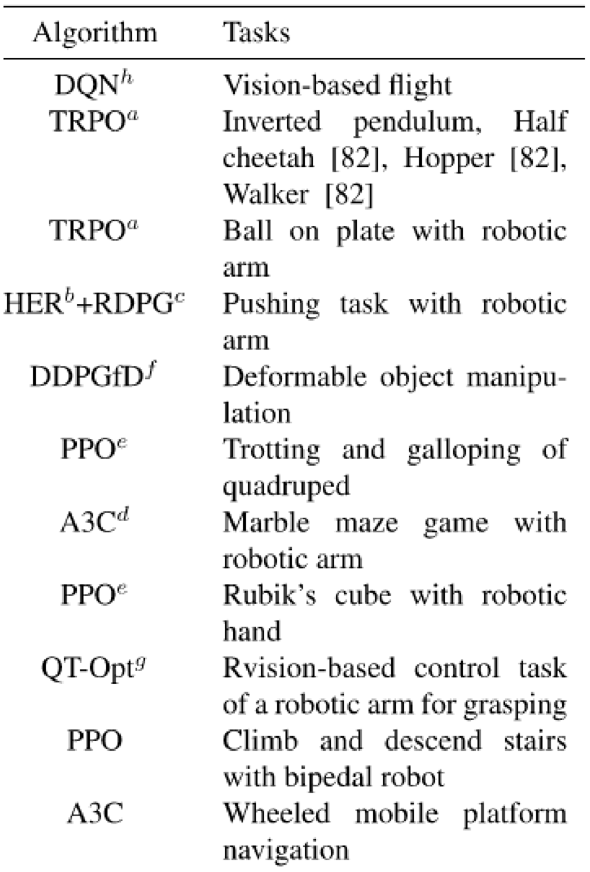
The general consensus on picking a reinforcement learning algorithm and a policy architecture is to proceed based on the task and the computational resources available and additionally apply dedicated sim2real techniques for increasing the tasks for a successful transfer. There has been in general little effort to systematically investigate the correlation between RL-Algorithm, Policy Architecture and Sim2Real transferability.

Salvato et al. [18] in their survey present a list with influential works in the field and the RL-Algorithms that have been utilized under the Sim2Real prism (See fig. 2.5)

It can be agreed that LSTM policy architectures perform better in complex dynamic scenarios and this exact generalization and robustness under different dynamics allows for a more successful Sim2Real transfer [38, 39, 16].

This methodological schism, namely a dichotomy in RL-Algorithm/Policy Architecture and Sim2Real methodology can be hypothesized that incorporates potential performance bottlenecks and general overall system and workflow inefficiency which is direct call for a holistic approach that should unify RL-Algorithms-Policy Architectures-Simulation/Model under a single framework.

Chi et al. [40] propose the a denoising diffusion process for representing the visuomotor policy, declaring an average 46.9% improvement against standard baselines.



Algorithm	Tasks
DQN^h	Vision-based flight
TRPO^a	Inverted pendulum, Half cheetah [82], Hopper [82], Walker [82]
TRPO^a	Ball on plate with robotic arm
HER^b+RDPG^c	Pushing task with robotic arm
DDPGfD^f	Deformable object manipulation
PPO^e	Trotting and galloping of quadruped
A3C^d	Marble maze game with robotic arm
PPO^e	Rubik's cube with robotic hand
QT-Opt^g	Rvision-based control task of a robotic arm for grasping
PPO	Climb and descend stairs with bipedal robot
A3C	Wheeled mobile platform navigation

Figure 2.5 RL-Algorithms and Tasks that have been utilized for Sim2Real scenarios and that have applied Domain-Randomization [18]

2.3.2 Observation Space Considerations

The formulation of the observation space as a combination of proprioceptive and exteroceptive sensor aggregate and the shaped feature space as a result of a potential embedding of high dimensional inputs, naturally affects a sim2real transfer.

In the case of [41] only depth readings are utilized from the RGB-D camera which in combination with privileged information training in simulation with a consequent teacher-student imitation learning pipeline result in a successful sim2real transfer. The crucial element in this case is the omission of the RGB component which has a high degree of discrepancy between sim and real and a focus on the pure geometric nature of the depth sensor.

Yin et al. [42] for the in-hand manipulation utilize exclusively a set of 16 binary low-cost Force-Sensing-Resistors (FSR) as exteroceptive information evenly covering the fingers and palm of the robotic hand instead of an vision RGB-D based approach which allows for a very efficient Sim2Real Gap minimization due to the binary nature of the sensors. The approach remains effective though as the total 16bits of touch resolution allows for a representation of a total of 2^{16} states.

Beddow et al. [43] on their work for grasping attempts with soft fingers and reinforcement learning strategies through pure force-feedback, conduct ablation studies where they note that using fewer tactile sensors, the sim2real transfer was more “variable”, balancing between potential overfitting and simulation bias when using palm force signals and poor training performance when working with fewer sensors.

We can overall conclude that an observation space which encodes measurements and/or features for exteroception that are more of a geometric nature such as distances and positions, or binary signals can help to bridge the sim2real gap.

2.3.3 Action Space Design

Action space design and formulation for an effective control authority, convergence in an RL context as well as a successful Sim2Real transfer is highly important. Aljabout et al. [35] explore exactly the effect that the choice of action space has in Sim2Real. More concretely for the tasks of “pushing” and “reaching” an object with a robotic manipulator they study the difference in position vs velocity vs torque, joint space vs task space, and “base” action space vs “delta” action spaces. We distinguish the following important findings:

- For handling the non-smoothness resulted from RL policies targeting in the Cartesian space position or velocity a direct utilization of Inverse-Kinematics for feeding joint level controllers is recommended.
- In this study direct joint torque policies were not able to perform for the pushing task as they result in abrupt motions outside the robot’s safety boundaries.
- Joint action spaces have less effect in sim2real gap compared to the Cartesian space ones based on the OTE metric.
- Feedback loops should be properly tuned for the implementation of control schemes and translating from velocities/positions to torques for being actually beneficial in sim2real scenarios.
- Multi-step Integrator ($v_d \leftarrow v_d + c \cdot a \cdot dt$) as delta action space behaves better for sim2real when compared to one-step integrator ($v_d \leftarrow v + c \cdot a \cdot dt$).
- In general velocity-based action spaces tend to perform better for sim2real in comparison to position-based strategies.
- Delta-action spaces perform better than base spaces.
- One-step integrator leads to lower tracking error OTE.
- **Joint-velocity as action space in general performs better than Cartesian space-velocity for sim2real overall.**

The joint velocities $\dot{q_d}$ generated by the policy are fed to a simple joint impedance controller in the form:

$$\tau = K(q_d - q) + D(\dot{q_d} - \dot{q})$$

This result is also empirically observed among the RL literature that attempts continuous control in robotics through RL and sim2real.

In practice filtering techniques are usually additionally applied at the action signal in the real-world setup in order to smoothen the resulted signal. Simple exponential moving average filters [42], low-pass filters or rate limiters are used in practice [35].

Lastly regarding the action space design it is important to mention, even outside the scope of Reinforcement-Learning, that the verdict is not exactly concluded. For example Xue et al. [44] in their work based on Diffusion-Style Annealing for sampling based Model Predictive Control, they command directly torques on the quadruped robot where they manage to surpass the previous state-of-the-art which seems totally contrary to the aforementioned results. They utilize carefully tuned system identification for matching the dynamics between simulation (Brax) and the real robot, and they command the desired torque with the addition of a damping velocity term in the form:

$$\hat{\tau}_i = \tau_i - d\omega_i$$

In that case it would be interesting to examine whether an alternative action choice, namely the recommended relative velocity term, could lead to potential improvements in performance.

2.3.4 Reward Function Shaping

It is very common that the reward function incorporates terms specifically designed for improving sim2real transfer.

Yin et al. [42] add a special velocity term $-||v_t||$ in the reward function in order to promote more stable object rotation for in-hand manipulation which in turn increases the transferability of the trained policy. They additionally place a term $-\langle|\tau|, |\dot{q}_t|\rangle$ for improving the smoothness of finger motion by penalizing the "work of the controller".

Yu et al. [45] design a $r_{sim2real}$ term specifically for deployment to real-world hardware setup in an Model Predictive Control (MPC) setting for keeping the overall system robot-object velocities low and in general discourage dynamic in-hand movements.

$$
\begin{aligned}
r_{\text{sim 2 real}} &= 3 \begin{cases} \mathbf{p}^{*}_{ee}, & \text{if } ||\dot{\mathbf{p}}_{ee}|| > 0.3 \\ 0, & \text{otherwise} \end{cases} \\
&+ \begin{cases} \dot{\mathbf{q}}, & \text{if } ||\dot{\mathbf{q}}|| > 0.7 \\ 0, & \text{otherwise} \end{cases} \\
&+ 0.05\mathbf{p} \overset{\circ}{\circ b j} \\
&+ 0.1 \begin{cases} \mathbf{p}_{ee} - \mathbf{p}_{obj}, & \text{if } \left\|\mathbf{p}_{ee} - \mathbf{p}_{obj}\right\| < 0.1 \\ 0, & \text{otherwise} \end{cases} \\
&+ 0.4 \begin{cases} \mathbf{q}_{\text{gripper}} - 1.0, & \text{if } \left\|\mathbf{p}_{ee} - \mathbf{p}_{\text{obj}}\right\| > 0.1 \\ \mathbf{q}_{\text{gripper}}, & \text{otherwise} \end{cases}
\end{aligned}
$$

Haarnoja et al. [46] include two reward terms specifically for improving sim2real transfer. The first penalty term aims to a stress reduction in joints by minimizing the time integral of torque peaks for the constraint forces on the targeted joints. The second term would mitigate the effect of a simulation bias behavior, namely a forward leaning motion which in reality would case balance lose and failure. A reward term for keeping the an upright pose mitigates the issue as the authors suggest.

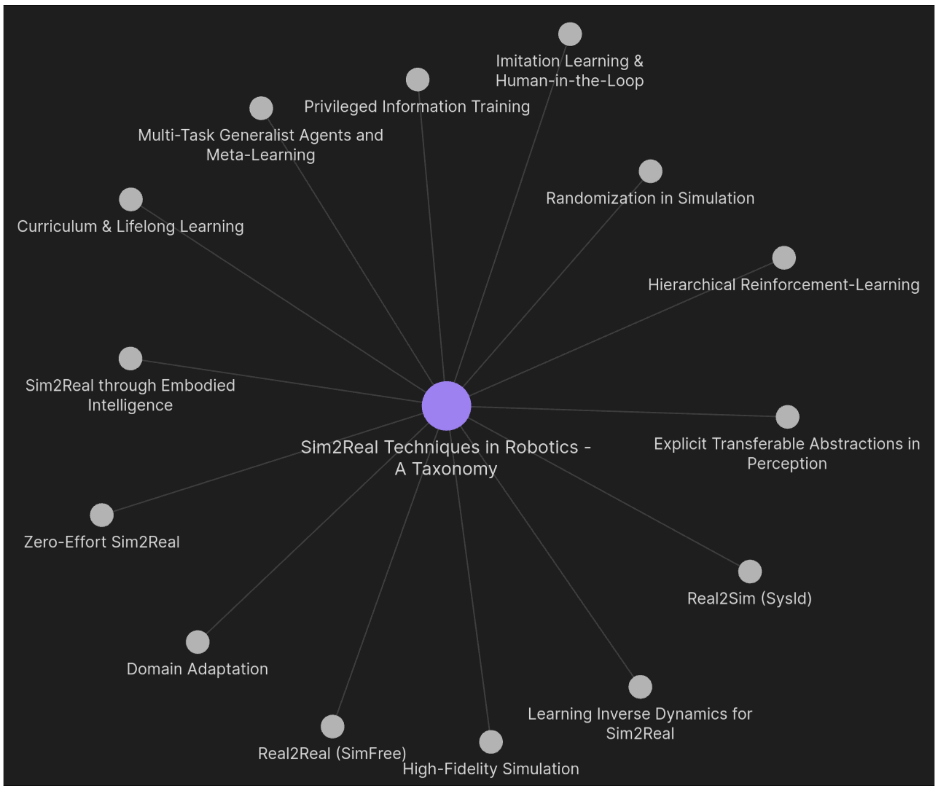

Sim2Real Techniques in Robotics - A Taxonomy
Imitation Learning & Human-in-the-Loop
Privileged Information Training
Multi-Task Generalist Agents and Meta-Learning
Randomization in Simulation
Curriculum & Lifelong Learning
Hierarchical Reinforcement-Learning
Sim2Real through Embodied Intelligence
Explicit Transferable Abstractions in Perception
Zero-Effort Sim2Real
Real2Sim (SysId)
Domain Adaptation
Learning Inverse Dynamics for Sim2Real
Real2Real (SimFree)
High-Fidelity Simulation

Figure 2.6 The Sim2Real Techniques as identified by our literature review.

2.4 Sim2Real Techniques in Robotics - A Taxonomy

Table 2.2 A Taxonomy on Sim2Real

Category	**Subcategory**	**Description**
Domain Randomization	Randomization for Perception	Varying visual aspects such as color, texture, and lighting in simulations.
	Randomization for Dynamics	Altering physical parameters like mass, friction, and elasticity.
	Automatic/Active/Dynamic Randomization	Adaptive approaches to randomization based on agent performance.

Category	**Subcategory**	**Description**
	Randomization for Data Generation/Augmentation	Enhancing training datasets with randomized synthetic data.
	Adversarial Perturbations	Introducing controlled disruptions to test robustness.
Domain Adaptation	Synthetic-to-Real Adaptation	Aligning synthetic and real-world data distributions.
	Task-Aware Domain Adaptation	Customizing adaptation strategies for specific robotic tasks.
	Unmodeled Dynamics Adaptation	Addressing uncertainties not captured in simulation.
Real2Sim Techniques	Traditional System Identification	Calibrating simulation parameters using real-world data.
	Simulator Tuning and Validation	Adjusting simulations to closely replicate real-world behaviors.
	Hybrid Sim & Real Reinforcement Learning	Combining simulated and real-world data for training.
Explicit Transferable Abstractions	Manual Observation Space Design	Predefining relevant features for simulations.
	Auto Feature Space Shaping	Using machine learning to extract transferable features.
Simulation Fidelity	High-Fidelity Simulation	Using detailed models for accurate simulation.

Category	**Subcategory**	**Description**
	Differentiable Simulation	Allowing gradient-based optimization through simulations.
Learning Strategies	Imitation Learning	Training agents using expert demonstrations.
	Curriculum & Lifelong Learning	Gradually increasing task complexity to improve agent adaptability.
	Multi-Task and Meta-Learning	Training generalist agents capable of adapting to multiple tasks.
Zero-Effort Sim2Real	Zero-Shot Generalization	Deploying agents trained in simulation directly to real-world tasks.
	Built-in Architectural Robustness	Designing agents inherently resistant to reality gaps.

2.4.1 Randomization in Simulation

2.4.1.1 Domain Randomization

Salvato et al. [18] make the observation that Sim2Real has been also been successfully achieved for controllers that do not fall in the RL Raum, and they draw parallels to control theory, namely “robust control under parametric or non parametric uncertainty.” Interestingly enough they also identify the first research work in robotics that defines the reality gap and the “treatment of noise” during simulation as a mitigation measure [47].

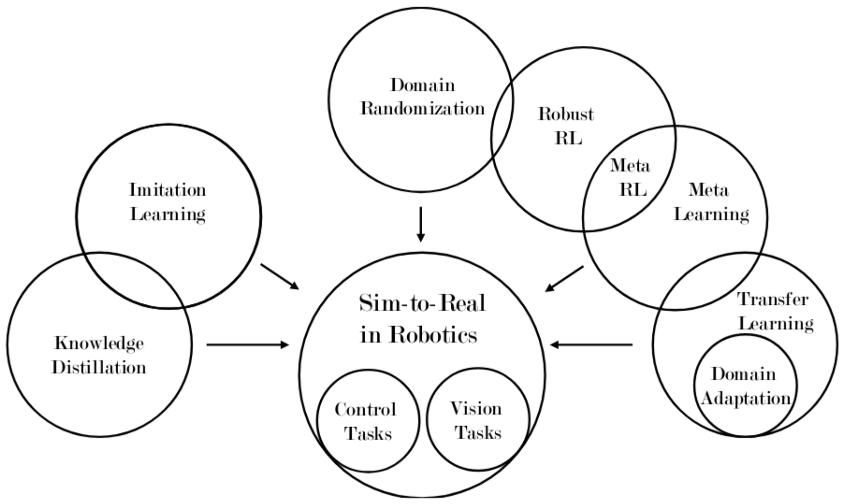


Figure 2.7 Sim2Real Techniques Taxonomy per Zhao et al. [20]

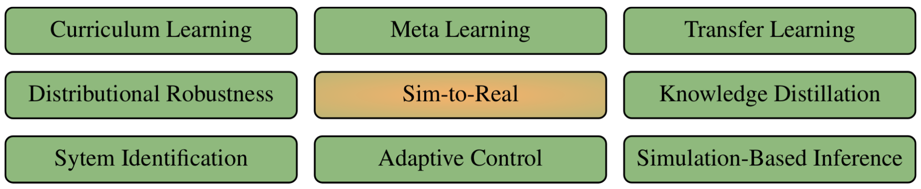


Figure 2.8 Muratore et al. [21] attempt to establish the relations of Sim2Real to other adjacent learning fields mostly with respect to domain-randomization.

Domain-Randomization is currently the most predominant and most strongly associated to Sim2Real technique in the field [17–21]. Domain-Randomization is defined within the frame of Reinforcement-Learning for continuous control with the usage of simulation for training. At its core the idea of introducing randomization in the simulation parameters during the agent training phase states that by doing so we "force" the agent to "robustify" against overfitting, thus learn better internal representations that when transferred to the real-world setup its perception would render it as just "another variation" that would allow the necessary generalization property. Muratore et al. [21] argue that domain-randomization can be seen as a regularization method preventing from overfitting to a specific simulation.

Muratore et al. [21] argue though a historical retrospective that randomized simulation is not a newly introduced concept in the history of computation, simulation and Reinforcement-Learning and that the probably newly introduced concept is a direct, in contrast to an "indirect", tuning of the simulation parameters. Attributing a reasoning behind the popularity in randomization in simulation during training we identify the breakthrough results achieved by the works of Tobin et al. [48], Sadegi et al. [15] and probably most importantly by OpenAI and it's Dactyl Project [16] which apart from gaining significant popularity in mainstream, demonstrated unrepresented results in dexterous robotic manipulation attributing domain-randomization as a key-factor for the significant success in the real transfer.

2.4.1.2 Mathematical Formulation

Muratore et al. [21] expand the typical Reinforcement Learning (RL) problem definition based on the Markov Decision Process (MDP) in order to define the goal as an overall return maximization as usual, but this time over a distribution of domain parameters as such:

$$J(\theta) = \mathbb{E}_{\xi \sim p(\xi)}[J(\theta, \xi)] = \mathbb{E}_{\xi \sim p(\xi)}\left[\mathbb{E}_{\tau \sim p(\tau)}\left[\sum_{t=0}^{T-1} \gamma^t r_\xi\left(s_t, a_t\right) \mid \theta, \xi, s_0\right]\right]$$

Where $\xi \in \mathbb{R}^{n_\xi}$ are the randomization parameters and $p(\xi) : \mathbb{R}^{n_\xi} \to \mathbb{R}^+$ their unknown distribution.

Although this extension lays the foundations for a mathematical formulation of domain randomization in RL the field still lacks a concrete mathematical framework [20] that would allow definite answers, like for example the physical plausibility of randomized parameters [21], in the main currently open questions as presented in the rest of this sections.

2.4.1.3 Domain-Randomization Types

A categorization in types of randomization is observed in the literature, based on which stage of the typical robotics software "perception-planning-control" stack it is applied, the way it is applied and the function it tries to solve.

A quite common categorization for domain-randomization is the distinction in "Randomizing for Perception" and "Randomizing for Control"[17, 20].

Muratore et al. [21] identify domain-randomization into "static", "adaptive" and "adversarial", where in static case a distribution for the simulation parameters is pre-selected before, in adaptive the distribution is updated, potentially from real-world data, during training. The adversarial domain randomization aims at mitigating the well known issue of deep layered architectures being brittle to adversarial attacks by introducing perturbations during the training in order to robustly the agent's performance and thus increasing the chances for a successful Sim2Real transfer. Additionally the authors make the distinction for each of these categories depending on whether data from real-world have been utilised for the determination of the distribution of the randomised parameters.

We propose the following categories for domain randomization based on where and how the randomization is applied:

Table 2.3 Domain-Randomization Function Based Categorization.

Domain Random-ization Function Based Type	Description
Randomization for Data-Generation Data-Augmentation	This category refers to the randomization that occurs for training data-generation or data-augmentation as a separate stage before any training occurs in an offline manner. This could be used for generating a set of distinct environments in which the agent will be trained on in a later stage.
Randomization for Perception	This category refers to the randomization of parameters in the environment aiming to enhance the perception capabilities of the agent, like e.g camera intrinsic and extrinsic parameters, scene colour noise etc.
Randomization for Dynamics-Control	This category refers to the randomization of parameters in the environment aiming to enhance the perception capabilities of the agent, like e.g camera intrinsic and extrinsic parameters, scene colour noise etc.

Domain Randomization Function Based Type	Description
Adversarial Perturbations	This category refers to any attempt in adding noise and perturbations during training in order to compensate and robustify against unmodeled phenomena. E.g adding random forces in a manipulated object, noise injection in control or perception signals should account to that.

Table 2.4 Domain-Randomization Automation Based Categorization.

Domain Randomization Auto-tuning based type	Description
Static Domain Randomization	Static domain randomization refers to drawing samples for the randomized from a constant predefined distribution.
Dynamic Domain Randomization	Dynamic domain randomization refers to dynamically changing the distribution characteristics during training.

Domain Randomization Auto-tuning based type	Description
Automatic Domain Randomization	Automatic domain randomization refers to auto-tuning methodologies for the randomization characteristics.

Table 2.5 Domain-Randomization Real-World Data Utilization Criterion.

Domain Randomization - Real-World Data utilization	Description
Real-World Data enhanced Domain Randomization	This is when real-world data are utilized either before or during training.

Table 2.6 Domain-Randomization Free Sim2Real Criterion.

Domain Randomization Free Sim2Real	Description
Randomization Free	This category is added for completeness in cases where no randomization has been applied but Sim2Real has been achieved.

Based on the categories defined above the randomization type can be precisely defined as potential combination e.g “static randomization for perception the real-world data enhancement”.

We attempt a further explanation and analysis, together with relevant literature references, for each of these categories in the following subsections.

2.4.1.4 Randomization for Data Generation/Augmentation

In [15] the authors report that at the core of the Sim2Real success in indoor autonomous drone flight, lies on the randomization of the 3D environments built in Blender with an emphasis on morphology, textures and rendering conditions. Additionally a set of pre-taining images was collected in simulation through a camera with randomized height and orientation.

In [49] the data set provided through human-in-the-loop in VR is consequently augmented through scene randomization. Although this is a Sim2Sim work, without real-world testing it provides an interesting approach with the initial human-in-the-loop element.

In [50] a data set for grasping pose prediction has been generated fully in simulation through randomized grasping trials with a naive manually crafted grasping strategy in clutters of randomized objects.

Based on the advent of Large Language Models (LLMs) Want et al. [51] utilize GPT-4 in order to generate robotic tasks in simulation for training multi-task agents which is a very promising direction in developing robust and generic agents with thus with strong sim2real potential.

2.4.1.5 Randomizing for Perception

Randomizing for perception is a widely utilized approach for Sim2Real and it has been attributed multiple times as the predominant strategy for solving Sim2Real.

Although the work in [48] is focused only in object localization based on a single monocular camera with an implicit impact in robotic manipulation some important milestones and precursors have been achieved which will later set the stage for end-to-end robotic manipulation.

Based on the literature references [52, 48, 53, 54, 17, 16, 55, 56, 41] we gather the following predominant candidate parameters for randomization for perception.

Table 2.7 Domain-Randomization Example Parameters for Perception.

Perception Randomization Parameters
Object Positions (In grasping & manipulation scenarios)
Object Textures (In grasping & manipulation scenarios)
Distractor Objects (Number, shapes, colours, textures, positions)
Scene Textures (e.g table, floor, robot, surounding objects)
Camera Pose
Camera Field of View
Camera Image Noise
Simulated Point Cloud Noise
Scene Lighting (adding, removing light sources)
Lighting source (6D pose, specular characteristics, intensivity)
Background Colours
Reflectance coefficients
Type of shadows

Perception Randomization Parameters
Height samples (quadruped robot) nominal noise
Root positioning localization error (quadruped robot) drift noise

One of the most successful and intricate attempts on randomization for perception can be bound at OpenAI's Dactyl project [16] where the rendering parameters in simulation have been heavily randomized. This is a bit in contrast to the belief that heavy domain randomization leads to unsuccessful transfers [18] and raises the question on where the line should be drawn on what exactly should be perceived as "heavy randomization".

Another important point is with regard to which of the aforementioned parameters are the most important to randomize for perception. Alghonaim et al. [57] point to the importance of distractors and textures. They also comment on the rendered image quality as "a small number of high-quality images is superior to a large number of low-quality images".

2.4.1.6 Randomizing for Dynamics

Based on the work by [52, 38, 58, 16, 55, 59, 56, 46, 60, 61] we are able to gather a set of parameters which are most typically randomized during training for dynamics.

Table 2.8 Domain-Randomization Example Parameters for Dynamics.

Dynamics Randomization Parameters
Contact Models
Robot's kinematics, e.g link length

Dynamics Randomization Parameters
Center of mass location
Mass of robot links
Damping of joints
Mass, friction, damping of manipulated objects
Motor friction
Motor strength
Working table height
Gains of any underlying controller
Control step time
IMU bias + noise
Backlash
Joint margin
Joint range
Joint friction
Joint angular offsets
external mass to a randomly chosen location on the robot torso

Dynamics Randomization Parameters
PD Gains for motor control

In [38] Peng et al. mention that randomization in the parameters of the system dynamics allow for memory-bases policies to generalize to the dynamics of the real world without the need for an exhaustive system calibration with system identification procedures. It is also worth mentioning that in the same work a set of a total 95 parameters have been randomized during training raising potentially again the question on what constitutes a “heavy randomization” which could hinder a successful Sim2Real transfer. In the work by Haarnoja et al. [46] for example a small number of parameters has been selected for randomization in order to a avoid a conservative policy, which is again in sharp contrast to the work by OpenAI [16].

Swapping Physics Engines During Training Another source of randomization for the scene dynamics could be the physics engine its self. As there are notable differences between physics engines when deploying in Sim2Sim scenarios [21], due to different coordinate representations, numerical solvers, friction and contact models utilized by the underlying physics engines. This could be utilised as source of randomization for dynamics during training.

2.4.1.7 Adversary Perturbations

The role of adversary perturbation is in effect to robustify the agent for phenomena which are not modeled in simulation during training. This can typically involve adding perturbations in action, thus with regard to unmodeled phenomena in actuation, and explicit force/torque perturbations e.g in the torso of a humanoid that learns to walk, in the manipulated object.

Haarnoja et al. [46] applied random perturbations to the humanoid torso as “an external impulse force of 5 to 15 Nm lasting for 0.05 to 0.15 s, to a randomly selected point on the torso every 1 to 3 s”. Li et al. [62] add noise and delay factors in observation with the additional periodical random “pushing” of the torso of the quadruped robot.

Chen et al. [41] apply random forces in the manipulated object.

Serifi et al. [60] perform random pushes on the root, head, hands, and feet and perturb the joint positions by a maximum of ε_q to account for inaccuracy in joint calibration,

Table 2.9 Adversary Perturbations Examples.

Adversary Perturbations
Latency and noise to the actuation
Latency and noise to the observation
Force/Torque perturbations on the robot e.g torso or in the manipulated object

2.4.1.8 When to Randomize

As mentioned by Muratore et al. [21], and as we also observe in literature, the most common approach for applying randomization is the episodic dynamics randomization. This seems to be an empirical result not based in some rigorous mathematical justification.

As argued by Muratore et al. [21] “randomizing the domain parameters at every time step instead would drastically increase the variance, and pose a challenge to the implementations since this typically implies recreating the simulation at every step”.

This is in contrast to some cases where the adversarial perturbations are applied randomly at every step drawn from a constant distribution until the randomization scheme is updated [21].

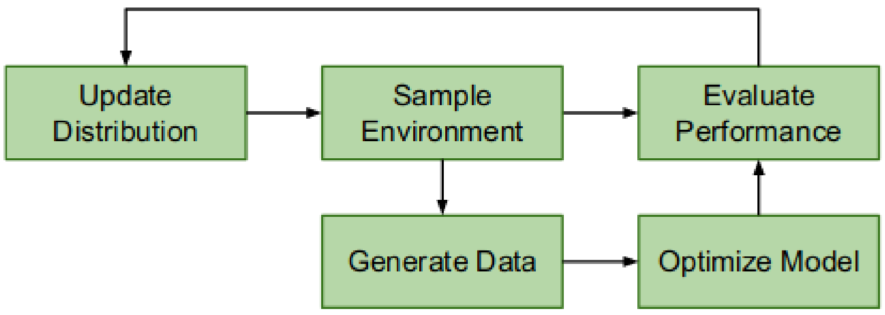


Figure 2.9 Automatic Domain Randomization (ADR) System Overview [16]

2.4.1.9 Automatic/Active/Dynamic Domain Randomization

A significant amount of research effort has been directed towards methodologies for dynamically and/or even fully autonomously setting the probability distribution characteristics for domain randomization in simulation. At it's core the main idea is to adjust the amount of randomness of the parameters based on some performance criterion during training, an idea closely resembling that of curriculum learning.

Automatic Domain Randomization (ADR) OpenAI [16] presents a methodology for an automatic definition for the distribution ranges of the randomized parameters in combination with a dynamic adaptation of those during training based on performance metrics of the current policy (see fig. 2.9).

OpenAI establishes a clear correlation between the Sim2Real transferability and a large ADR Entropy which requires extensive periods for training in large scale in order to outperform manual tuning, which also in turn would require extensive manual experimentation.

DOmain RAndomization via Entropy MaximizatiON (DORAEMON) Very close to the central concept of the aforementioned Automatic Domain Randomization Tiboni et al. [63] introduce the DOmain RAndomization via Entropy MaximizatiON (DORAEMON) which again is based in the idea of a gradual increase, curriculum learning style, of the randomization for the simulation parameters during training based on the current policy "maturity". DORAEMON will increase the the diversity of the sampled dynamics parameters if the chances of success of the current policy are sufficiently high.

DORAEMON and ADR seem conceptually at least identical and analytical ablation studies are necessary in order to address the differences and the potential introduced novelty.

Active Domain Randomization (ADR) In their work "Active Domain Randomization (ADR)" Mehta et al. [26] are introducing a methodology for actively searching for simulation parameters with the highest training disruption influence rendering essentially an adversarial agent as the simulation parameters generator. The problem is indeed formulated as a Reinforcement Learning problem with ADR policy being parametrized with the Stein Variational Policy Gradient [64].

Random Network Adversary (RNA) The generation of adversarial forces that would cause perturbations in the system in order to robustify the agent against unmodeled dynamics can be also subject to automation. In [65, 66] the problem is treated holistically as an adversarial RL problem where the adversarial agent has to learn to challenge the main policy as a destabilizing factor.

Inspired by this approach OpenAI [16] introduce the random network adversary (RNA) which is essentially networks with randomly sampled weights for generating the adversarial perturbation forces during training. These weights are sampled at the beginning of each episode.The authors support that this method outperforms the aforementioned adversarial policies as ultimately the supplied diversity a randomized network is more crucial than a targeted trained and sophisticated adversarial.

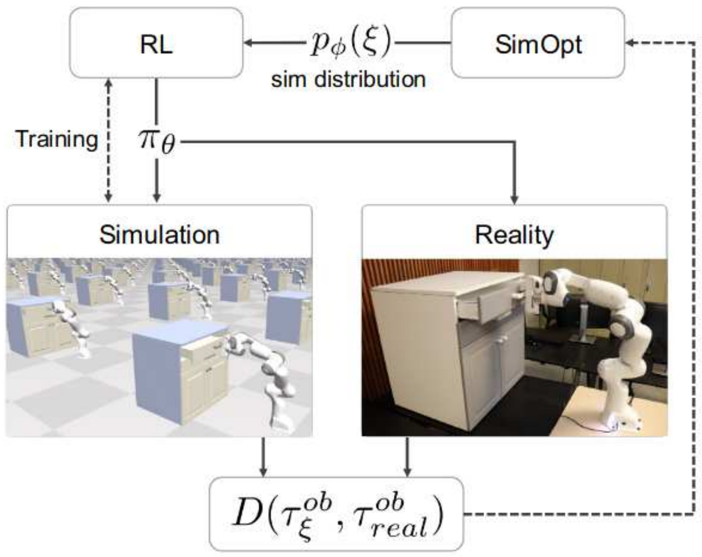


Figure 2.10 Tuning the simulation parameters randomization distribution by measuring a sim2real discrepancy [34]

2.4.1.10 Utilization of Real-World Data

Real-World data can be utilized for a more purposeful and sharp domain randomization with regards to values ranges and stochasticity characteristics. For example in [16] the first environment instantiation for the policy training is performed with parameters measured from the real-world setup.

Hietala et al. [67] provide a methodology where they derive a set of simulated cloths based on the top performing simulation parameters when compared with expert demonstration in the real-world.

Chebotar et al. [34] adapt dynamically the simulation randomized parameters distribution after deployment to the real world and measuring the sim2real discrepancy.

2.4.1.11 Sim2Real without Randomization

Although randomization in simulation during training would seem a necessary element for a Sim2Real transfer, there are some significant testimonies in the literature that would suggest otherwise. Crucial factor to the criticality of randomization in simulation as a necessary

Sim2Real component seems to be the overall architecture, where composite architectures with, a potential model-based, control architecture with an operational space controller (OSC) seem to be inherently more robust to dynamics deviancy between Sim and Real.

Xie et al. [68] in their findings suggest that Sim2Real transfer is possible without dynamics randomization or domain adaptation and they strongly advocate for conservative randomization policies if and only if there are necessary with a careful selection of the truly impactful parameters that should be randomized using domain specific knowledge.

Kaspar et al. [69] utilize indeed an OSC framework with system identification and high-quality model of the robot where they are able to transfer policies without dynamics randomization for a peg-in-hole task.

Dao et al. [70] **make the crucial and usually very unexplored point with regards to the effects of mechanical design, and most importantly the effects of compliant elements**, in this case springs, where they are able to transfer to real due to the compliant mechanics.

2.4.1.12 Addressing Computational Overheads and Performance Trade-offs in Domain Randomization

The integration of domain randomization (DR) into robotics pipelines introduces a critical tension between computational efficiency and real-world policy robustness. While DR enhances sim-to-real transfer by training policies across randomized dynamics [28, 25] , its computational overhead and potential over-regularization require careful consideration. This section synthesizes recent advancements to address these challenges.

Computational Overheads in Traditional Domain Randomization Traditional DR methods randomize *all* dynamics parameters (e.g., friction, material properties, sensor noise) uniformly during training [28, 25]. While effective, this approach scales poorly with:
- **Parameter dimensionality**: Training time grows exponentially with the number of randomized parameters [71, 26]. - **Exploration complexity**: Policies must adapt to a combinatorially

large space of dynamics variations, delaying convergence [27, 72]. - **Over-regularization risk**: Excessively broad randomization can dilute task-specific learning, leading to suboptimal policies [73].

For instance, Muratore et al. [71] demonstrated that naively expanding DR parameter ranges increases sample complexity by 3–5× in tasks like robotic grasping.

Balancing Randomization and Performance Modern DR frameworks address this trade-off through strategic parameter selection and adaptive training:

1. **Focused Randomization**:

   Prioritize parameters with high real-world uncertainty (e.g., friction coefficients over gravity) [28, 25]. Let

   $$\mathscr{P} = \{p_1, \dots, p_n\}$$

   denote dynamics parameters, and

   $$\mathscr{I}(p_i)$$

   represent their real-world variance. The training objective becomes:

   $$\max_{\theta} \mathbb{E}_{p_i \sim \mathscr{U}(\mathscr{I}(p_i))} \left[R(\theta; p_i) \right],$$

   where $\mathscr{U}$ samples parameters proportional to their real-world variability [71].

2. **Sequential and Curriculum-Based DR**:

   Methods like Continual Domain Randomization (CDR) [27] decompose training into stages, gradually introducing parameters:

   - **Stage 1**: Train in nominal simulation.

- **Stage 2**: Randomize dominant parameters (e.g., actuator dynamics).

- **Stage 3**: Introduce secondary variations (e.g., sensor noise).

  This reduces peak exploration complexity by 40–60% compared to full randomization [27].

3. **Bayesian Adaptive DR**:

   Bayesian Domain Randomization (BayRn) [71] optimizes parameter distributions using real-world feedback:

$$\phi^* = \arg\max_{\phi} \mathbb{E}_{\xi \sim \nu(\phi)} \left[J_{\text{real}}(\pi_\theta) \right],$$

   where ϕ parameterizes the simulator distribution ν. This focuses training on high-impact parameters, cutting wasted iterations by 30% [71].

Key Metrics for Efficiency-Performance Trade-offs

Table 2.10 Key Metrics for Efficiency-Performance Trade-offs

Metric	Description	Impact on Overhead
Training time	Wall-clock hours to converge	Directly reduces

Metric	Description	Impact on Over-head
Sample efficiency	Episodes required for policy stability	Indirectly re-duces
Real-world success rate	Policy performance post-transfer (e.g., grasp success %)	Prioritizes quality
Parameter relevance score	Correlation between randomized parameters and real-world variability [28, 71]	Guides opti-miza-tion

Practical Recommendations

1. **Parameter Prioritization**: Use real-world system identification to rank parameters by variability [28, 25].

2. **Adaptive Scheduling**: Implement CDR or BayRn to phase in parameters [27, 71].

3. **Validation-Driven Early Stopping**: Terminate randomization phases once policy performance plateaus on held-out test domains [73].

Recent benchmarks show that adaptive DR achieves 85–92% real-world task success with 50% lower compute than uniform DR [27, 71], proving that strategic randomization—not maximal randomization—bridges the sim2real gap efficiently.

Conclusion This analysis underscores that computational overheads in DR are not inherent but stem from suboptimal implementation. By aligning randomization with real-world priors and adopting adaptive frameworks, practitioners achieve robust policies without prohibitive costs.

2.4.2 Domain Adaptation

In this section we present a set of techniques for sim2real that attempt an explicit so called "domain adaptation", namely from the simulation domain to the reality domain, usually by utilizing an explicit mapping in the the space of the residual unmodelled dynamics and/or through learning an invariant set of features, which should allow for a direct sim2real transfer.

2.4.2.1 Domain Adaptation for Synthetic2Real

Bousmalis et al. [74] attempt to learn such invariant features that are transferable from one domain to another by learning a "private" set of features for each domain which encodes essentially what makes each domain unique (see fig. 2.11).

An interesting implication of this approach would be a direct translation of rendered images in simulation to realistic in "synthetic-to-real" scenarios.

Bousmalis, Irpan et al. [75] attempt to train a grasping network by enriching the training dataset with synthetic images resulted in simulation and which are consequently adapted in order to resemble the realistic setup (see fig. 2.12).

This pixel-level domain-adaptation for synthetic-to-real images is learned through a GAN based procedure which results in a "generator" as a fixed module in the pipeline [76]. A

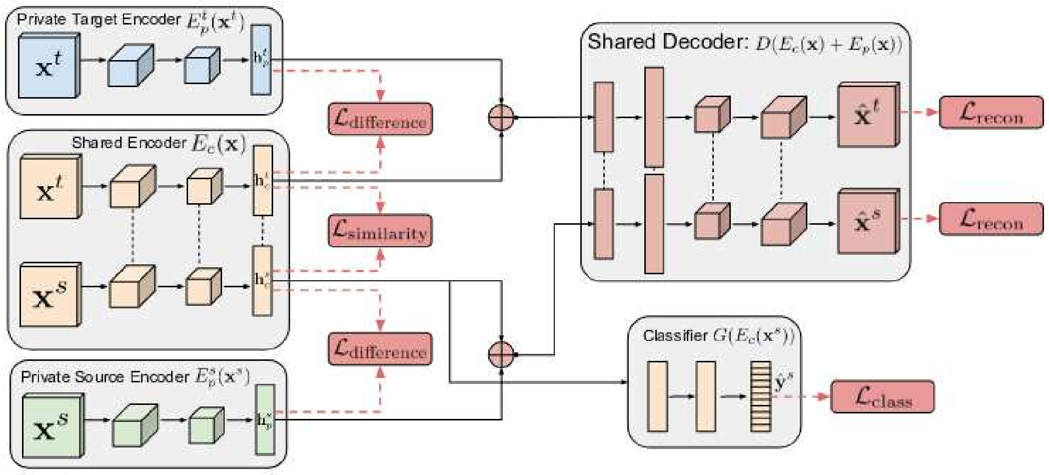

Private Target Encoder $E_p^t(\mathbf{x}^t)$
$\mathbf{x}^t$
h_p^t
Shared Encoder $E_c(\mathbf{x})$
$\mathbf{x}^t$
h_c^t
$\mathbf{x}^s$
h_c^s
Private Source Encoder $E_p^s(\mathbf{x}^s)$
$\mathbf{x}^s$
h_p^s
$\mathcal{L}_{\text{difference}}$
$\mathcal{L}_{\text{similarity}}$
$\mathcal{L}_{\text{difference}}$
Shared Decoder: $D(E_c(\mathbf{x}) + E_p(\mathbf{x}))$
$\hat{\mathbf{x}}^t$
$\hat{\mathbf{x}}^s$
$\mathcal{L}_{\text{recon}}$
$\mathcal{L}_{\text{recon}}$
Classifier $G(E_c(\mathbf{x}^s))$
$\hat{\mathbf{y}}^s$
$\mathcal{L}_{\text{class}}$

Figure 2.11 Domain Separation Networks Architecture [74]

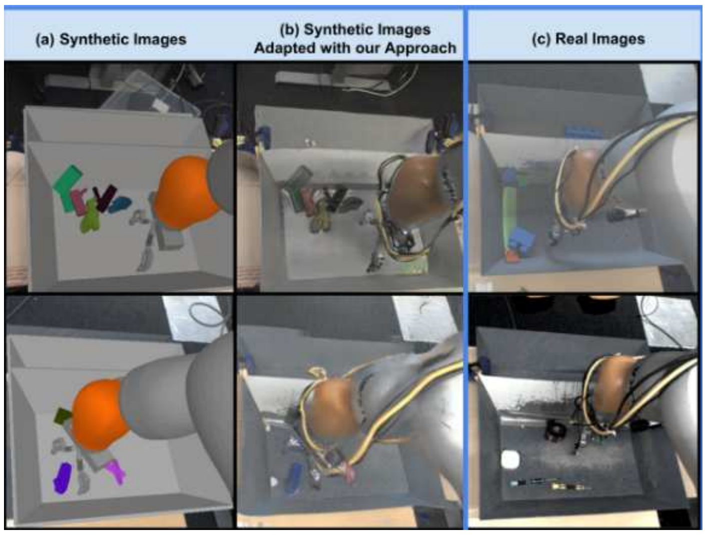


Figure 2.12 Training a grasping network with adapted and real images [75].

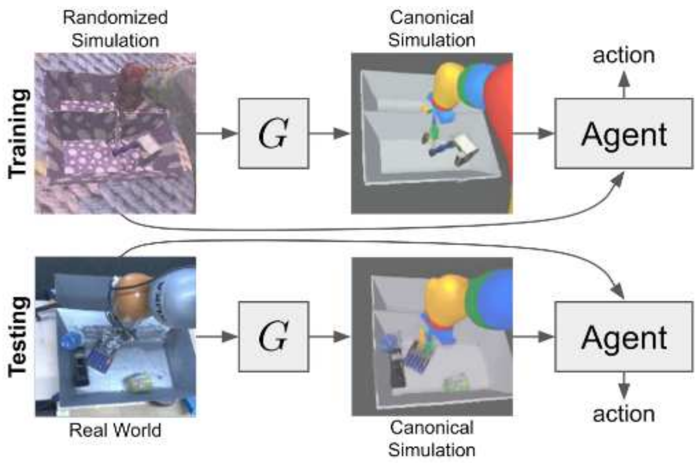


Figure 2.13 The generator G is able to translate images from real2sim[79]

feature level adaptation module [77] is consequently applied for both images and motor level commands.

Combining Pixel-Level Adaptation with Feature-Level Domain Adaptation is attempted by Park et al. [78] as well as an effort to minimize the use of real-world data.

James et al. [79] moving in the opposite direction learn a generator that maps “disrupted-randomized” images to simulated ones which in turn are observed by the agent. This generator is able to generalize to real images and thus mapping them to simulated ones, in a real-to-synthetic direction, which is a form that the agent can operate on as it is canonical to its training set.

Bharadhwaj et al. [80], instead of translating to a canonical simulated image, attempt to directly encode real images in the same way as they would be decoded if they where synthetic and thus allowing the learned planner to execute robustly and accordingly. The feature encoding is performed through a discriminator and an adversarial domain transfer. A final fine-tuning step with real world expert demonstration completes the pipeline.

Zhang et al. [81] also similarly attempt an adversarial transfer for reducing the amount of real labeled data needed for fine-tuning when transferring to real.

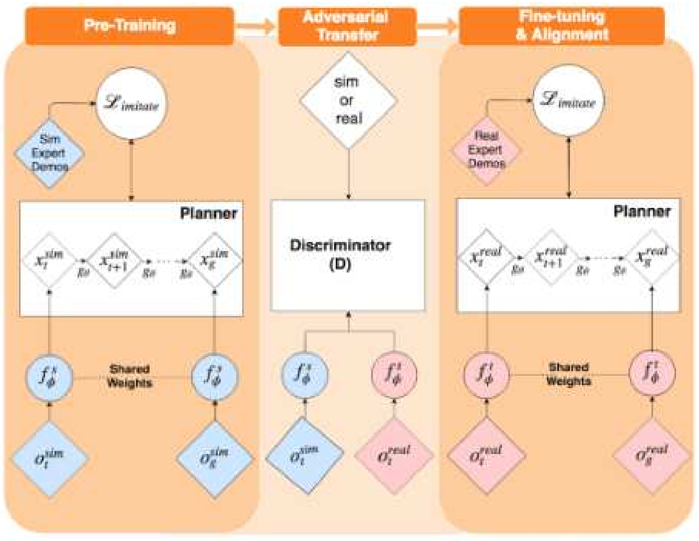


Figure 2.14 Adversarial adaptation for shared feature encoding for sim and real[80]

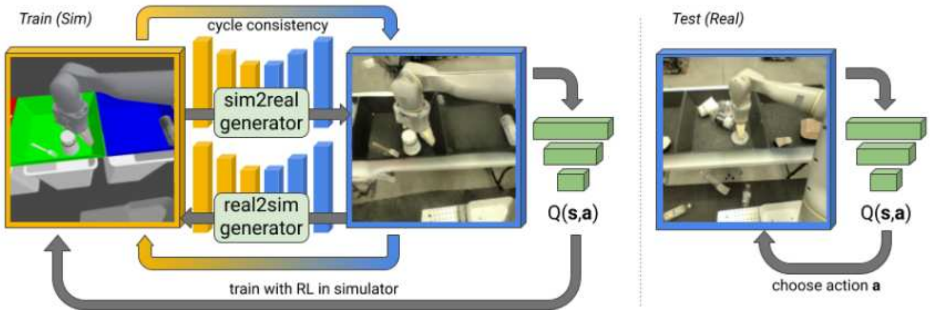


Figure 2.15 RL Q Value Consistency Criterion ensures task relevant image generation [82]

2.4.2.2 Task Aware Domain Adaptation

Rao et al. [82] make the critical observation that the domain adaptation techniques, as described in the previous subsection, are task-agnostic and consequently do not necessarily encode task critical information. The propose solution in this case is the introduction of a loss which can guarantee a translation operator which is invariant with respect to the Q Values for the image (see fig. 2.15).

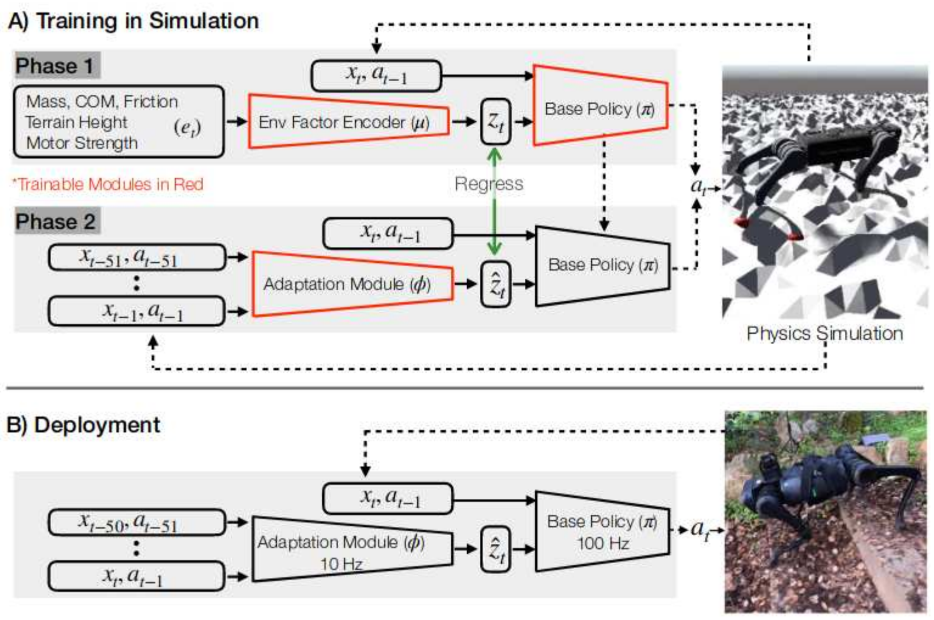


Figure 2.16 Rapid Motor Adaptation Architecture [83]

2.4.2.3 Adaptation Module for Unmodelled Dynamics

Kumar et al. [83] based on the idea that extrinsic system dynamics are encoded in the history of proprioceptive information as state-action pairs [84, 38], they train an adaptation module in simulation that learns to predict the latent extrinsics vector z_t which can be then be directly deployed to the real-world setup (see fig. 2.16).

Qi et al. [85] apply the exact same principle for the task of in-hand object rotation where the adaptation module allow for generalization to a diverse set of objects where the agent has not been trained on.

2.4.2.4 Progressive Nets

Rusu et al. [86] propose the utilization of “Progressive Nets” (see fig. 2.17) as an effective way to utilize the knowledge learned in simulation in reality. Progressive Nets allow a direct access and reuse of the feature space of one domain to another rendering it an excellent candidate for a sim2real scenario. This concept can be arbitrarily expanded in order to

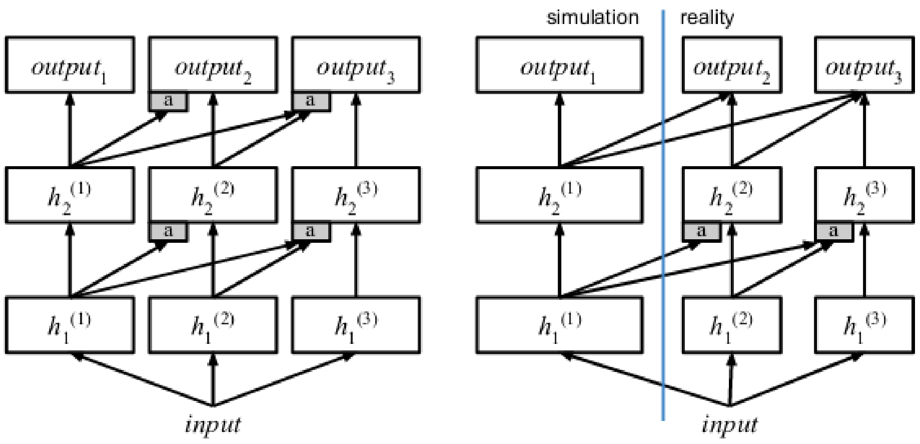


Figure 2.17 Sim2Real through a modified Progressive Network [86]

accommodate multi-task and life-long learning with the additional benefit of allowing for different input multi-modalities as the columns allow for heterogeneous data structures.

2.4.3 Real2Sim - Real World Data Utilization for Sim2Real

Utilizing real-world data for fine-tuning the simulation framework is one of the most predominant techniques for Sim2Real, as closely matching the real-world dynamics and phenomena allow for a seamless transition for an agent trained in simulation to the real-world.

This simulation/model parameter tuning is well studied problem and widely applied in control engineering, traditionally under the “System Identification” nomenclature.

We see that an incorporation of real-world data for simulation/model fine-tuning within a Reinforcement-Learning Sim2Real context is an integral part of numerous lines of works, mostly in order to match the actuation dynamics, whose predictable behavior in simulation is crucial for successful Sim2Real transfer.

2.4.3.1 Traditional System Identification (SysId)

System identification methodologies are commonly applied in many research works as an attempt to minimize the Sim2Real gap by matching real-world dynamics.

Tan et al. [58] proceed in an analytical system identification which includes a disassembly of the Minitaur robot for measuring the dimensions and inertia characteristics which are then incorporated to the URDF file. They additionally conduct experiments for identifying the motor friction characteristics.

OpenAI [16] in their influential work in the Sim2Real field for manipulating a Rubik's Cube with a Dexterous Robot Hand, also apply system identification methodologies for matching the coupled joint dynamics to the real-world configuration by minimizing the root mean square error between simulated and real joint trajectories. Similar system identification approaches can be found in [59, 46]

Zhang et al. [87] also employ system identification for matching the real-world dynamics with the simulation, by utilizing the the computational speed of Julia and the Lyceum MoJoCo wrapper.

2.4.3.2 Identifying Actuation Dynamics

Hwangbo et al. [61] in order to model the intricate actuator net (software/hardware) dynamics for a quadruped robot they train an "actuator-net" with real-data for this purpose. This "actuator-net" is utilized in simulation jointly with the stochastic rigid-body model of the robot for training a policy. This policy is then directly deployed to the real-robot successfully.

2.4.3.3 Action Space Alignment for Real-World Deployment

Wu et al. [55] for learning the task of deformable objects manipulation they incorporate real-world images of cloths in their training data-set. Additionally, as their action space is defined as points in the image space, they need to align those points to with the real-world robot actions. For doing so they collect a small number or data-points for linearly mapping the robot coordinates to the image pixel locations.

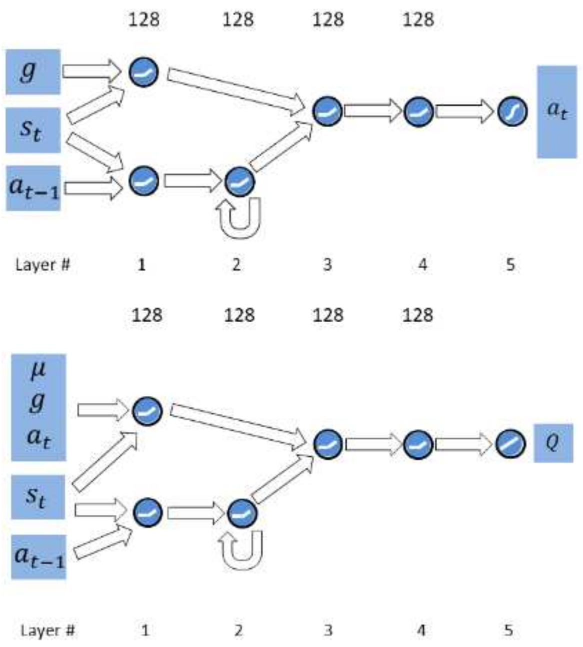


Figure 2.18 State-Action pair for implicit SysId with recurrent policy and value network architectures [38]

2.4.3.4 Implicit SysId Policy Embedding

As covered previously in the domain-adaptation section for sim2real, it is possible to infer information about the system dynamics based on the short history of state-action pairs, as a sort of an on-line system identification procedure, which benefits the overall policy's robustness and generalization capabilities.

As Peng et al. [38] note though this approach might impose limitations in more complex systems. They propose an implicit embedding of SysId through a recurrent policy model where the internal memory encodes the information contained in the history of state and actions, namely some useful information about the system dynamics. The authors therefore pass through the recurrent policy (LSTM) only the features that are relevant to system dynamics, whereas they propagate through forward networks all the rest for the policy and value networks respectively (see fig. 2.18).

2.4.3.5 Simulator Tuning for Sim2Real

Real-World data have been utilized in numerous research works in order to tune the simulation parameters in order reduce the Sim2Real gap. For the most part, and as traditionally there have been non-differentiable rigid-body physics engines mostly available, parameter tuning has been relying on gradient-free methodologies, such as Bayesian optimization and Evolutionary Algorithms [36].

Chen et al. [41] use such an evolutionary search algorithm (CMA-ES [88]) for searching a set of optimal parameters in order to approximate as much as possible the real-world reference trajectories. This process is performed with massive GPU based parallelization, which speeds up the search, taking into advantage the inherit capabilities of the NVIDIA's Isaac Gym physics engine.

Collins et al. [36] attempt parameter tuning with Differential Evolution for a set of 5 physics engines including PyBullet, PyRep and V-Rep versions. Their strategy based on this special version of Evolutionary Algorithm seems to be improving the performance across all physics engines tested. Their methodology allowed an evaluation of the importance of specific parameters, which they found to be the simulation time-step, lateral object friction and joint maximum velocity.

Du et al. [89] attempt to search for for best fitting parameters for the simulation based only on **pure RGB observations from the real-world** (see fig. 2.19). They formulate the parameter tuning as a search problem, and they train a model as a binary classifier which predicts the probability that the true parameters that generate the current trajectory are either lower or higher than a given value.

2.4.3.6 Hybrid Sim & Real Reinforcement Learning

Kang et al. [90] attempt to learn a real-world control policy with simulated data and a limited amount of real-world data. The authors suggest that the real world data and simulated data

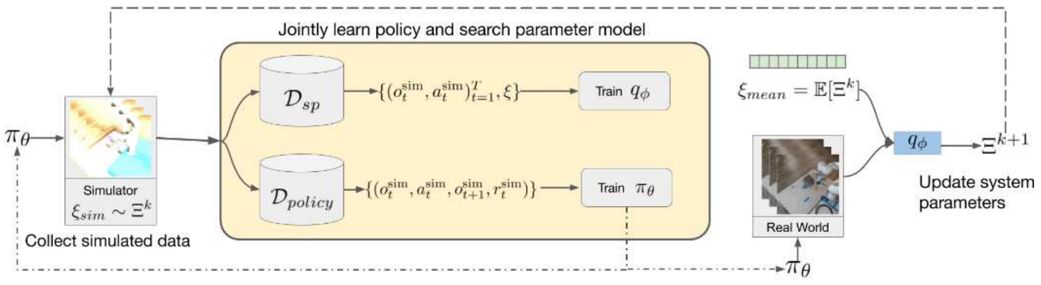

Jointly learn policy and search parameter model
$\mathcal{D}_{sp}$
$\{(o_t^{\mathrm{sim}}, a_t^{\mathrm{sim}})_{t=1}^{T}, \xi\}$
Train q_ϕ
$\mathcal{D}_{policy}$
$\{(o_t^{\mathrm{sim}}, a_t^{\mathrm{sim}}, o_{t+1}^{\mathrm{sim}}, r_t^{\mathrm{sim}})\}$
Train π_θ
π_θ
Simulator
$\xi_{sim} \sim \Xi^k$
Collect simulated data
$\xi_{mean} = \mathbb{E}[\Xi^k]$
q_ϕ
Ξ^{k+1}
Update system parameters
Real World
π_θ

Figure 2.19 Simulation Parameters Tuning based only on RGB observations from the real-world [89]

are offering complementary qualities, namely accuracy for real and diversity for sim. Thus they proceed in learning the dynamics of the robot from the real-world data and leveraging the simulation for learning a generalizable visual perception system. More concretely the separately train a task-specific perception module in simulation which in turn they fix and transfer it in the real-world setup for training the residual agent, which is based on an LSTM network. Interestingly enough, due to the limited real-world data, the agent learns to predict future rewards based on actions for a limited horizon. The action is picked for maximizing the total horizon rewards in an MPC fashion.

2.4.3.7 Data-Augmentation with Imperfect Sim & Proxy Task Fine-Tuning

Zhang et al. [87] in order to tackle the demanding and delicate task of cherry-picking, which requires a precise, reactive and robust agent, propose a multi-stage learning schema which starts from training from a small amount of trajectories from a hand-written heuristic controller, continues with enriching the sample trajectories with off-line simulated ones, which are also domain-randomized, and concludes with a final fine-tuning in the real-world on a “Proxy” task which is easy to repeat using an asynchronous updating policy. The perception module is based on estimating the object centroid position from an Azure Kinect RGB-D camera.

2.4.3.8 Online Iterative Simulation Tuning and Policy Learning Tuning - Real2Sim2Real

Online BayesSim Possas et al. [91] use probabilistic inference to learn distributions over simulation parameters conjointly with the controller in an end-to-end fashion for the task of autonomous driving. They utilize likelihood-free inference (LFI) for Bayesian analysis and they solve the inherent problem of high-conditionality for the state-action pairs by embedding them through a Recurrent Neural Network which functions as a Mixture of Gaussians estimator. This on-line tuning of the simulation parameters leads in increased performance after only some rollouts in the real-world setup.

Grounding Simulation Learning (GSL) Farchy et al. [92] introduce the Grounding Simulation Learning (GSL) algorithm for iteratively learning in simulation, deploying in to the real world and finally utilizing this real-world experience for improving or "grounding" the simulation in order to the consequently improve the policy through a more accurate and consistent with reality training procedure. The authors report an overall 25% increase in the walking speed of the Nao robot by utilizing this GSL approach.

Grounding Action Transformation (GAT) Hanna & Stone [93] inspired by the aforementioned GSL algorithm they propose the Grounding Action Transformation (GAT) algorithm as an attempt to eliminate some fundamental limitations of GSL, most notably that the action dynamics do not seem to be accounted in simulation. For this they introduce the modification function g (see fig. 2.20), which conjointly learned, in order to estimate which action in simulation will bring the system to the state that it would have been observed in reality. This time the authors suggest a 43% increase in velocity again for the Nao robot.

Stochastic Grounded Action Transformation Desai et al. [94] inspired by [93], they extend the GAT algorithm in order to account for the inherit stochasticity than naturally the real-world setup embeds. This time the f_{real} (see fig. 2.20) predicts a distribution over the

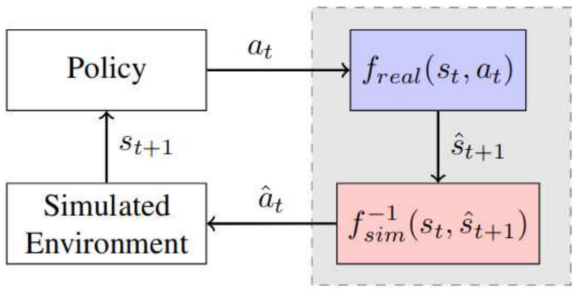


Figure 2.20 Grounding Action Transformation (GAT) modification function *g* [93]

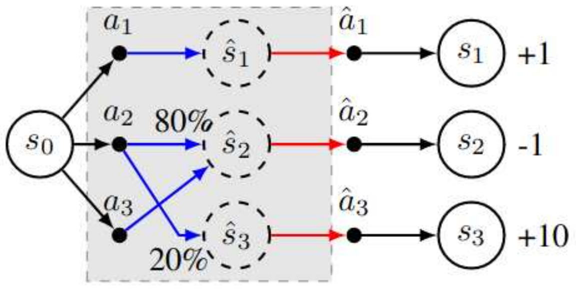


Figure 2.21 In the SGAT algorithm the f_{real} learns to predict a distribution over the next states [94]

next states instead of the "most likely next state" (see fig. 2.21). This approach when applied again to the Nao robot demonstrates improved robustness during walking in uneven terrains, when compared to GAT.

Mutual Alignment - Transfer Learning Wulfmeier et al. [95] propose a "mutual-alignment" solution (see fig. 2.22) between simulation and real-world training based on "auxiliary" rewards that guide the exploration in the real-world based on performance in simulation. This approach aims to increase sample efficiency in the real-world by harnessing learning in simulation in parallel.

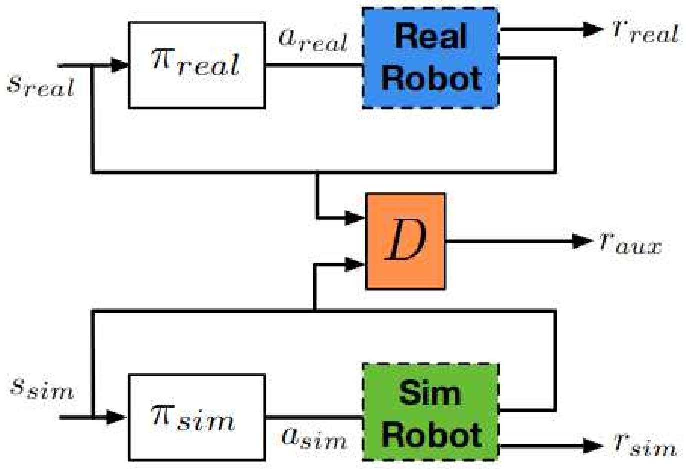


Figure 2.22 Mutual Alignment Transfer Learning [95]

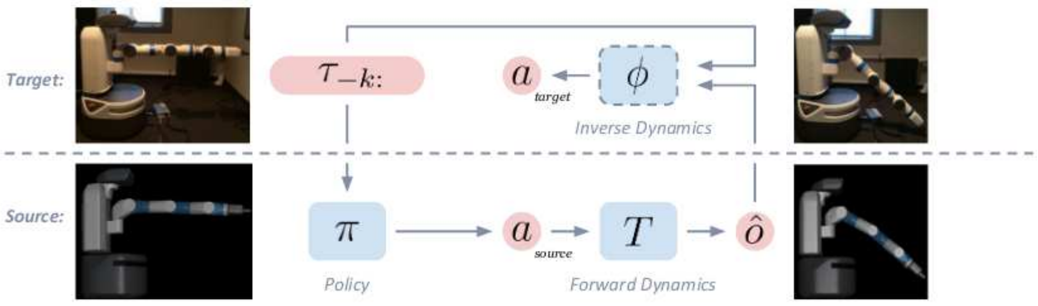


Figure 2.23 Learning Inverse Dynamics [96]

2.4.4 Learning Inverse Dynamics for Sim2Real

By learning an inverse dynamics model it is possible to explicitly account for the discrepancy between the simulated and real-world dynamics by learning to predict the actions that would lead to a desired state as dictated by the source (simulation) domain [96, 97].

This concept is inversely related to the Grounding Action Transformation (GAT) presented in the previous section “Real2Sim - Real World Data Utilization for Sim2Real”.

2.4.5 High-Fidelity Simulation

Intuition suggests that a high-fidelity simulation framework with a photo-realistic rendering pipeline and highly precise dynamics closely matching the real-world system dynamics should be beneficial for learning based methodologies and a sim2real transfer. Indeed this has been suggested in the literature numerous times. Alghonaim and Johns [57] and as pointed by Muratore et al. [21], they conclude that "a small number of high-quality images is superior to a large number of low-quality images". The simulation accuracy has been also a point of debate in [19] where it was pre-juxtaposed in the conversation that a highly accurate simulation of reality "remains a pipe-dream", as everyone agreed that this would be arguably beneficial overall, where "there is no alternative to accurate simulation" and generally there has been strong argumentation against the "imprecise simulation". On the same line of thought [18, 20] also points that "simulators improvement is an essential requirement" as even robust controllers would struggle matching the dynamics discrepancy.

Lastly photo-realistic rendering in simulation has been the objective of numerous research works as sim2real transfer measure [98, 76] in order to train closer to reality.

2.4.5.1 The Case Against High-Fidelity Simulation

Although is seems that a general consensus has been reached, namely that a high-fidelity simulation is a desired property, Truong et al. [99] challenge this core belief. They conclude that higher simulation fidelity "does not enable learning better high-level visual navigation policies" and that "dynamic policies tend to overfit to low-level simulation details". They also mention about the good transfer characteristics of the kinematic policies, as we analyse in section about "expicit transferable abstractions".

2.4.6 Hierarchical Reinforcement-Learning

One of the most important characteristics of the Machine Learning Breakthroughs was the unsupervised, auto-built encoding of feature hierarchies that was taking place during training [12]. One of the most important inherit architecture properties with the introduction of the Convolution Neural Networks (CNNs) for Computer Vision, it was the feature hierarchies, from low level edge detection, to gradually shapes and objects, which was in close alignment to the more traditional, manually crafted feature engineering that was taking place before. This explainability layer of the presented features allowed for better understanding and easier mass adoption into commercial applications.

Although this has been the norm with the Computer Vision pipelines, in the realm of Reinforcement Learning for continuous control for dynamic systems we haven't seen such system hierarchical learning and for the most part the policy implementations remain an uninterpreted black-box which severely hinders targeted scientific and engineering advancements and a consequent mass adoption in commercial applications.

Within the context of a Sim2Real pipeline for robotics such a hierarchical layering would allow for a system decomposition where the high-level abstract layers could reason seamlessly between simulation and reality and, with the low-level layers being susceptible to the sim2real gap.

Such decomposition is in essence at the core of the "domain-adaptation", "explicit transferable abstractions" where such encoding is attempted either at the observation or the action space. A holistic framework for Reinforcement Learning that would allow a concrete fundamental handling as such is still largely missing and till recently it has been underexplored with potentially the fields of "metal-learning" and "multi-task" learning driving towards this same goal.

2.4.6.1 Multi-Task and Multi-Agent Systems

Shu et al. [100] propose a framework for hierarchical policies which autonomously decide between exploitation of a previously learned policy and learning a new skill.

Julian et al. [101] propose skill decomposition as a direct solution to the sim2real transfer, where they explicitly learn a set of "high-level" which in turn control "low-level" tasks (see fig. 2.24).

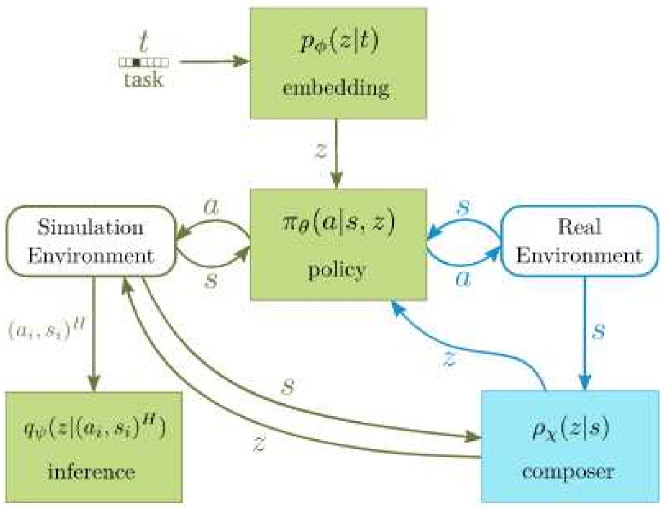


Figure 2.24 Hierarchical skill decomposition for sim2real transfer [101]

Wu et al. [55] learn a composite action space for pick-and-place where they factor the policy as:

$$\pi_{\text{factor}} \equiv \pi_{\text{pick}}(a_{\text{pick}} \mid o) \cdot \pi_{\text{place}}(a_{\text{place}} \mid o, a_{\text{pick}})$$

Chen et al. [102] combine multiple dexterous sub-policies for dexterous manipulation by learning a “Transition Feasibility Function” (see fig. 2.25) which is responsible for an optimal “on-the-fly” of these. The authors report a “zero-shot” transfer to the real-world.

2.4.6.2 Decomposing the “What” from the “How”

A very interesting recent research direction on the problem of skill decomposition and hierarchical learning structures is a problem formulation that examines separately the “What” from the “How”[103]. This approach can be applied in a sim2real transfer scenario, where the simulation environment is responsible for the “what” needs to be happened as an abstract task long-horizon planner, without the “how” specifics which can be then analysed separately in a later stage of the overall pipeline.

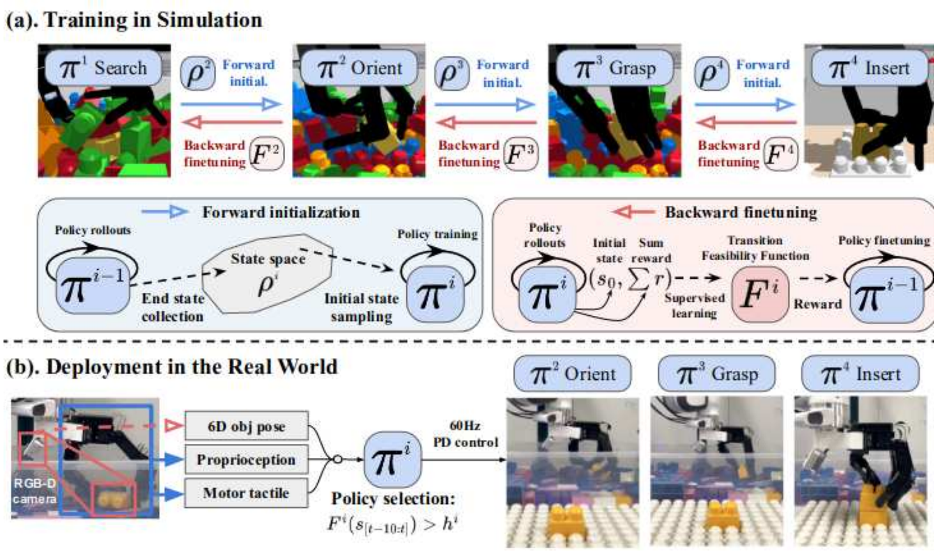


Figure 2.25 Sequential Dexterity for Long-Horizon Manipulation [102]

2.4.7 Explicit Transferable Abstractions in Perception

By the term "explicit transferable abstraction" we refer to states or features that have the property of one-to-one correspondence between simulation and reality. Usually these refer to either (most usually) to the perception part and the observation space or the action space and include pure geometric characteristics or a specifically designed feature space for a direct sim2real correspondence. This characteristic property often times allows for a zero-shot sim2real transfer.

2.4.7.1 Manual Observation Space Design

We proceed with a listing of some influential research works that make use of such representation for the observation space, enabling a direct sim2real transfer.

Table 2.11 Examples of manually crafted “explicit transferable abstractions” for zero-shot Sim2Real transfer.

Explicit Abstraction
Image Segmentation Mask and current configuration of the robot arm. Yan et al. [104]
Depth only readings from a depth camera, omitting the RGB component which would hinder a direct sim2real transfer. Breyer et al.[50] Chen et al.[105]
Bounding Boxes for representation of obstacles. Zhang et al. [106]

2.4.7.2 Auto Feature Space Shaping for Sim2Real Transfer

Alternatively to a manual approach in feature shaping as an explicit sim2real transferable abstraction, it is also common to auto-encode such representation with a sim2real discrepancy minimization criterion as an encoding guidance.

Table 2.12 Examples of auto-encoded “explicit transferable abstractions” for zero-shot Sim2Real transfer.

Explicit Abstraction
An intermediate representation of the observation is specifically designed for consistency between simulation and reality for autonomous highly dynamic drone navigation. Kaufmann et al. [107]

Explicit Abstraction
A “hybrid discrete-continuous action representation for temporally-abstracted and spatially-grounded object-centric action representation” is learned for 6D Non-Prehensile Manipulation. Zhou et al. [108]
A continuum state representations is defined with “hand-crafted numerical states to encoded image-based representations, with decreasing levels of induced task-specific knowledge”. Petropoulakis et al. [109]

2.4.8 Privileged Information Training

One of the most important aspects when working with a simulated environment is access to the full state. This can be exploited in order to train the value function with the privileged information present in simulation and the policy function independently with the observation that is actually gonna be present in the real-world. This “asymmetric actor-critic” [53] framework has been part of numerous sim2real approaches [21, 110, 105, 111].

2.4.9 Imitation Learning & Human-in-the-Loop

In this section we examine the role of imitation learning and Human-in-the-Loop within the scope of sim2real transfer. A very common usage of imitation learning schemes is the inclusion of those as initial prior “seeds” for learning acceleration, especially under the presence of sparse rewards [21, 54]. These demonstration can be either from human “teachers” or from manually crafted controllers that take advantage of the privileged information present in simulation.

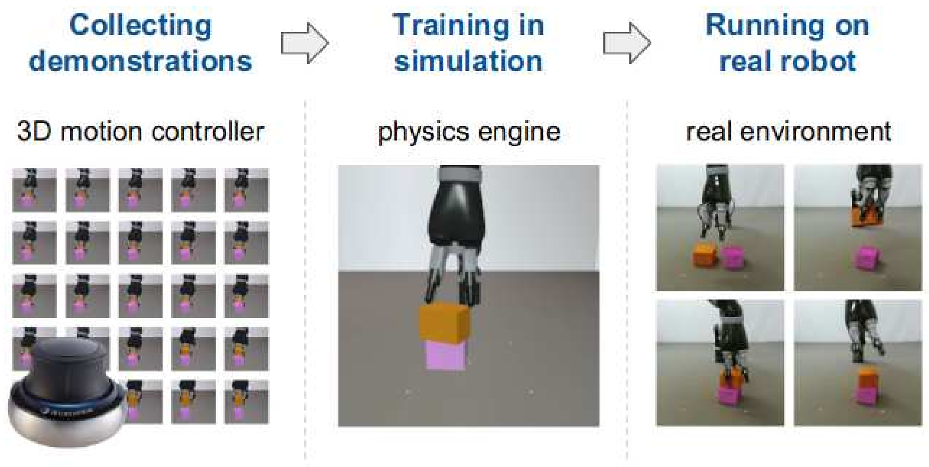


Figure 2.26 Collecting demonstration priors for consequent training in simulation [112]

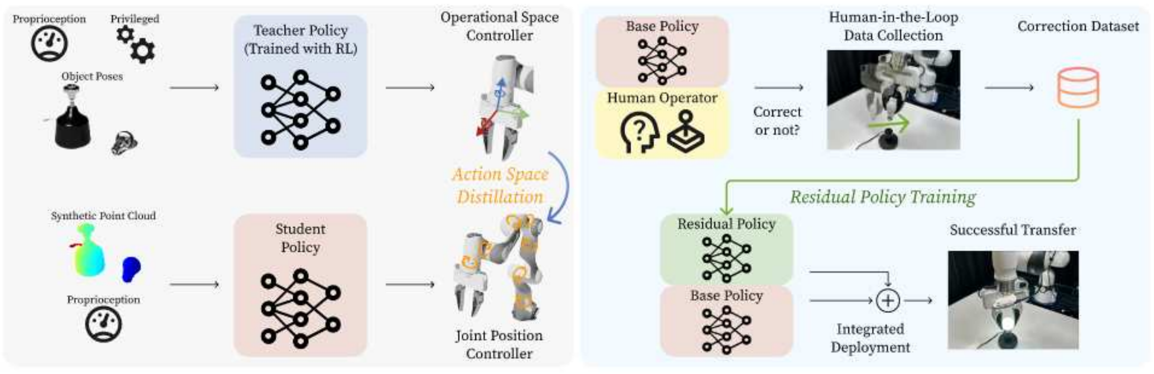


Figure 2.27 Human operator for residual policy training with TRANSIC [113]

Li et al [62] use flipping demonstrations for inferring the task reward function and consequently learning an agent by training in simulation.

Jiang et al. [113] with the “TRANSIC” framework employ a “human-in-the-loop” framework for enabling a sim2real transfer. They are doing so by allowing human users to augment and correct simulation policies in order to overcome any sim2real gaps.

2.4.10 Multi-Task Generalist Agents and Meta-Learning

As defined by Caruana [114] “Multitask Learning is an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias”. By using a shared-shaped representation the agent is force to encode optimally features that are able to readily transfer to new unforeseen tasks [24]. Multitask learning has received a lot of attention in the research community.

This has immediate implications for Sim2Real transfer as a well trained generalist agent, that has properly embedded transferable features, could easily adapt to a deployment into the real-world.

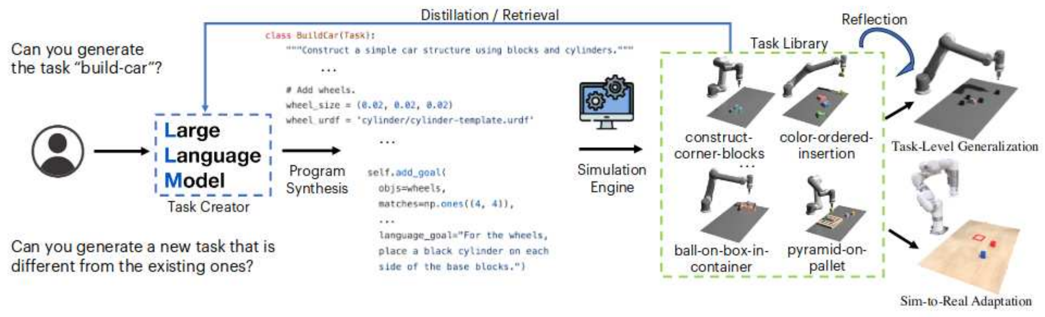
We can observe direct connections to domain-randomization, where in the first case the agent implicitly is forced to become a "generalist" with the exposure to randomized environments. This has been observed and analysed by OpenAI [16] where the claim that this exposure to diverse environments with the simultaneous use of a recurrent neural network (LSTM) as an agent architecture gave rise to emergent metal-learning properties. More concretely the authors in that case claim that this policy implicitly learned to update its belief about the true transition probability $P(s_{t+1}|s_t, a_t)$.

Nagabandi et al. [115] introduce an online adaptation scheme to new tasks by incorporating meta-learning to train a dynamics model in way that it can rapidly adapt with the introduction of new observations. The authors report remarkable robustness to even "crippled body parts" where the agent is able to quickly online adapt it's behavior.

2.4.10.1 LLMs - Foundation Models inspired Meta-Learning

With the advent of the recent success of LLMs, Visual-Language models and a quest for "foundational models", a lot of research work is focused in elaborating some of these techniques in the realm of robotics.

Wang et al. [51] utilize LLMs for generating a large number of diverse tasks in simulation for training generalist agents which are then able to perform long-horizon tasks in the real-world, as the authors suggest (see fig. 2.28).



```
class BuildCar(Task):
    """Construct a simple car structure using blocks and cylinders."""
        ...
    # Add wheels.
    wheel_size = (0.02, 0.02, 0.02)
    wheel_urdf = 'cylinder/cylinder-template.urdf'
            ...
        self.add_goal(
          objs=wheels,
          matches=np.ones((4, 4)),
          ...
          language_goal="For the wheels,
          place a black cylinder on each
          side of the base blocks.")
```

Figure 2.28 GenSim generates numerous tasks in simulation automatically with the help an LLM [51]

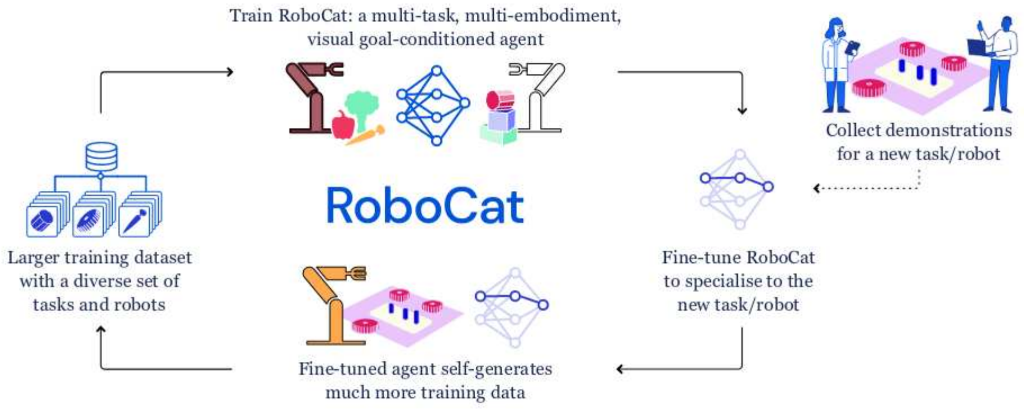

Train RoboCat: a multi-task, multi-embodiment, visual goal-conditioned agent
RoboCat
Collect demonstrations for a new task/robot
Fine-tune RoboCat to specialise to the new task/robot
Fine-tuned agent self-generates much more training data
Larger training dataset with a diverse set of tasks and robots

Figure 2.29 RoboCat as a "multi-task, multi-embodiment, visual goal-conditioned agent that can self-improve" [117]

More recently "multi-model", "multi-task" and "multi-embodiment" generalist agents started to appear [116, 117] as clear trend towards foundational models for robotics setting a clear research trend.

2.4.11 Curriculum & Lifelong Learning

In this section we are interested in how "curriculum" and "life-long" learning techniques can be implemented and incorporated for sim2real transfer.

Curriculum learning is commonly incorporated in Reinforcement Learning as it allows for a gradual increase in difficulty by changing the simulation environment characteristics. This approach allows for better convergence, especially in the presence of sparse rewards

[110]. For Sim2Real transfer curriculum learning has an immediate application where the real-world environment could be seen as another evolution of the difficulty level of the environment [118].

Life-long learning is also a widely researched topic with also an immediate application on Sim2Real where such an approach would allow for learning to continue after deployment in the real-world and thus auto adapt and fine-tune.

2.4.12 Sim2Real through Embodied Intelligence

An aspect that is generally overlooked under the Sim2Real prism is that of the affect of the mechanical design of the robotic setup in the overall transfer from simulation to reality.

Dao et al. [70] **make the crucial and usually very unexplored point with regards to the effects of mechanical design, and most importantly the effects of compliant elements**, in this case springs, where they are able to transfer to real due to the compliant mechanics.

Similarly [41] note that the use of soft fingertips allow the system to be naturally more compliance under it's collision/contact/friction dynamics in combination with the manipulated object, increasing the degree of tolerances and thus the chances of a successful sim2real transfer.

The compliance effect for sim2real transfer was also noted by Beddow et al. [43] where their usage of soft fingers "as their compliance inherently smoothed the forces" which is closer to the MuJoCo soft constraints physics.

Compliance seems therefore to be an important factor for sim2real, as suggested by the literature, and a more holistic approach that encompasses agent design conjointly with the actual robotic mechanical design under an embedded intelligent design criterion should be more thoroughly and formally researched.

2.4.13 Zero-Effort Sim2Real

In numerous research articles there is an explicit mention of a “zero-shot” transfer from Sim2Real which can be interpreted as due to the utilization of an effective Sim2Real strategy that it did not require a further adaptation, fine-tuning down the line, or simply an inherit property of the architecture **and** methodology that simply allowed for an “effortless” sim2real transfer without any explicit sim2real technique.

2.4.13.1 Zero-Shot Generalization

Kansky et al. [119] introduce the “Schema Networks” which are able to learn the underlying world model dynamics and consequently exploit them in order to achieve the desired goals. This understanding of the world-model allow for a zero-shot generalization as the authors suggest in “Sim2Sim” arcade game scenarios.

Valassakis et al. [120] try to analyse the “zero-shot” transfer criteria and requirements for a “zero-shot” sim2real transfer. They focus in transfer of the system dynamics and for that they employ again explicit transferable abstractions in the perception and proprioception observation space. They demonstrate good sim2real transferability under adversarial randomization training in simulation.

2.4.13.2 Zero-Effort as an Inherit Architecture Property

We observe that usually when there seems to be a “zero-effort” sim2real transfer, it usually incorporates an architecture that includes explicit transferable abstractions in the observations space, either through manual feature assignment or some auto-encoding scheme which favors spacial and non-dynamic characteristics. We analyse some of these “explicit transferable abstraction” related works in the corresponding section.

2.4.14 Real2Real (SimFree)

This section showcases a line of work that **does not** utilize any form of simulation for training, but it rather learns directly from real-world demonstrations, in a sample-efficient manner, through some form of imitation learning scheme. This, rather provocatively, raises the question on the ultimate necessity of training in simulation, at least for some cases of robotic employment scenarios [121–124, 23, 125].

2.5 Physics Engines & Simulation Frameworks for Sim2Real

Physics Engines and the Simulation Frameworks for Robotics wrapped around the former lie at the core of the Sim2Real pipeline. Physics Engines and the surrounding software tooling for rendering has been predominantly a product of the 3D gaming explosion of the 90s. This fundamental development timeline for the gaming industry usually meant that the physics should mostly look realistic enough, be robust and stable as well as fast to compute. This of course creates fundamentally a Sim2Real discrepancy. Despite that physics engines such for rigid body dynamics, such as the Open Dynamics Engine [126] or Bullet [127], have been in common usage among the robotics community even before the Machine and Reinforcement Learning dominance, during the more model-based era. Since the major breakthroughs in Reinforcement Learning for continuous control, which started from evaluation in simple dynamic simulated environments such as arcade video games and toy dynamic 2D environments, we have seen major deployment of those aforementioned rigid-body dynamics engines for training, as they most closely match a real-world setup with potential for a consequent deployment and evaluation (fig. 2.30).

Physics Engines and Simulation Frameworks are increasingly developing in more complex highly sophisticated integral components of the learning community for robotics control

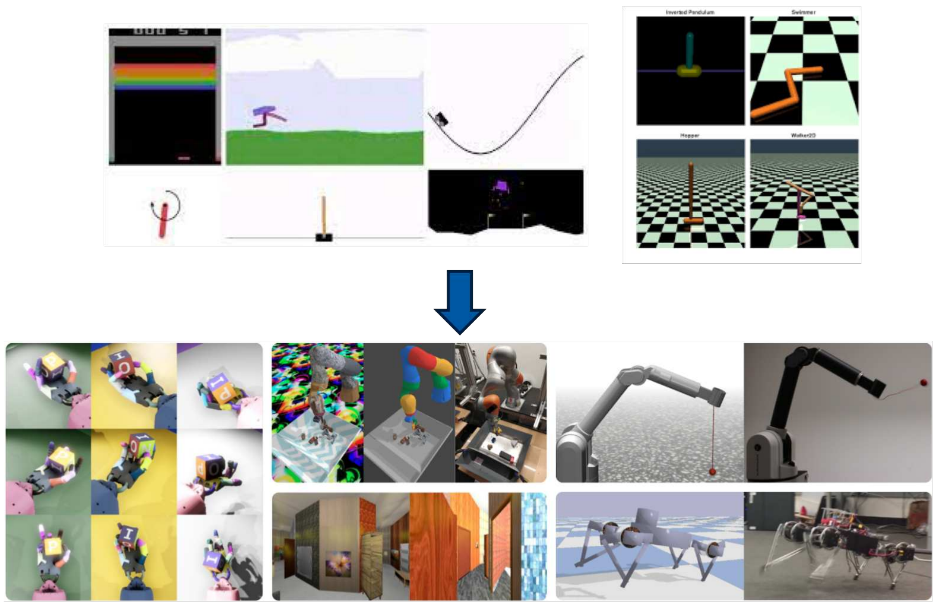


Figure 2.30 Reinforcement Learning Control for Continuous Dynamics. From toy examples in simulation to real-world robotic actuation.

and are gradually advancing and are incorporating advanced dynamic capabilities and rich tool-chains for robotics control.

2.5.1 Choosing a Physics Engine for Robotics Simulation

Choosing a physics engine for training for a robotic application with a sim2real pipeline in mind can be a complex process and requires a lot of parameters that need to be taken into account.

Even starting with typical robotic tasks in non-dynamic situations that involve rigid-body dynamics the differences between physics engines can be significant [128, 21].

Ivaldi et al. [129] have conducted a simulation usage survey among robotics concerned users, with a quite thorough questionnaire at the year 2014. In this work there has been an attempt to infer the most important features and criteria that roboticists prioritize when choosing a physics engine and they have tried to categorize the most predominant physics engines of the era based on this user feedback. Interestingly enough one of the most requested improvement was the realisticity of the simulator, without though differentiating between rendering photorealism and dynamics. Although at the moment 10 years have gone by, it still seems that most the core issues with regard to physics engines, simulation frameworks and robotics research remain.

Erez et al. [130] have made significant efforts to compare between the most dominant rigid-body simulation frameworks Bullet, Havok, MuJoCo, ODE, PhysX with introduction of quantitative measures and multiple ablative testing in different robotic task scenarios.

Collins et al. [131] have explicitly compared three physics engines, namely, MuJoCo [132], PyBullet [127] and the V-Rep [133] environment against a real-world setup for determining quantitatively the Sim2Real gap for manipulated oriented tasks. The work concludes that the simulation of the kinematic model and control of the manipulators is "largely solved" and **points to the problematic nature of simulated interaction between objects where the simulated dynamics fail to track.**

Korber et al. [134] also make a comparison study, this time between Gazebo, MuJoCo, PyBullet and Webots and with the scope of robotics and reinforcement learning, with the later translating mostly about performance characteristics under heavy parallelization.

Lastly [37] make an ablative study for comparing the MuJoCo, Bullet, Flex, and SOFA for deformable object simulation, where they state that MuJoCo was able to track better the real cloth dynamics.

2.5.2 AI Enhanced Physics Engines for Sim2Real

There have been numerous attempts to utilize some of the latest data-driven learning methodologies for augmenting the dynamics of the physics engines in order to compensate for residual unmodeled phenomena and thus bridging the Sim2Real gap.

Golemo et al. [135] with their Neural-Augmented Robot Simulation are able to learn the sim2real discrepancies when running the same actions in sim and real. They train and LSTM

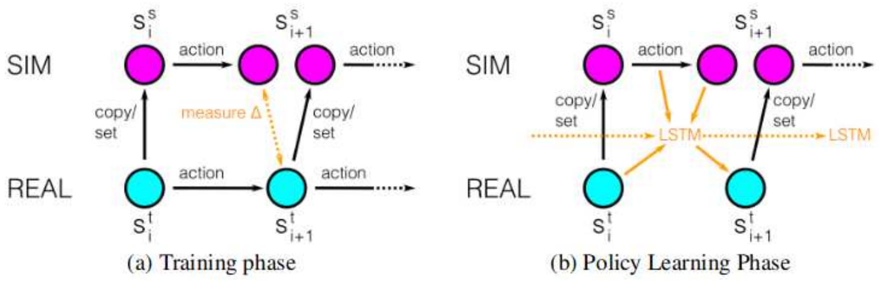


Figure 2.31 Neural-Augmented Robot Simulation [135]

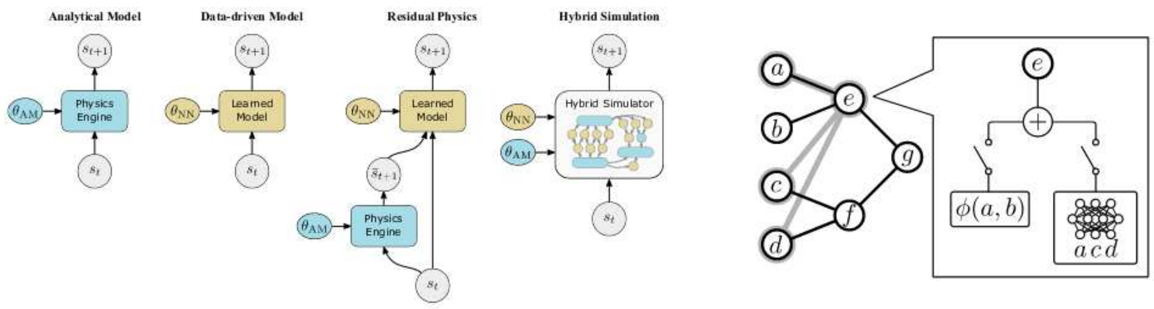


Figure 2.32 Potential AI-Augmented Physics Engines Schemas [137]

model based on these differences which they then deploy adjointly to the simulation in order to get more accurate and realistic state transitions (see fig. 2.31).

Ajay et al. [136] are focusing in enhancing the contact dynamics, as they are one of, if not the most, problematic source of sim2real discrepancies. They propose a hybrid dynamics model, the “simulator-augmented interaction networks (SAIN)”, where they combine a physics engine with an object-based neural network for dynamics modeling.

Heiden et al. [137, 138] introduce the “NeuralSim” augmenting a differentiable rigid-body physics engine with neural networks (see fig. 2.32).

Zeng et al. learn to model residual physics which are necessary in order to achieve the highly dynamic task of grasping objects and tossing them to a desired box location. The system predicts grasping and throwing motion primitives in an open-loop non reactive manner by self supervised training in simulation and the real-world.

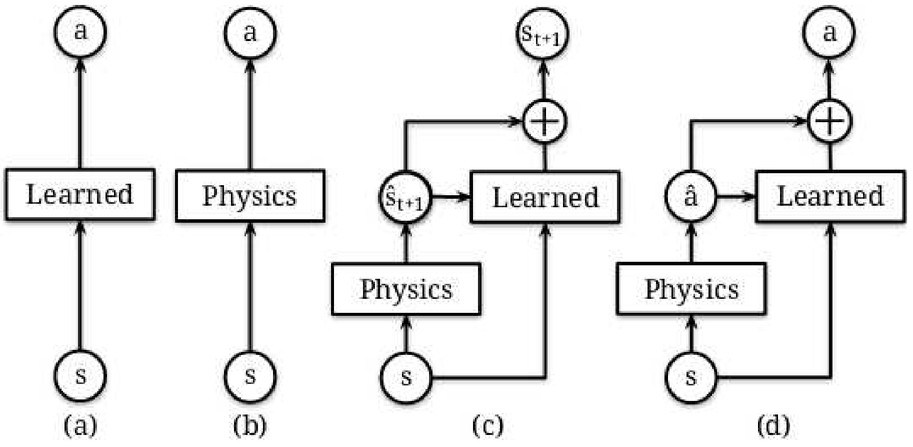


Figure 2.33 Hybrid Architectures for combining analytical and data-driven leaned residual physics [139]

Ota et al. [140] also learn residual physics, where they utilize MuJoCo for providing the analytic model which is then corrected using Gaussian process regression.

2.5.3 Physics Enhanced AI

A very interesting research direction, this time to the opposite direction in comparison the the previous subsection “AI Enhanced Physics”, is the “Physics Enhanced AI” for inducing strong physical laws as learning priors in order to increase the modeling accuracy, such as the “Lagrangian Neural Nets” [141, 142] or the “Hamiltonian Neural Nets” [143]

2.5.4 Differentiable Simulation

As the Deep Learning Architectures became predominant in Machine Learning there has been a rising interest in automatic differentiation, as the training of those relies on backpropagation in deep and non-linear networks.

It would be therefore appealing also for being able to automatically differentiate physics engines in order to be able to tune its parameters through some iterative gradient descent schema as a way to easily perform system identification, controller tuning, such as adap-

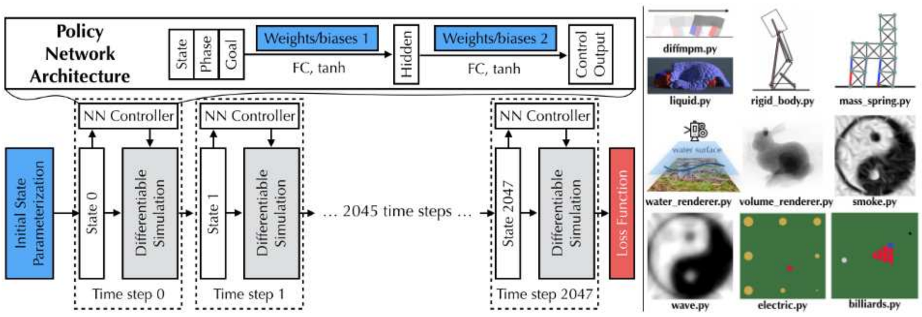


Figure 2.34 DiffTaichi allows for integration of a neural network controller and a physical simulation module [151]

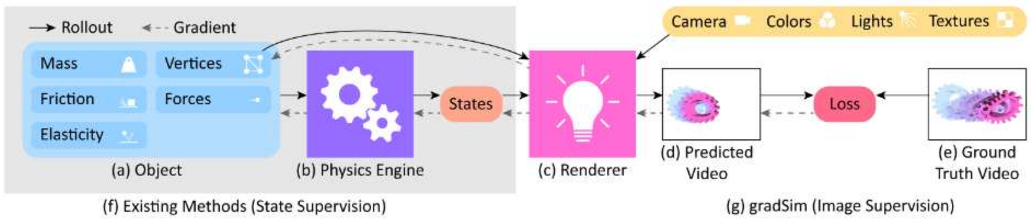


Figure 2.35 GradSim pipeline allows for end-to-end training of visuomotor policies[152]

tive MPC or Reinforcement Learning Agent training, even during robot development for determining its geometries and dynamics [144–150].

Hu et al. [151] present DiffTaichi which is a programming language specifically for assembling differentiable physics simulations (fig. 2.34). DiffTaich supports massive parallelization with GPU.

Jatavallabhula et al. [152] after auto-differentiating NVIDIA's flex, which is a physics engine for fast FEM simulation, for producing "dflex", they extend to a differentiable renderer as well for an end-to-end visuomotor policy training for controlling deformable structures (see fig. 2.35). Heiden et al. [148] also apply similar methodology, with the "DiSECt" project for learning policies for allowing cutting with a robotic enabled knife in simulation. Both of the aforementioned lines of work are limited in simulation trials. A similar pipeline and methodology in the real-world is developed for applying system identification by Arnavaz et al.[153] in an end-to-end fashion, namely from depth to simulation parameters for soft robotic fingers.

Xu et al. [154] demonstrate the effectiveness of differentiable simulation for accelerating policy learning with parallel differentiable simulation and they report 17 fold decrease in training time when compared to the best alternative baseline.

2.5.5 Material Deformations & Failure Modes

As the real-world is non-rigid, for the most part, rigid body physics engines, even in an ideal-state, which they are not, introduce a sim2real gap.

There has been a very strong research effort towards the simulation and control of deformable objects [37, 155–157].

Xian et al. [158] introduce “FluidLab” which is a differentiable environment for complex fluid robotic manipulation.

Material failure does not seem to receive much attention in simulation and is hardly ever discussed, with some notable exceptions [148].

2.5.6 A Taxonomy on Physics Engines

Deciding on investing in a Simulation Framework for the robotics application at hand can be a very overwhelming experience as a large number of options exist in the market and a large number of considerations needs to be taken into account, which requires highly specialized knowledge, which is hard to practically obtain.

In this section we attempt to list all the currently available physics engines and simulation frameworks for robotics applications, divided in a set of 9 main categories, namely:

- Representation-Agnostic Physics Simulation
- Physics Engines - Simulation Frameworks with Advanced Features for Learning
- Physics Engines - Rigid Body Dynamics
- Meta-Simulation Frameworks
- Simulation Frameworks for Soft Bodies and Deformable Bodies Simulation

- Frameworks allowing for complex multi-physics modeling
- Frameworks allowing for complex multi-physics modeling
- Libraries & Tools for Robotics Modeling & Simulation
- Physics Engines and Simulation Frameworks for the Web
- Physics Engines Misc.

This aims to be a helpful initial orientation guide for choosing the right software simulation framework in a robotics related application.

2.5.6.1 Representation-Agnostic Physics Simulation

Table 2.13 Representation-Agnostic Physics Simulation Frameworks.

Simulation Frame-work	**Description**
NVIDIA Kaolin	PyTorch API for working with a variety of 3D representations and includes a growing collection of GPU-optimized operations such as modular differentiable rendering, fast conversions between representations, data loading, 3D checkpoints, differentiable camera API, differentiable lighting with spherical harmonics and spherical gaussians, powerful octree acceleration structure called Structured Point Clouds, interactive 3D visualizer for jupyter notebooks, convenient batched mesh container, quaternion operations, representation-agnostic physics simulation It includes reduced elastic simulations of 3D objects in any geometric representation such as 3D Gaussian Splats, SDFs, point-clouds, and even medical scans. Our mesh-free, grid-free method utilizes implicit neural fields to construct a physics-aware subspace of the object via our data-free training process using the latest Simplicits method.

2.5.6.2 Physics Engines - Simulation Frameworks with Advanced Features for Learning

In this category we fetch a set of the most recent and advanced physics engines and simulation frameworks for robotics with incorporate features relevant for machine and reinforcement learning such as, CPU, TPU GPU parallelization and differentiability or any other learning related feature.

Table 2.14 Advanced Physics Engines towards Learning.

Simulation Frame-work	**Description**
MuJoCo	MuJoCo is a free and open source physics engine that aims to facilitate research and development in **robotics**. Designed for the purpose of **model-based optimization**, and **in particular optimization through contacts**. It features simulation in generalized coordinates, avoiding joint violations, Inverse dynamics that are well-defined even in the presence of contacts, constraints include soft contacts, limits, dry friction, equality constraints, simulation of particle systems, cloth, rope and soft objects, actuators including motors, cylinders, muscles, tendons, slider-cranks, choice of Newton, Conjugate Gradient, or Projected Gauss-Seidel solvers, Euler or Runge-Kutta numerical integrators, multi-threaded sampling and finite-difference approximations, MyJoCo Python Bindings. Around MuJoCo the "dm_control" software stack offers environments for Reinforcement Learning training. Lastly the MuJoCo MPC is a software framework for real-time predictive control. MuJoCo includes MuJoCo XLA (MJX) under the mjx directory. MJX allows MuJoCo to run on compute hardware supported by the XLA compiler via the JAX framework. MJX runs on a all platforms supported by JAX: Nvidia and AMD GPUs, Apple Silicon, and Google Cloud TPUs.

Simulation Frame-work	**Description**
NVIDA Isaac Sim	NVIDIA Isaac Sim™ is a reference application enabling developers to **design, simulate, test, and train AI-based robots and autonomous machines in a physically-based virtual environment.**Isaac Sim, built on NVIDIA Omniverse, is fully extensible, enabling developers to build their own Universal Scene Description (OpenUSD)-based custom simulators or integrate core Isaac Sim technologies into their existing testing and validation pipelines.**NVIDIA Isaac Lab is a lightweight sample application built on Isaac Sim and optimized for robot learning that's pivotal for robot foundation model training. Isaac Lab optimizes reinforcement, imitation, and transfer learning** and can train all types of robot embodiments, including the Project GR00T foundation model for humanoids.
brax	Brax is a fast and fully differentiable physics engine used for research and development of robotics, human perception, materials science, reinforcement learning, and other simulation-heavy applications.Brax is written in JAX and is designed for use on acceleration hardware. It is both efficient for single-device simulation, and scalable to massively parallel simulation on multiple devices, without the need for pesky datacenters.Brax simulates environments at millions of physics steps per second on TPU, and includes a suite of learning algorithms that train agents in seconds to minutes 1) Baseline learning algorithms such as PPO, SAC, ARS, and evolutionary strategies2) Learning algorithms that leverage the differentiability of the simulator, such as analytic policy gradients. Brax offers four distinct physics pipelines that are easy to swap: MuJoCo XLA - MJX, Generalized, Positional and Spring.

Simulation Frame-work	**Description**
jaxsim	JaxSim is a **differentiable physics engine** and **multibody dynamics library** designed for applications in control and robot learning, implemented with JAX. Its design facilitates research and accelerates prototyping in the intersection of robotics and artificial intelligence. Physics engine in r**educed coordinates supporting fixed-base and floating-base robots**. Multibody dynamics library providing all the necessary components for developing model-based control algorithms. Completely developed in Python with google/jax following a functional programming paradigm. Transparent support for running on CPUs, GPUs, and TPUs. Wide range of fixed-step explicit Runge-Kutta integrators. Support for variable-step integrators implemented as embedded Runge-Kutta schemes. Soft contacts model supporting full friction cone and sticking-slipping transition. Being developed with JAX, all the RBDAs support automatic differentiation both in forward and reverse modes. All fixed-step integrators are forward and reverse differentiable. Ideal for sampling synthetic data for reinforcement learning (RL). Ideal for designing physics-informed neural networks (PINNs) with loss functions requiring model-based quantities. Ideal for combining model-based control with learning-based components. JaxSim currently focuses on locomotion applications. Only contacts between bodies and smooth ground surfaces are supported. Ideal for combining model-based control with learning-based components.

Simulation Frame-work	Description
PhysX	NVIDIA's PhysX is a real-time physics engine widely used in gaming, simulation, and robotics applications for its ability to handle complex physical interactions and high-performance computations. PhysX provides **FEM soft body simulation**, **cloth**, **particles**, and **fluid simulation** with two way coupled interaction under a unified solver framework. PhysX simulations can run on a wide range of platforms, from low-power mobile CPUs to high-end GPUs. This includes a new GPU API designed for end-to-end **GPU-based reinforcement learning through Isaac Lab (prev. Isaac gym)**. Through collision detection and the solver, PhysX offers simulation stability for more robust stacking and joints. PhysX also includes momentum conservation for the articulation system and gyroscopic forces in the rigid body system. Reduced coordinate articulations provide a linear-time, guaranteed joint-error-free simulation of a tree of rigid bodies. PhysX's implementation closely matches analytical models. **Finite Element Method (FEM) soft bodies simulate measurable properties of hyperelastic materials to form an accurate and efficient model of elastic deformable bodies**. Signed Distance Field based collision representation allows PhysX to simulate non-convex shapes like gears and cams without convex decomposition. NVIDIA PhysX SDK includes **Blast, a destruction and fracture library**.
NVIDIA warp	Warp **auto- differentiable Python developer framework** for writing high-performance simulation and spatial computing graphics *GPU* code for GPUs. Warp takes regular Python functions and JIT compiles them to efficient kernel code that can run on the CPU or GPU.Warp is designed for spatial computing and comes with a rich set of primitives that make it easy to write programs for physics simulation, perception, robotics, and geometry processing. In addition, Warp kernels are differentiable and can be used as part of machine-learning pipelines with frameworks such as PyTorch and JAX. Warp generates a forward and backward (adjoint) version of each kernel definition. The backward version of a kernel can be used to compute gradients of loss functions that can be back propagated to machine learning frameworks like PyTorch.

Simulation Frame-work	**Description**
Taichi	Taichi Lang is an **open-source, imperative, parallel programming language for high-performance numerical computation**. It is embedded in Python and **uses just-in-time (JIT) compiler frameworks, for example LLVM, to offload the compute-intensive Python code to the native GPU or CPU instructions.** The **language has broad applications spanning real-time physical simulation**, numerical computation, augmented reality, **artificial intelligence, vision and robotics, visual effects in films and games, general-purpose computing, and much more**. Taichi Lang shares almost the same syntax with Python, allowing you to write algorithms with minimal language barrier. It is also well integrated into the Python ecosystem, including NumPy and PyTorch. It provides a set of generic data containers known as SNode (/snod/), an effective mechanism for composing hierarchical, multi-dimensional fields. This can cover many use patterns in numerical simulation (e.g. spatially sparse computing). With the ti.kernel decorator, Taichi Lang's JIT compiler automatically compiles your Python functions into efficient GPU or CPU machine code for parallel execution. Currently, Taichi Lang supports most mainstream GPU APIs, such as CUDA and Vulkan. A cross-platform, Vulkan-based 3D visualizer, differentiable programming, quantized computation (experimental), etc.
SAPIEN	SAPIEN is a realistic and physics-rich simulated environment that hosts a large-scale set for articulated objects. It enables various robotic vision and interaction tasks that require detailed part-level understanding. SAPIEN is a collaborative effort between researchers at UCSD, Stanford and SFU. SAPIEN Engine provides physical simulation for articulated objects. It **powers reinforcement learning and robotics with its pure Python interface**. SAPIEN provides **rasterized and ray traced rendering with Vulkan**. SAPIEN releases PartNet-Mobility dataset, which is a collection of 2K articulated objects with motion annotations and rendernig material. The dataset powers research for generalizable computer vision and manipulation. The entire stack is "as open-source as possible".

Simulation Frame-work	Description
EAGERx	EAGERx (*Engine Agnostic Graph Environments for Robotics*) to easily define new (Gymnasium compatible) environments with modular robot definitions. It enables users to: Define environments as graphs of nodes Visualize these graph environments interactively in a GUI Use a single graph environment both in reality and with various simulators EAGERx explicitly addresses the differences in learning between simulation and reality, with native support for essential features such as: Safety layers and various other state, action and time-scale abstractions Delay simulation & domain randomization Real-world reset routines Synchronized parallel computation within a single environment
Tiny-differentiable-simulator	Tiny Differentiable Simulator is a header-only C++ (and CUDA) physics library with zero dependencies.It currently implements various rigid-body dynamics algorithms, including forward and inverse dynamics, as well as contact models based on impulse-level LCP and force-based nonlinear spring-dampers. Actuator models for motors, servos, and Series-Elastic Actuator (SEA) dynamics are implemented.**The entire codebase is templatized so you can use forward- and reverse-mode automatic differentiation scalar types, such as CppAD, Stan Math fvar and ceres::Jet**. The library can also be used with regular float or double precision values. Another option is to use the included fix-point integer math, that provide cross-platform deterministic computation.TDS can run thousands of simulations in parallel on a single RTX 2080 CUDA GPU at 50 frames per second.

2.5.6.3 Physics Engines - Rigid Body Dynamics

This list contains what is probably now considered a “classic” physics engine for rigid-body and multi-body dynamics simulation. These physics engines are stemming from usage in the gaming industry and are mostly concerned with delivering a realistic feel to the eye, rather than accuracy, numerical stability and speed, requirements necessary for gaming applications. Nevertheless, these physics engines have become an invaluable tool in robotics, in both a more classical model-based design paradigm as well as a more modern data-driven and "learning" approach. Most of these implementations are inspired by Roy Featherstone's book "Rigid Body Dynamics Algorithms" [159].

Table 2.15 "Traditional" Rigid Body Dynamics Physics Engines

Physics Engines	**Description**
Drake	Drake is a C++ toolbox, with Python bindings, **created for model-based design and verification in robotics.** Drake's most important component is its physics engine, which provides state-of-the-art implementations of rigid and compliant body physics. Drake's physics engine focuses on robust numerics for **contact mechanics, using a method called hydroelastic contact to simulate contact forces**. This method calculates contact over finite-area patches between objects instead of using the conventional "contact point" method. Hydroelastic contact more realistically simulates the deformation of objects in contact by precalculating a "pressure field" throughout the interior volume of a compliant object. In addition to the physics engine, Drake offers a "systems framework" for building and combining systems from a library using a block diagram approach. Drake also provides an optimization framework that can be used to solve problems such as trajectory optimization and parameter estimation. Drake is open-source and supported by the Toyota Research Institute.
Bullet	The Bullet physics engine is an open-source physics engine used in games, visual effects, robotics, and machine learning.Bullet uses a Sequential Impulse constraint solver. It works by iteratively applying impulses (instantaneous changes in momentum) to objects to satisfy constraints, such as collisions or joints. Bullet includes: 1) Rigid body dynamics 2) Collision detection 3) Constraint solving: Bullet can handle various constraints, such as joints (like hinges or ball-and-socket joints) and contact constraints that prevent objects from penetrating each other.Bullet Physics offers support for simulating elastic deformations in objects using the Finite Element Method.Furthermore, a python API is offered, namely the "pybullet".

Physics Engines	Description
ODE	ODE is an open source, high performance library for simulating rigid body dynamics. It is fully featured, stable, mature and platform independent with an easy to use C/C++ API. It has advanced joint types and integrated collision detection with friction. ODE is useful for simulating vehicles, objects in virtual reality environments and virtual creatures. It is currently used in many computer games, 3D authoring tools and simulation tools.
DART	Nimble, developed at Stanford, is an analytically differentiable fork of the popular DART physics engine. That means everything from **fast Jacobians of dynamics, to using a physical timestep as a non-linearity in your PyTorch neural networks**.
RaiSim	RaiSim is a cross-platform multi-body physics engine for robotics and AI. It fully supports Linux, Mac Os, and Windows. RaiSim is closed-source and is distributed under a few different types of license. RaiSim is a physics engine developed by RaiSim Tech Inc. It is designed to provide both the accuracy and speed for simulating robotic systems. However, it is a generic rigid-body simulator and can simulate any rigid body very efficiently. RaiSim **seems to be used predominantly for legged robots and mobile platforms.**
OpenRAVE	OpenRAVE provides an environment for testing, developing, and deploying motion planning algorithms in real-world robotics applications. The main focus is on simulation and analysis of kinematic and geometric information related to motion planning. OpenRAVE's stand-alone nature allows is to be easily integrated into existing robotics systems. It provides many command line tools to work with robots and planners, and the run-time core is small enough to be used inside controllers and bigger frameworks. **An important target application is industrial robotics automation.**
Newton Dynamics	Newton Dynamics is a cross-platform life-like physics simulation C++ library. It can easily be integrated into game engines and other applications and provides top of it's class performance and simulation stability. Ongoing development and a permissive license makes Newton Dynamics a top choice for all kinds of projects from scientific projects to game engines. Newton Dynamics implements a deterministic solver, which is not based on traditional LCP or iterative methods, but possesses the stability and speed of both respectively. This feature makes Newton Dynamics a tool not only for games, but also for any real-time physics simulation.

Physics Engines	**Description**
havok	Havok is a commercial physics engine primarily used for game development with three leading product offerings: Havok Navigation, Havok Cloth and Havok Physics. It also offers integrations for all these products in the Unreal Engine as well as a Havok Physics implementation in Unity. Havok Physics offers the **Havok Visual Debugger** tool to empower you with rich capturing, profiling and debugging abilities. Havok **offers world–class support** for all our products. We have teams in Europe, North America and Japan, so there is always someone in your time zone to cater to your needs.
Simbody	Simbody is a high-performance, open-source toolkit for science- and engineering-quality simulation of articulated mechanisms, including biomechanical structures such as human and animal skeletons, mechanical systems like robots, vehicles, and machines, and anything else that can be described as a set of rigid bodies interconnected by joints, influenced by forces and motions, and restricted by constraints. Simbody includes a multibody dynamics library for modeling motion in generalized/internal coordinates in O(n) time. This is sometimes called a Featherstone-style physics engine. Simbody provides a C++ API that is used to build domain-specific applications; it is not a standalone application itself. For example, it is used by biomechanists in OpenSim, by roboticists in Gazebo, and for biomolecular research in MacroMoleculeBuilder (MMB). Python wrappers for the SimTK core libraries do exist under the name “PySimTK”.
React Physics 3D	ReactPhysics3D is a C++ physics engine library that can be used in 3D simulations and games. The library is developed by Daniel Chappuis and is released under the open-source ZLib license. It features, Rigid body dynamics, Discrete collision detection, Collision shapes (Sphere, Box, Capsule, Convex Mesh, Static Concave Mesh, Height Field), Multiple collision shapes per body, Broadphase collision detection (Dynamic AABB tree), Narrowphase collision detection (SAT/GJK), Collision response and friction (Sequential Impulses Solver), Joints (Ball and Socket, Hinge, Slider, Fixed), Collision filtering with categories, Ray casting and Sleeping technique for inactive bodies.

Physics Engines	**Description**
Rigid Body Dynamics	RigidBodyDynamics.jl is **a rigid body dynamics library in pure Julia**. It aims to be **user friendly** and **performant**, but also **generic** in the sense that the algorithms can be called with inputs of any (suitable) scalar types. This means that if fast numeric dynamics evaluations are required, a user can supply `Float64` or `Float32` inputs. However, if symbolic quantities are desired for analysis purposes, they can be obtained by calling the algorithms with e.g. `SymPy.Sym` inputs. If gradients are required, e.g. the `ForwardDiff.Dual` type, which implements forward-mode automatic differentiation, can be used.
siconos	Siconos is an open-source scientific software package for modelling and **simulation of nonsmooth dynamical systems** in C++ and Python. Our main applications are mechanical systems (rigid or solid) with unilateral contact and Coulomb friction and impact (nonsmooth mechanics, contact dynamics, multibody systems dynamics or granular materials).Other constitutive models are being developed to model plasticity, damage, cohesive zones.
Robotics Library	The Robotics Library (RL) is a self-contained C++ library for robot kinematics, motion planning and control. It covers mathematics, kinematics and dynamics, hardware abstraction, motion planning, collision detection, and visualization.It is being used by several research projects (e.g., JAHIR, JAMES, JAST, SMErobotics) and in education, available under a BSD license, and free for use in commercial applications.RL can be run on all machines from real-time patched Linux to Windows desktop PCs. It uses CMake as a build system, may be compiled with GCC and Visual Studio.

Physics Engines	**Description**
npphysics	npphysics is a physics engine **designed for the Rust programming** language, supporting 2D and 3D simulations. It supports rigid body dynamics, deformable bodies and continuous collision detection. It also allows the simulation of deformable bodies, employing either a mass-spring system or the finite-element method. To cater to different accuracy and performance needs, the engine offers a selection of contact models, such as the Signorini and approximate Signorini-Coulomb models. It also facilitates interaction handling by providing access to collision events, empowering developers to implement custom logic based on object interactions. Additionally, it features sensors, specialized colliders designed to detect proximity. To ensure robust collision detection, even for rapidly moving objects, nphysics incorporates continuous collision detection (CCD) support for both standard colliders and sensors.
Trep	Trep is a Python module for modeling articulated rigid body mechanical systems in generalized coordinates. Trep supports basic simulation but it is primarily designed to serve as a calculation engine for analysis and optimal control algorithms that require 1st and 2nd derivatives of the system's dynamics. ROS users can now get trep binaries directly from the ROS repositories. Check the *python_trep* wiki page for more information.
Matali Physics	Matali Physics is an advanced, modern, multi-platform, high-performance 3d physics environment **intended for games**, VR, AR, physics-based simulations and robotics. Matali Physics consists of the 3d physics engine Matali Physics Core and other physics-driven modules. It supports **destructible scenes** and **deformable triangle meshes**. Commercially Licensed and Written in native, modern C++.
IBDS	The IBDS physics library is a C++ library for the dynamic simulation of multi-body systems. The library is open source, under the zlib license, and is therefore free for commercial use. IBDS simulates rigid bodies, particles, collisions (with static and dynamic friction), and many different joint types. IBDS stands for Impulse-Based Dynamic Simulation. IBDS also stands for Institut für Betriebs- und Dialogsysteme, the University of Karlsruhe institute where IBDS was developed.

Physics Engines	Description
RBDL	The Rigid Body Dynamics Library (RBDL) contains code for both forward and inverse dynamics for kinematic chains and branched models. It includes 1) Recursive Newton Euler Algorithm (RNEA) 2) Composite Rigid Body Algorithm (CRBA) 3) Articulated Body Algorithm (ABA). RBDL is written in C++ and it also contains python bindings.
Klampt	Klamp't is a cross-platform software package for modeling, simulating, planning, and optimization for complex robots, particularly for manipulation and locomotion tasks. It has been developed at Indiana University since 2009 primarily as a research platform, and has been used in classrooms beginning in 2013. It has been used in several real-world projects, including the Amazon Picking Challenge, TeamHubo in the DARPA Robotics Challenge, and was the platform for the IROS 2016 Robot Grasping and Manipulation Challenge simulation track. Supports legged and fixed-based robots. Built in models NASA ATHLETE, Rethink Robotics Baxter, AIST HRP-2, Willow Garage PR2, KAIST Hubo-II+, Robonaut 2, Staubli TX90, Puma 760, Robotiq Adaptive Gripper. Forward and inverse kinematics, forward and inverse dynamics Contact mechanics computations: force closure, support polygons, stability of rigid bodies and actuated robots.
RBDyn	RBDyn provides a set of classes and functions to model the dynamics of rigid body systems in C++ and with python bindings as well.This implementation is based on Roy Featherstone Rigid Body Dynamics Algorithms book and other state of the art publications.
Jolt Physics	A multi core friendly rigid body physics and collision detection library. Suitable for games and VR applications. Used by Horizon Forbidden West. It allows for soft and deformable object simulation e.g. a soft ball or piece of cloth, as well as buoyancy calculations.

Physics Engines	Description
qu3e	qu3e is a compact, light-weight and fast 3D physics engine in C++. It is has been specifically created to be used in games. It is portable with no external dependencies other than various standard c header files (such as **cassert** and **cmath**). qu3e is designed to have an extremely simple interface for creating and manipulating rigid bodies.qu3e is of particular interest to those in need of a fast and simple 3D physics engine, without spending too much time learning about how the whole engine works. In order to keep things very simple and friendly for new users, **only box collision is supported. No other shapes are supported** (capsules and spheres may be added in the future if requested).

2.5.6.4 Meta-Simulation Frameworks

Under this category we place "Meta-Simulation Frameworks", namely high-level software suites for robotics with advanced GUI, user-friendly, capabilities which incorporate more than one physics engines.

Table 2.16 “Meta” Simulation Frameworks

Meta-Framework	**Description**
Gazebo	Gazebo is an open-source robotics simulator that provides a versatile and robust platform for designing, testing, and validating robotic systems in complex environments. It offers high-fidelity simulation of physical interactions, including rigid body dynamics, collision detection, and sensor feedback, supporting a wide range of sensor and actuator models. Gazebo’s integration with the Robot Operating System (ROS) enhances its utility for developing and testing robotic algorithms in a realistic virtual environment. However, Gazebo can be computationally intensive, potentially leading to performance bottlenecks when simulating large-scale environments or highly detailed models. Its steep learning curve and the complexity of setting up simulations can pose challenges, especially for users with limited experience in robotic simulation. Additionally, while Gazebo’s extensive plugin architecture allows for significant customization, ensuring compatibility and stability across different plugins and ROS versions can be problematic, requiring meticulous configuration and testing to achieve reliable performance. Gazebo supports the following physics engines: ODE, Bullet, DART, and Simbody.
CoppeliaSim	CoppeliaSim is used for fast algorithm development, **factory automation** simulations, fast prototyping and verification, robotics related education, remote monitoring, safety double-checking, as digital twin, and much more. The robotics simulator CoppeliaSim is based on a distributed control architecture. Each object/model can be individually controlled via: Python, Lua, C/C++, an embedded script (Python or Lua), remote API client like Python, Lua, Java, MATLAB, Octave, C, C++, Rust, or a custom solution. It supports 5 physics engines (MuJoCo, Bullet Physics, ODE, Newton and Vortex Dynamics) Forward/Inverse kinematics calculations for any type of mechanism (branched, closed, redundant, containing nested loops, etc.). An embeddable version of the IK/FK algorithms is available. Powerful, realistic and exact volumetric proximity sensor simulation: performs an exact minimum distance calculation within a customizable detection volume. Operates on meshes, octrees and point clouds. Simulation of vision sensors with many image processing options, fully customizable and extendable (e.g. via plugin). Path planning / motion planning is supported in a very flexible way via the OMPL library wrapped in a plugin for CoppeliaSim.**PyRep** is a toolkit for robot learning research, built on top of CoppeliaSim.

Meta-Framework	**Description**
AMBF	Asynchronous Multi-Body Framework (AMBF) offers a real-time dynamic simulation of robots, free bodies, and multi-link puzzles coupled with **real-time haptic interaction via several haptic devices** (CHAI-3D) (including dVRK Manipulators and Razer Hydras). It also provides a **Python client for training NN and RL Agents on real-time data with the simulation in the loop**. This framework is built around several external tools that include an extended version of **CHAI-3D** (developed alongside AMBF), **BULLET-Physics**, Open-GL, GLFW, yaml-cpp, pyyaml, and Eigen to name a few.
Vortex Stu-dio	Vortex Studio is a commercial solution simulation and visualization platform designed for modeling and testing complex mechanical systems, **particularly in robotics and heavy machinery applications**. It provides robust tools for simulating real-time physics, including rigid body dynamics, collision detection, and multi-body interactions.
Webots	Webots is an open-source robotics simulation platform that provides an environment for developing, testing, and validating robotic systems in realistic 3D environments. It supports a wide range of robot models and offers detailed physics simulations, including rigid body dynamics, sensors, and actuator dynamics, which are crucial for accurate performance evaluation. Webots features a graphical interface that facilitates rapid prototyping and integration with various programming languages, including C++, Python, and MATLAB. Webots utilized the ODE physics engine but it also offers the possibility to add custom physics plugins making possible to e.g. to design an aerodynamics model for a flying robot, a hydrodynamics model for a swimming robot, etc.
MARS	The MARS provides physics simulations, enabling modeling of robot dynamics, sensors, and interactions with the virtual environment in C++. MARS supports various robotic platforms and is equipped with features for simulating rigid body dynamics, soft body physics, and multi-body systems. It runs on (Ubuntu) Linux, Mac and Windows and consists of a core framework containing all main simulation components, a GUI (based on Qt), 3D visualization (using OSG) and a physics core (based on ODE). It focuses on developing and validating algorithms for navigation, control, and perception. It is developed by developed at the Robotics Innovation Center of the German Research Center for Artificial Intelligence (DFKI-RIC) and the University of Bremen.

Meta-Framework	Description
SimSpark	simulation system for various multiagent simulations. It supports developing physical simulations for AI and robotics research with an open-source application framework. SimSpark uses the Open Dynamics Engine (ODE) for detecting collisions and for simulating rigid body dynamics.

2.5.6.5 Simulation Frameworks for Soft Bodies and Deformable Bodies Simulation

In this list we include physics engines and simulation frameworks which are highly specialized in the simulation of soft bodies and deformable objects.

Table 2.17 Simulation Frameworks for Soft/Deformable Bodies.

Physics Engines	Description
NVIDIA FleX	FleX is a particle based simulation technique for real-time visual effects.Traditionally, visual effects are made using a combination of elements created using specialized solvers for rigid bodies, fluids, clothing, etc. Because FleX uses a unified particle representation for all object types, it enables new effects where different simulated substances can interact with each other seamlessly FleX capabilities on particle simulation should not be confused with PhysX's similar capabilities. From github discussion NVIDIA stuff: "The PhysX particle system is indeed not a straight port of Flex, and it will likely never become a 1:1 replacement. Flex went a long way being a general simulator using only particles which comes with great uniformity at the interface level and provides possibilities which other systems don't provide that easily like phase transitions. It also has disadvantages for very common use-cases. PhysX is trying to strike another balance." The derivative version of FleX, the "dflex", as result of automatic differentiation, has been incorporated in recent research [152, 154] allowing for policy learning with differentiable simulation.

Physics Engines	**Description**
Obi	*Obi* is able to simulate the interaction between ropes, cloths, liquid and general volumetric bodies. Obi has experimentally proven it's ability to robustly simulate ropes for bi-manual robotic manipulation in the case of the highly demanding shoe lacing task [160]. In this case Obi was used in combination with *Unity*.
SOFA	Simulation Open Framework Architecture (SOFA) for interactive mechanical simulation, with emphasis on biomechanics and robotics is an open-source library distributed under LGPL license, hosted on GitHub. It offers state-of-the-art constitutive laws and algorithms to efficiently **compute soft and rigid body dynamics**. **Linear and non-linear elastic models as well explicit/implicit integration schemes and traditional linear solvers.** All SOFA simulations can be succinctly described by a python script.
Position Based Dynamics	This library supports the physically-based simulation of mechanical effects. In the last years position-based simulation methods have become popular in the graphics community. In contrast to classical simulation approaches these methods **compute the position changes in each simulation step directly, based on the solution of a quasi-static problem**. Therefore, position-based approaches are fast, stable and controllable which make them well-suited for use in interactive environments. However, these methods are generally **not as accurate as force-based methods but still provide visual plausibility.** Hence, the main application areas of position-based simulation are virtual reality, computer games and special effects in movies and commercials.The PositionBasedDynamics library allows the position-based handling of many types of constraints in a physically-based simulation. **Library supports many constraints, lastic rods, eformable solids.**
Sorotoki	Sorotoki is an **open-source MATLAB toolkit for soft robotics that aims to facilitate the development of novel research in the field by providing a comprehensive set of tools for design, modeling, and contro**l. The toolkit includes a diverse array of scientific disciplines relevant to soft robotics, such as continuum mechanics, dynamic systems and control theory, topology optimization, and computer graphics. The Sorotoki toolkit aims to make it easier for new researchers to learn about soft robotics by providing a range of tools that cover various important aspects of the field. This can significantly reduce the amount of time and effort required to get up to speed on the topic.

Physics Engines	Description
SomoGym	SoMoGym (SoftMotion Gym) [157] is an open-source framework that builds on SoMo (SoftMotion) in order to facilitate the simulation of continuum manipulator (CM) motion in pybullet. SoMo makes it easy to create URDFs of such approximated manipulators and load them into pybullet's rigid body simulator. With SoMo, environments with various continuum manipulators, such as hands with soft fingers (xxx links), or snakes, can be created and controlled with only a few lines of code.
SoftGym	SoftGym [155], a set of open-source simulated benchmarks for manipulating deformable objects, with a standard OpenAI Gym API and a Python interface for creating new environments. SoftGym builds on top of Nvidia FleX physics simulator.
ARCSim	ARCSim is a simulation engine used for animating sheets of deformable materials. The simulator is well-suited for materials such as cloth, paper, plastic, and metal. ARCSim uses adaptively refined triangle meshes to efficiently resolve the geometric and dynamic detail of the simulated objects. It accomplishes this through adaptive anisotropic remeshing, which aligns mesh edges with features like wrinkles and creases, efficiently resolving fine details. This technique anticipates buckling and wrinkle formation, preserving fine-scale dynamic behaviour. Additionally, ARCSim includes an efficient implementation of strain limiting, working for arbitrary non-uniform and anisotropic meshes and converging faster than other solvers.

2.5.6.6 Libraries & Tools for Robotics Modeling & Simulation

This list includes a set of libraries which include tools for robotic modeling and simulation, which do not directly fit in the "physics engine" and "simulation framework" category.

Table 2.18 Libraries & Tools for Robotics Modeling and Simulation

Libraries and Tools	**Description**
Pinocchio	Pinocchio is a robust robotics library focused on the efficient modeling and simulation of rigid body dynamics. It implements kinematic and dynamic algorithms, leveraging articulated body dynamics to optimize performance for real-time applications. It instantiates the state-of-the-art Rigid Body Algorithms for poly-articulated systems based on revisited Roy Featherstone's algorithms. Pinocchio provides the analytical derivatives of the main Rigid-Body Algorithms like the Recursive Newton-Euler Algorithm or the Articulated-Body Algorithm. Pinocchio's architecture enables easy integration with other software frameworks, such as ROS and OpenHRP. Its capabilities make it a valuable tool for researchers and engineers tackling complex robotics problems, although careful consideration is required for integration in large-scale systems due to potential overhead in more intricate scenarios. It is built upon Eigen for linear algebra and FCL for collision detection. It comes with a Python interface for fast code prototyping.
iDynTree	iDynTree is a C++ library that provides algorithms for robot dynamics calculations, useful for control, estimation, and simulation. Although specifically designed for robots with a free-floating base, it can also be used with fixed-base robots. iDynTree is written in C++ and offers bindings for Python and MATLAB. The library employs an undirected graph data structure iDynTree::Model to represent robots. This allows for flexible changes to the base link used in kinematics and dynamics computations without requiring model reloads or changes to joint/link serializations. It supports reading and writing URDF files from an iDynTree::Model, which is beneficial for tools modifying robot models and saving them. This feature was designed for developing tools that identify kinematics and dynamics parameters. While it defaults to the mixed representation for link quantities, iDynTree allows users to opt for body or inertial representations. This caters to research in floating-base whole-body controller synthesis. Joint Torque Estimation: iDynTree includes an algorithm used by the iCub humanoid robot, which estimates joint torques without requiring dedicated joint torque sensors. It leverages the library's unique undirected graph data structure.

Libraries and Tools	**Description**
KDL	KDL, the Kinematics and Dynamics Library, is an application-independent framework for modelling and computing kinematic chains like robots, human models, animated figures and machine tools. KDL provides C++ class libraries for 1) Geometric Primitives: These include points, frames, and twists. 2) Kinematic Trees: KDL uses kinematic trees to represent the structure of kinematic chains. 3) Kinematic and Dynamic Solvers: KDL provides various generic forward and inverse kinematic algorithms and redundancy resolution.4) Motion Trajectories: This includes Cartesian paths, velocity profiles and Cartesian trajectories.KDL also includes parameters for dynamics, such as inertia. All real-time-safe operations and functions in KDL are deterministic in time and do not use dynamic memory allocation. KDL also provides Python bindings, typekits and transport-kits for Orocos/RTT and is integrated into ROS.
Kindr	The Kindr library by ANYbotics is a C++ tool designed for efficient and rigorous mathematical operations in robotics, leveraging the Eigen library for high-performance computations. It supports a wide range of types and operations for 3D vectors, rotations, and transformations, with a consistent and intuitive API. Kindr handles different coordinate frames and transformations, crucial for managing sensors, actuators, and reference frames in robotics applications. Extensible and customizable, Kindr integrates well with other software components, is compatible with ROS, and supports cross-platform use. Developed and maintained by ANYbotics, it is robust, reliable, and ideal for research, development, simulation, and control in autonomous robotic systems.
MRPT	Mobile Robot Programming Toolkit (MRPT) provides developers with portable and well-tested applications and libraries covering data structures and algorithms employed in common robotics research areas.It is open source, released under the 3-clause BSD license.

2.5.6.7 Physics Engines and Simulation Frameworks for the Web

In this list we include a set of physics engines and simulation frameworks specialized for web applications written in JavaScript.

Table 2.19 Physics Engines for the Web.

Physics Engine	**Description**
OimoPhysics	The OimoPhysics engine is a lightweight, high-performance 3D physics simulation library designed for applications in games and interactive simulations. It is implemented in JavaScript and optimized for use with WebGL, making it particularly suitable for web-based applications. The engine supports a variety of physical behaviors, including rigid body dynamics, collision detection, and constraint resolution. Key features include support for convex hull shapes, spheres, boxes, and cylinders, as well as a broad phase collision detection algorithm to enhance computational efficiency. OimoPhysics also incorporates iterative constraint solvers to ensure stable and realistic simulations, even in complex scenarios. Its API is designed for ease of integration and flexibility, allowing developers to customize and extend its functionality to meet specific requirements.
Rapier	Rapier is a set of 2D and 3D physics engines written using the Rust programming language. It targets applications requiring real-time physics like video games, animation, and robotics. It is designed to be fast, stable, and optionally cross-platform deterministic. Rapier features include among other, 1) rigid-body collisions and forces. 2) Joint constraints. 3) Contact events and sensors. 4) JavaScript bindings.Free and Open-Source built with a FOSS mindset.
PHY	PHY Universal physics language on Worker or Direct for three.js Phy simplify game creation, is a bridge between three.js and physics. You can use compress or full version of physics engines. **It supports Oimo, Ammo, Rapier, Jolt, Havok and Physx.**
ammo	ammo.js is a direct port of the Bullet physics engine to JavaScript, using Emscripten. The source code is translated directly to JavaScript, without human rewriting, so functionality should be identical to the original Bullet.‘ammo’ stands for “Avoided Making My Own js physics engine by compiling bullet from C++”.ammo.js is zlib licensed, just like Bullet.

2.5.6.8 Frameworks allowing for complex multi-physics modeling

For completeness sake's we also include a set of multi-physics and FEM analysis simulation suites.

Table 2.20 Multi-Physics Simulation Frameworks.

Physics Engine	**Description**
Project Chrono	Project Chrono is an open-source physics engine written in C++ and allows for multibody dynamics, nonlinear finite element analysis, large-scale simulation, collision detection. A Python wrapper, Pychrono, is available via Anaconda. Project Chrono offers a DEM-Engine, described as an "External GPU solver for DEM simulations". This suggests that Chrono leverages GPU computing for Discrete Element Method simulations, likely employing solvers optimised for parallel processing on GPUs.
Simscape	MathWorks' Simscape is a simulation tool integrated with MATLAB and Simulink, enabling multidomain physical modeling essential for robotics applications. It facilitates the simulation of mechanical, electrical, hydraulic, and thermal components of robotic systems, offering a broad library of pre-built and customizable components for **modeling robotic joints, actuators, sensors, and controllers.**

Physics Engine	**Description**
ANSYS	ANSYS is a simulation software suite renowned for its capabilities in finite element analysis (FEA), computational fluid dynamics (CFD), and electromagnetics, among other domains. It provides robust tools for simulating a **wide range of physical phenomena, including structural mechanics, fluid flow, thermal management, and electromagnetic fields**. ANSYS excels in high-fidelity modeling and offers comprehensive material libraries, advanced meshing algorithms, and powerful solver technologies, enabling precise and reliable simulations of complex engineering systems. The software supports multiphysics simulations, allowing for the coupling of different physical domains to accurately capture the interactions in real-world scenarios. Despite its strengths, ANSYS is known for its steep learning curve and high computational requirements, which can pose challenges for new users and require substantial computational resources. Additionally, the complexity of setting up and validating intricate simulations can lead to lengthy preparation times and necessitate significant expertise. While ANSYS offers extensive documentation and support, effectively leveraging its full potential often demands a deep understanding of both the software and the underlying physical principles.
Abaqus	Abaqus, developed by Dassault Systèmes, is an advanced simulation software suite used extensively for **finite element analysis (FEA)** in various engineering disciplines. It is particularly renowned for its robust capabilities in **simulating complex material behaviors and nonlinear interactions, making it ideal for structural analysis, stress testing, and failure analysis**. Abaqus supports a wide range of material models, including metals, polymers, composites, and hyperelastic materials, allowing for accurate representation of real-world conditions. Its comprehensive suite includes Abaqus/Standard for static and low-speed dynamic events and Abaqus/Explicit for high-speed dynamic events and complex contact problems. However, the sophisticated features and extensive customization options of Abaqus come with a steep learning curve and significant computational demands. Setting up and validating detailed simulations can be time-consuming and require a deep understanding of both the software and the specific application domain. Despite these challenges, Abaqus is widely regarded for its precision and versatility, offering extensive documentation and a strong support network to assist users in leveraging its full potential.

Physics En-gine	**Description**
COMSOL	COMSOL Multiphysics is a simulation software that provides a comprehensive environment for modeling and solving physics-based problems across various engineering and scientific disciplines. It supports a wide range of physics interfaces, **including structural mechanics, fluid dynamics, electromagnetics, heat transfer, and chemical reactions**. The platform's multiphysics capabilities allow for the **seamless coupling of different physical phenomena** within a single model, enabling the simulation of complex interactions and real-world conditions. COMSOL's powerful solver algorithms and customizable physics interfaces offer high accuracy and flexibility, but the software's steep learning curve and significant computational demands can be challenging. Effective utilization of COMSOL often requires substantial expertise in both the software and the underlying physical principles. Additionally, while COMSOL provides extensive documentation and support, the complexity of setting up multiphysics simulations can lead to long development times, necessitating careful planning and validation to ensure accurate and reliable results.
MBDyn	MBDyn is an open-source multibody dynamics analysis software distributed under the GNU GPL 2.1 license, developed by the Department of Aerospace Science and Technology at Politecnico di Milano, Italy. It supports the simulation of multibody and multiphysics systems, including rigid and flexible body dynamics, smart materials, and various physical networks. The software can integrate with external solvers for co-simulation of multiphysics problems, such as CFD and terradynamics, via a straightforward API. MBDyn is used extensively in aerospace, wind energy, automotive, and mechatronics for dynamic system analysis and simulation. It allows for runtime loading of user-defined modules to extend its capabilities and supports real-time execution on GNU/Linux with RTAI. It is a command-line tool designed for researchers, not a commercial software with a user-friendly GUI or round-the-clock support.

2.5.6.9 Physics Engines Misc.

In this list we include a set of miscellaneous, highly specialized simulation frameworks in a specific area.

Table 2.21 Residual-Miscellaneous Simulation Frameworks

Physics Engine	**Description**
Chai3D	launched in 2003 at the Robotics and Artificial Intelligence Laboratory at Stanford University, CHAI3D is a powerful cross-platform C++ simulation framework with over 100+ industries and research institutions developing CHAI3D based applications all around the world in segments such as automotive, aerospace, medical, entertainment and industrial robotics.Designed as a **platform agnostic framework for computer haptics, visualization and interactive real-time simulation**, CHAI3D is an open source framework that supports a variety of commercially-available three-, six- and seven-degree-of-freedom haptic devices, and makes it simple to support new custom force feedback devices.CHAI3D's modular capabilities allows for the creation of highly-performing **native haptic applications** as well as for hybrid development where you can choose which components provide the best haptic and visual user experience.
habitat-sim	physics-enabled 3D simulator developed by Facebook Research with support for 3D scans of i**ndoor/outdoor spaces**, CAD models of spaces and piecewise-rigid objects, Configurable sensors (RGB-D cameras, egomotion sensing), Robots described via URDF, Rigid-body mechanics (via Bullet Physics Engine). The **design philosophy of Habitat is to prioritize simulation speed over the breadth of simulation capabilities**. When rendering a scene from the Matterport3D dataset, Habitat-Sim achieves several thousand frames per second (FPS) running single-threaded and reaches over 10,000 FPS multi-process on a single GPU.
MINOS	MINOS is a simulator designed to support the development of multisensory models for goal-directed navigation in complex indoor environments. MINOS leverages large datasets of complex 3D environments and supports flexible configuration of multimodal sensor suites.

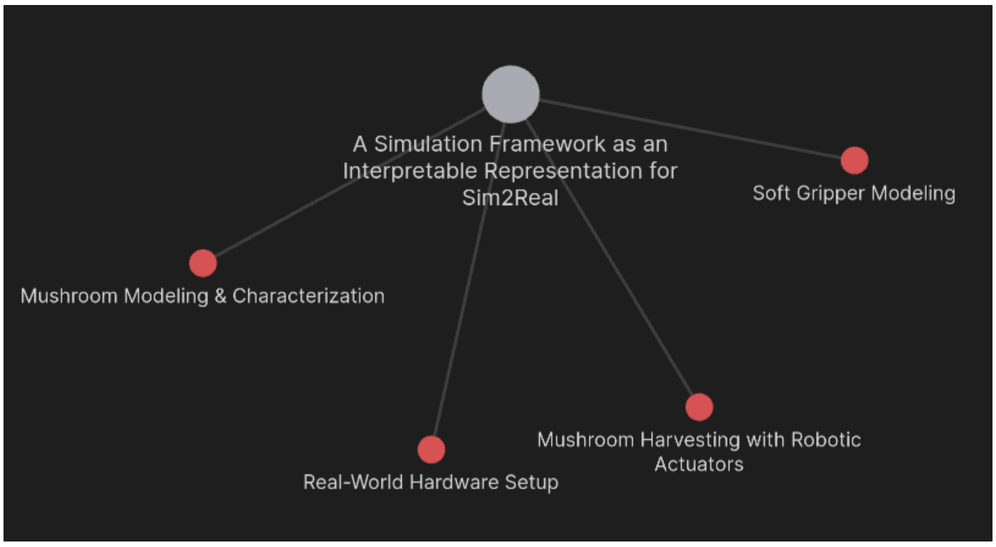
Physics Engine	**Description**
Deepmind Lab	*DeepMind Lab* is a 3D learning environment based on id Software's Quake III Arena via ioquake3 and other open source software. *DeepMind Lab* provides a suite of challenging 3D navigation and puzzle-solving tasks for learning agents. Its primary purpose is to act as a testbed for research in artificial intelligence, especially deep reinforcement learning.
Gibson	Gibson is a virtual environment based off of real-world, as opposed to games or artificial environments, to support learning perception. Gibson enables developing algorithms that explore both perception and action hand in hand.
iTHOR	iTHOR is an environment within the AI2-THOR framework, which includes a set of interactive objects and scenes and provides accurate modeling of the physics of the world.
AGX Dynamics	Modeling and simulation of complex mechanical systems, handle impacts, contacts, and friction at large, fixed step. can model robots, heavy vehicles and machinery, cranes, and other complex mechanical systems found in manufacturing and transportation for instance. C++ SDK comes with C# and Python bindings OpenSceneGraph viewer allows for quick modeling and data analysis. The AGX Dynamics for Unity and AGX Dynamics for Unreal modules help develop interactive applications with stunning graphics. Also available are FMI and Simulink export functions, as well as SpaceClaim integration. Commercially Licensed.
matlab-whole-body-simulator	matlab-whole-body-simulator is a **simulator for the humanoid robots. It has been designed to work in Simulink**. In the simulator the ground is assumed to be flat and the contact forces are computed using the Maximum dissipation principle and under the linear approximation of the friction cones assumption.

Chapter 3

A Simulation Framework as an Interpretable Representation for Sim2Real

"For it is the same thing that can be thought and that can be."

—Parmenides, *On Nature*


A Simulation Framework as an Interpretable Representation for Sim2Real
Soft Gripper Modeling
Mushroom Modeling & Characterization
Mushroom Harvesting with Robotic Actuators
Real-World Hardware Setup

Our strategy for tackling this complex and multifaceted problem revolves around the development of a simulation framework with the PyBullet [127] physics engine to capture the dominant dynamics involved by first-order approximations of the mushroom and gripper overall system. By doing so, we can take advantage of a rich set of tools & capabilities that the physics engine provides, such as Newton-Euler governing motion dynamics, collision detection between scene objects, and friction models coupled with RGB-D rendering with scene segmentation. As we are dealing with complex, nonlinear dynamics with a high degree of uncertainty, like object deformations, failure modes, and potentially soft-gripper actuation with the additional requirement for fast computation for allowing data-driven and MPC-like techniques, we are utilizing first-order approximations of the equivalent continuum mechanics models. For the derivation of the modeling parameters, we conducted real-world experiments for the characterization of mushroom stiffness with force sensors and the Phasespace tracking system [161].

3.1 Mushroom Harvesting with Robotic Actuators

Previous attempts have been made on white button mushroom harvesting with robotic actuators. [162] have presented a sense-plan-act approach with a camera-based mushroom localization and a twisting action primitive for detachment with a suction cap gripper, resulting in a reported success rate of 57%. [163] report an average success rate of 70% by using a combination of bending and twisting. At a later stage, [164] will present a more fine-tuned version for the 2D planar mushroom localization, based again on a monochromatic camera and, most importantly, on a gripper capable of bending and twisting motions composites, resulting this time in an improved average success rate of 68% for the overall system. [165] also based on a suction cap gripper with a root bending action only as an optimal strategy for bruising avoidance [166] reports 90% and 94.2% success rates for first and second pick, respectively.

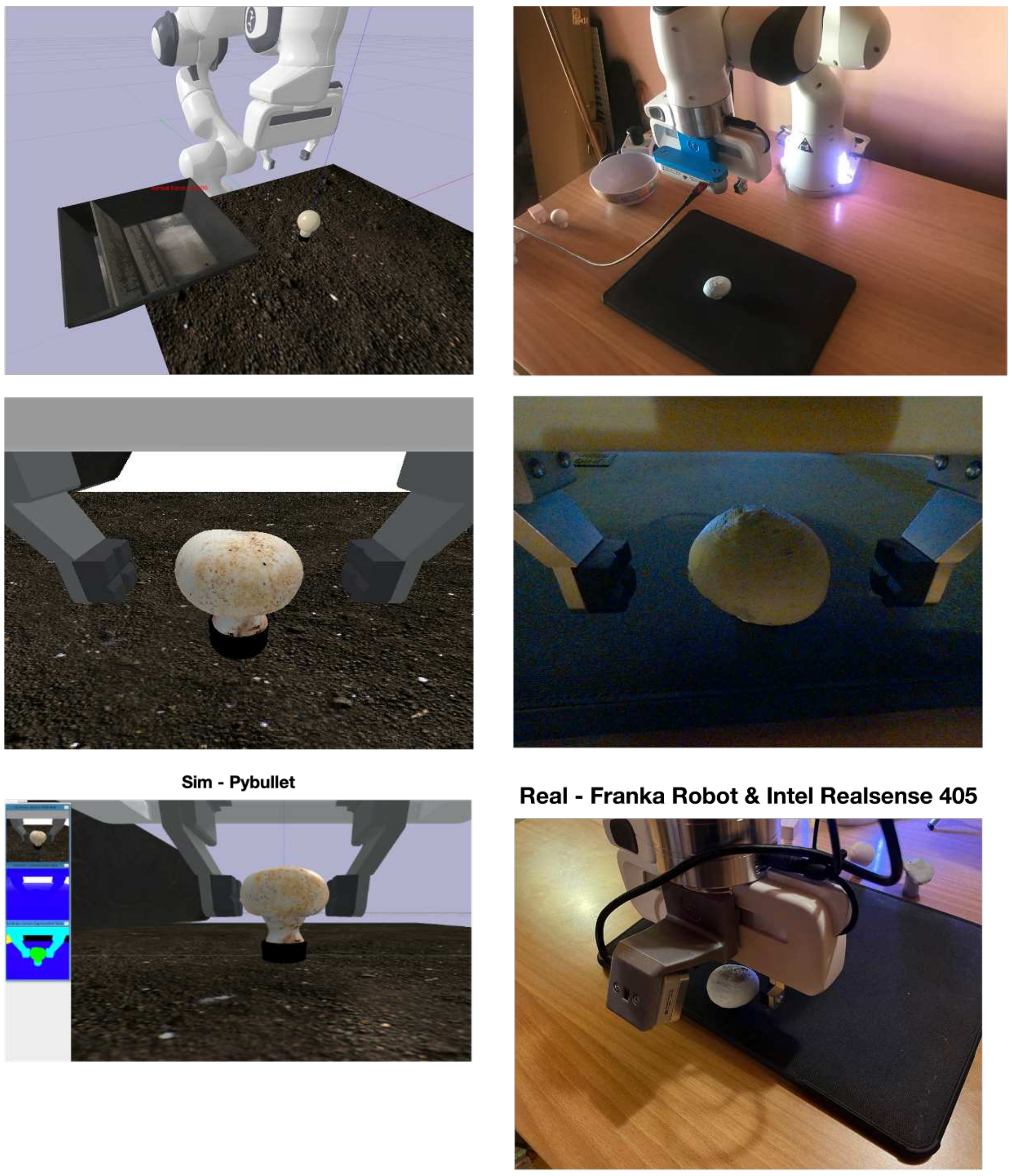


Figure 3.1 The Simulated Environment for Mushroom Harvesting and the Real-World Equivalent Setup.

The definition of the mechanical properties of the mushroom root system has been the subject of previous studies. In the works of [167] and [168] the anisotropic elastic properties of the mushroom stiffness are studied and experimental values for the Young Elastic Modulus and strain are derived. Similar methodologies for crop characterization can be found in other types of produce [169].

Similar challenges arise in other types of produce in agriculture. [170] where an adaptive impedance controller has been developed for apple harvesting, following an analytical modeling and characterization process. [171] study optimal picking patterns for apple harvest based on an analytical Finite Element Model (FEM). [172] develop a strategy based on human demonstrations on a physical twin for the harvest of raspberries. [173] present a visual servo controller for autonomous citrus harvesting.

The task of mushroom harvesting can be examined as an attempt at purposeful fracturing of an object with a robotic actuator, which seems to be a surprisingly underexplored topic in robotics research on manipulation. The work of [174] seems to be the single instance that studies this specific problem and is based on material failure theories and optimization techniques on grasp quality measures.

3.2 Mushroom Modeling & Characterization

3.2.1 Mushroom Modeling

An analytical modeling of the mushroom root system introduces a set of challenging requirements, given the anisotropic elastic material in combination with the failure modes that must be predicted during execution. In addition, we add the requirement of fast execution time for fast evaluations for potential utilization of Reinforcement Learning (RL) and Model Predictive Control (MPC) techniques. Ideally, for this type of system, we would choose an analytical finite element method (FEM), but the execution time would be prohibitively

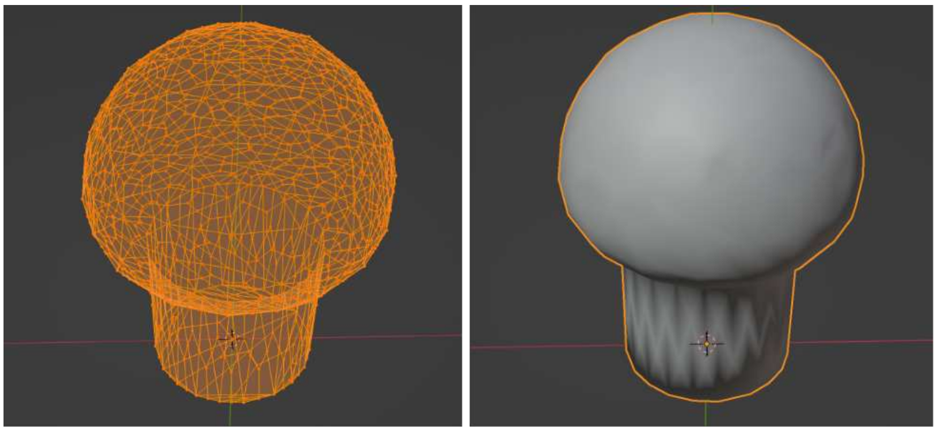


Figure 3.2 Mushroom 3D Model wire & object diagram.

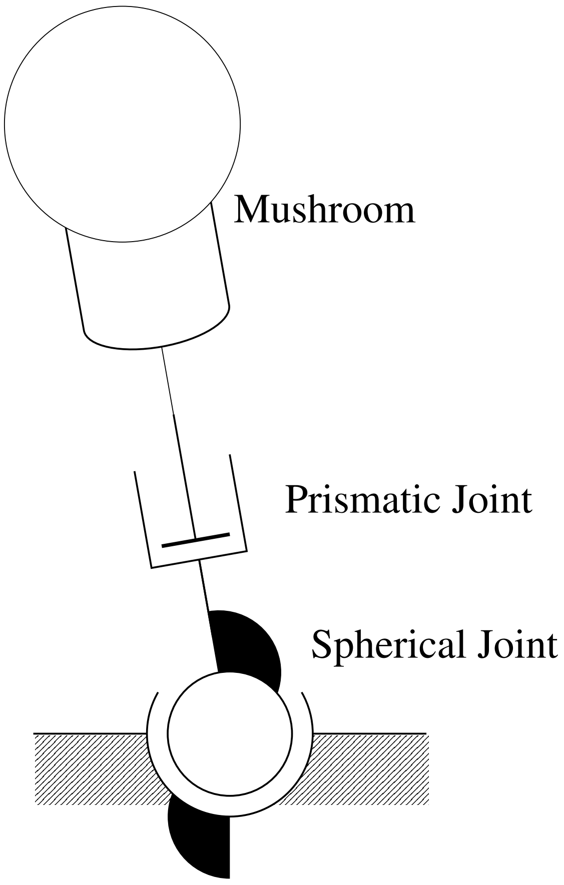


Figure 3.3 The Mushroom Model Mechanical Schematic Drawing.

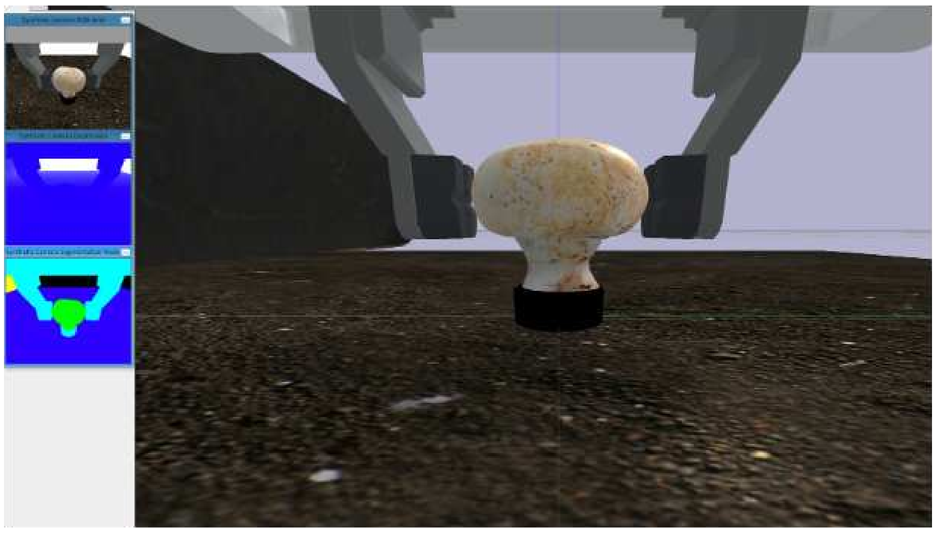


Figure 3.4 The PyBullet debug GUI view of the simulation framework for mushroom harvesting.

slow. Given the empirical knowledge that ultimately the mushroom-picking maneuvers from human experts seem to distill around a basic set of primitives, namely “bending”, “twisting”, and “pulling” around the root, we suggest that capturing a set of first-order approximations of the continuum mechanics models of the mushroom root system should suffice. Based on the work found in [157, 175, 176] , we result in a mushroom root model as a combination of a spherical and a prismatic joint, controlled with a fixed position PD law to emulate the stiffness and damping characteristics (see fig. 3.3) of elastic deformation. To calculate the stress threshold necessary for material failure and the consequent detachment of the mushroom from the ground, the VonMises criterion is utilized. Lastly, instead of incorporating the complete robot model, we only include the gripper by defining a fixed constraint in the flange frame in the 3D space, as suggested in [50]. For the implementation of the proposed model, we rely on the PyBullet rigid multibody dynamics physics engine, as it easily provides the ability to add and remove fixed constraints between objects during run-time.

The axial, rotational, and bending stiffness components of an isotropic cylindrical rod representing a mushroom-root model are defined as follows.

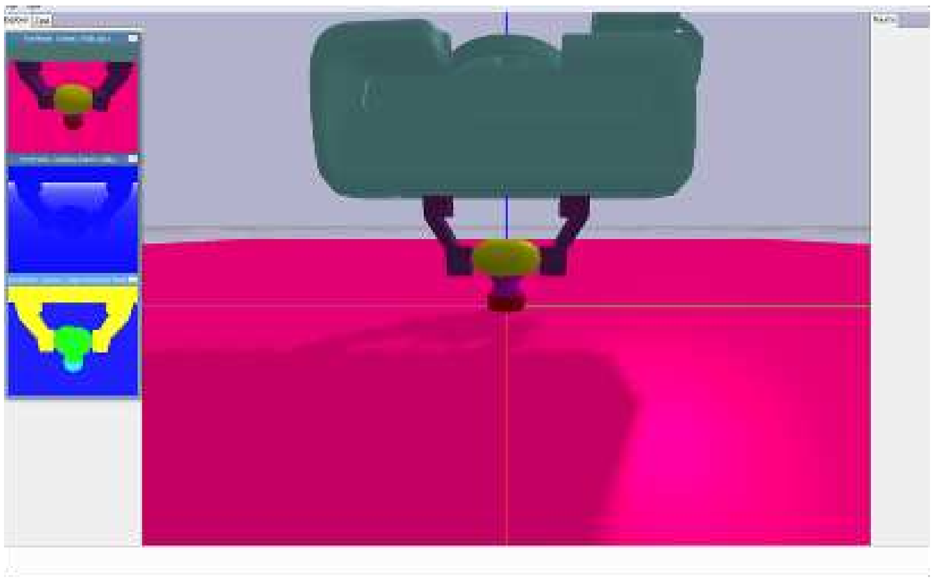


Figure 3.5 Simulation environment with randomized colors.

Table 3.1 Mushroom Model Parameters Definitions

Mushroom Model Parameters	**Values**
pose	$[p_x, p_y, p_z, \omega_x, \omega_y, \omega_z]$ m, rad
mass	m Kg
cap/stem diameter & height	$[c_d, c_h, s_d, s_h]$
young modulus	Y_m Pa
poisson ratio	P_r
yielding stress	Y_s Pa
cap contact stiffness	c_k N/m
cap contact damping	c_d Ns/m
cap restitution	c_r
cap lateral friction	c_c
cap spinning friction	c_{sc}

$$K_{\text{pull}} = \frac{EA}{I}, \qquad K_{\text{rot}} = \frac{GJ}{I}, \qquad K_{\text{bend}} = \frac{GA}{I}L^2$$

Where E, G are the Young and Shear Modulus terms and A, I, L are the axial surface area, second moment of area, and length, respectively. For bending, the definition is for transversal action given the relatively short length of the mushroom stem.

The VonMises stress factor for a cylindrical rod under shear and axial stress is defined as follows.

$$\sigma_{VM} = \left(\sigma_x^2 + 3\tau_{zx}^2\right)^{1/2}$$

Failure and therefore mushroom yielding will occur when the VonMises stress factor exceeds the yielding stress factor value, namely:

$$\sigma_{VM} > S_y$$

Regarding the damping terms of the PD controllers, we manually tune them in order to approximate critical damping characteristics.

3.2.2 Mushroom Model Parameters Identification (SysID)

To tune the stiffness parameters and determine the yielding stress threshold, we proceed with a series of experiments. We attach a Touchence Shokac chip force sensor to the index, middle fingers, and thumb of an expert mushroom picker and a Phasespace-led tracker to the mushroom cap to determine the relative displacement. The goal here is to identify the stress-strain diagrams for distinctive "pulling", "bending", and "twisting" motions (see fig. 3.11 fig. 3.12 fig. 3.13 fig. 3.14). A similar approach has been adopted by [166] where an IMU unit was used for position tracking.

Stress - Strain experiments results for mushroom root stress characterization (SysId). We derive critical modeling parameters for the Young modulus, Poisson ratio, and yielding stress with simple linear regression over the close to linear stress-strain regions.

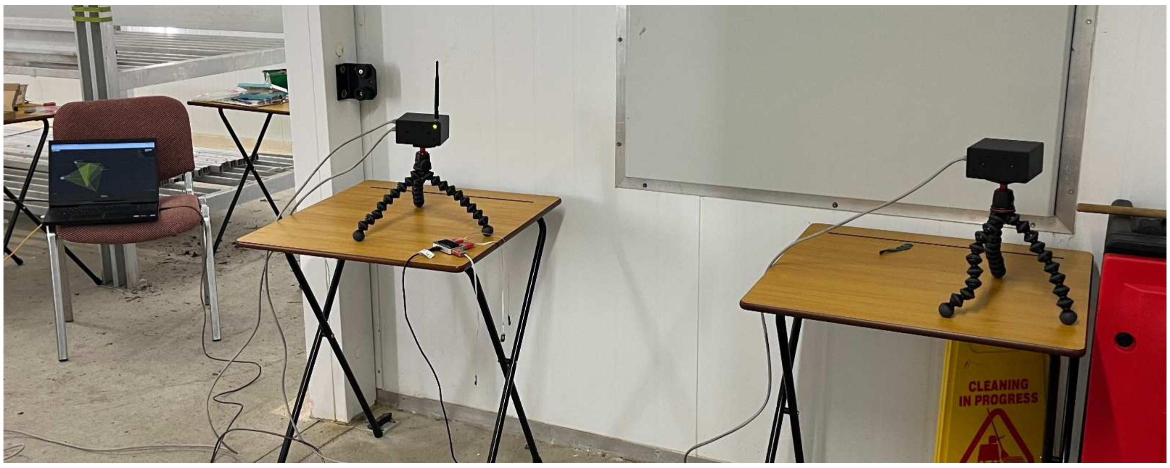


Figure 3.6 Phasespace equipment experimental setup.

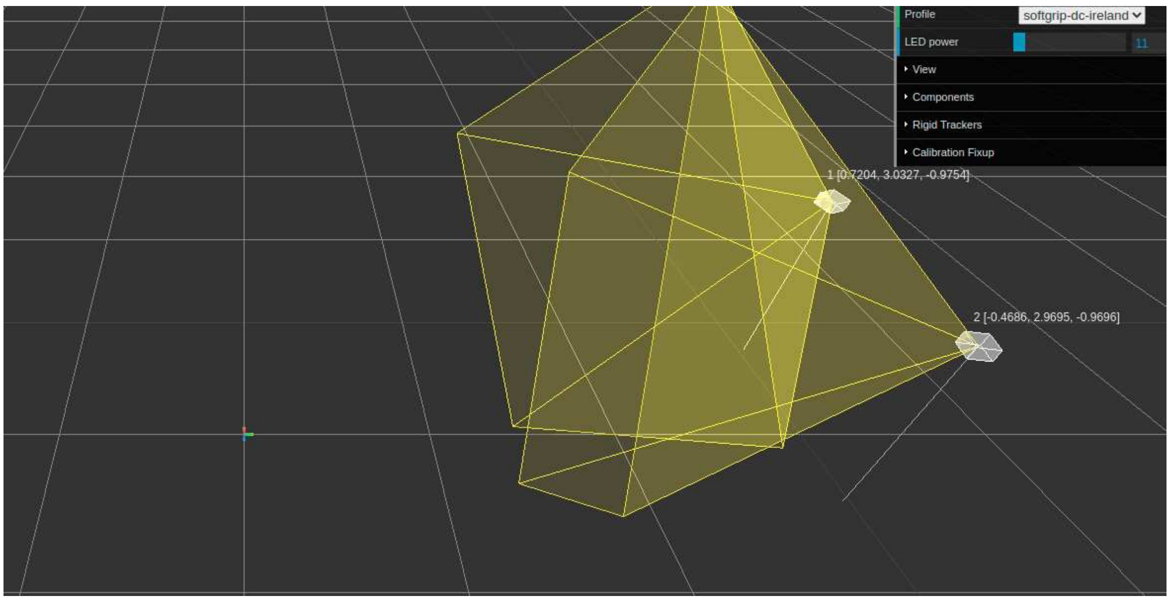


Figure 3.7 Phasespace working space capture.

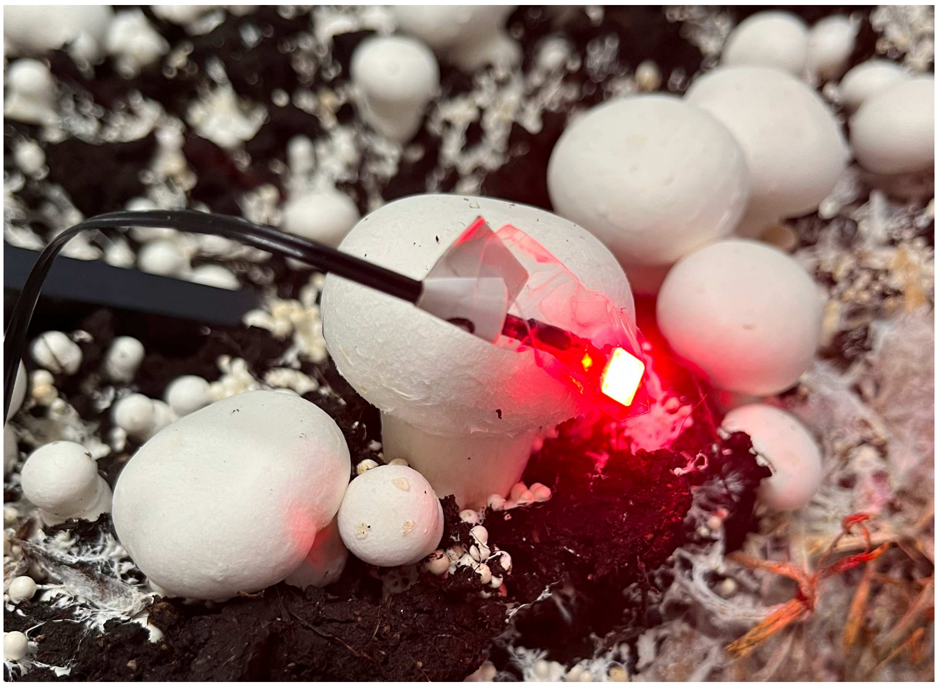


Figure 3.8 Phasespace LED attached on mushroom cap.

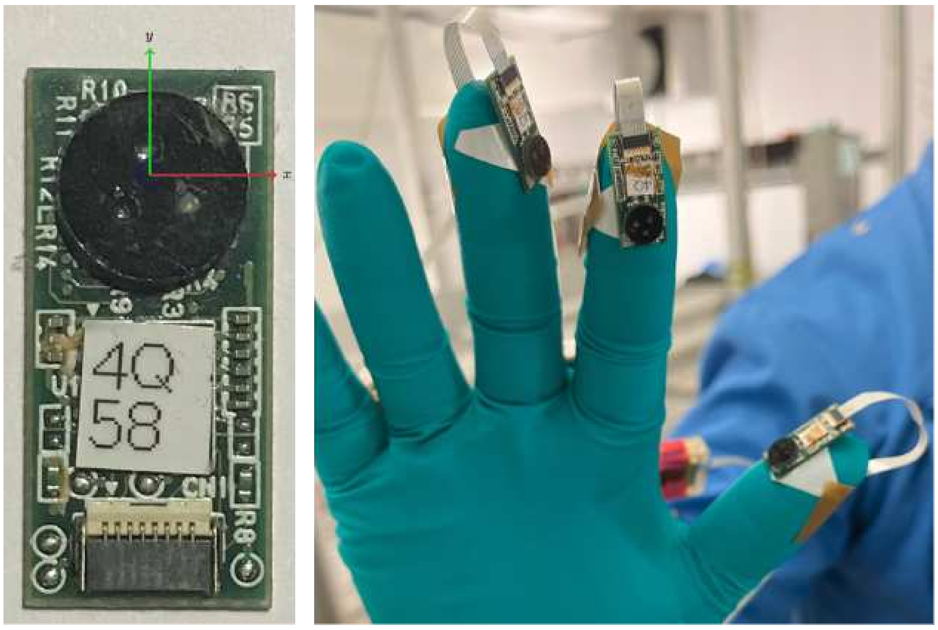


Figure 3.9 Tekscan force sensor attachment on mushroom expert picker.

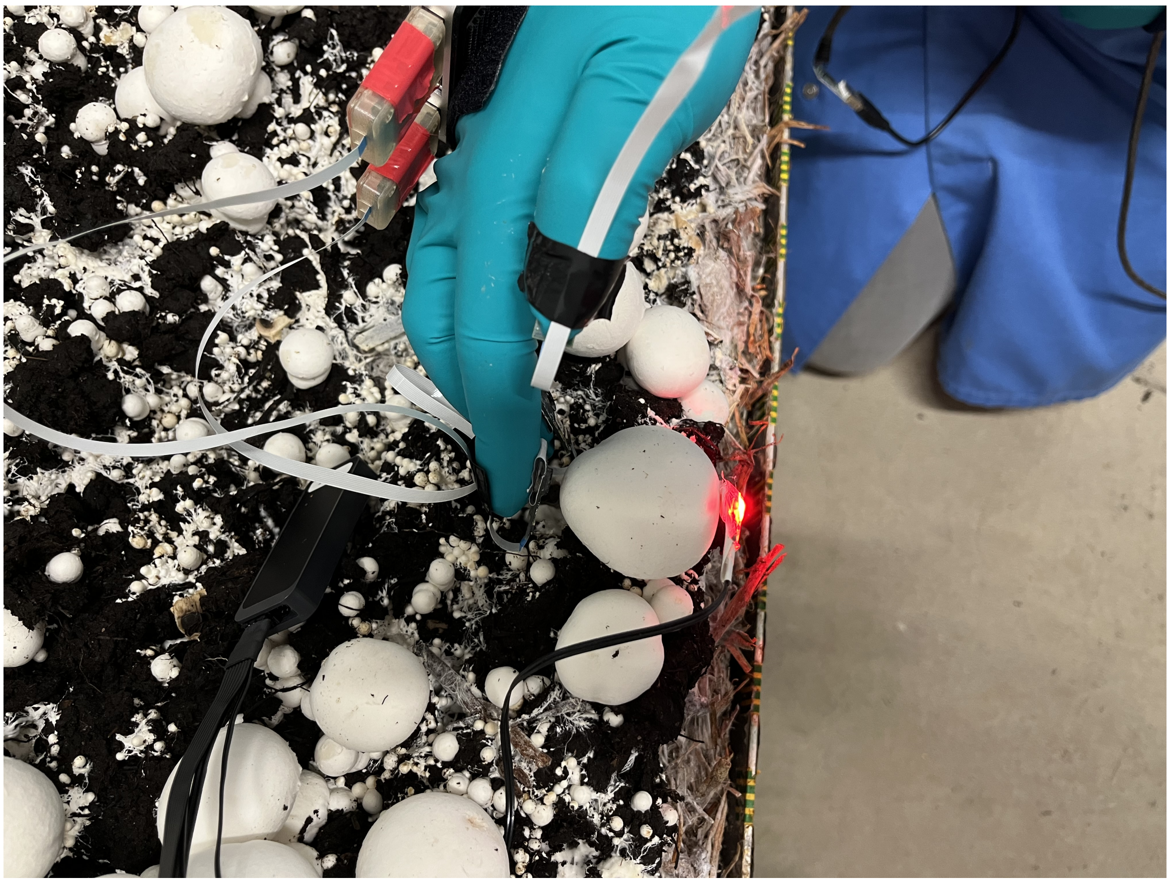


Figure 3.10 Experimental configuration for root stiffness characterization during bending action with a Touchence Shokac Chip force sensor attached on the finger that will perform the action, and the Phasespace tracker led for measuring the total displacement.

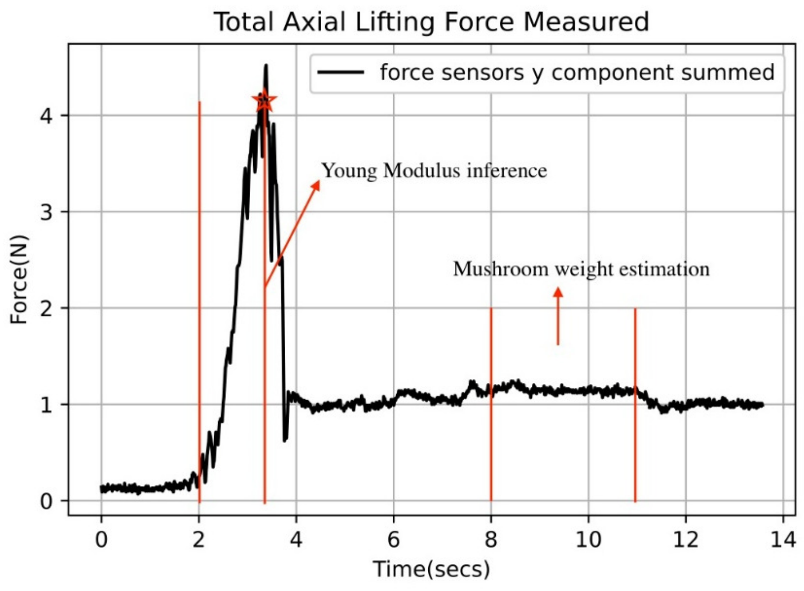


Figure 3.11 Mushroom tensile testing plot.

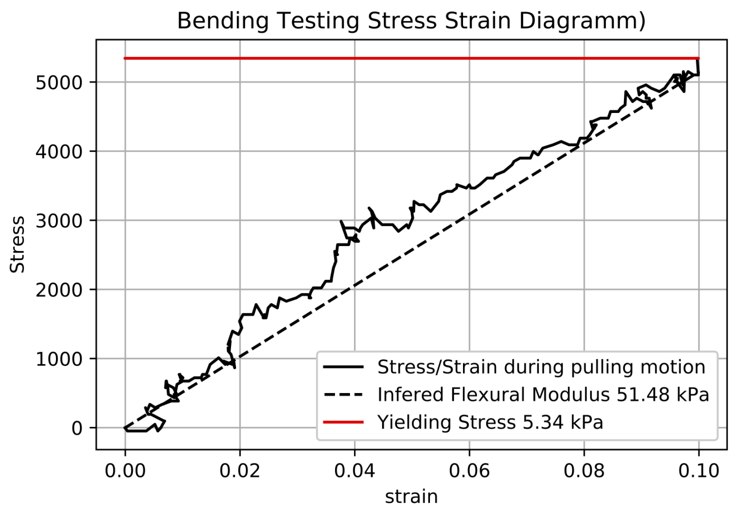


Figure 3.12 Mushroom bending stress strain diagram.

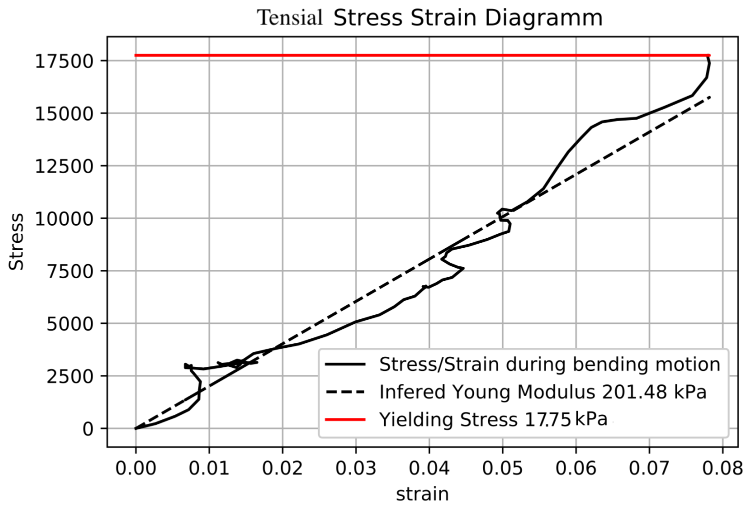


Figure 3.13 Mushroom tensile stress strain diagram.

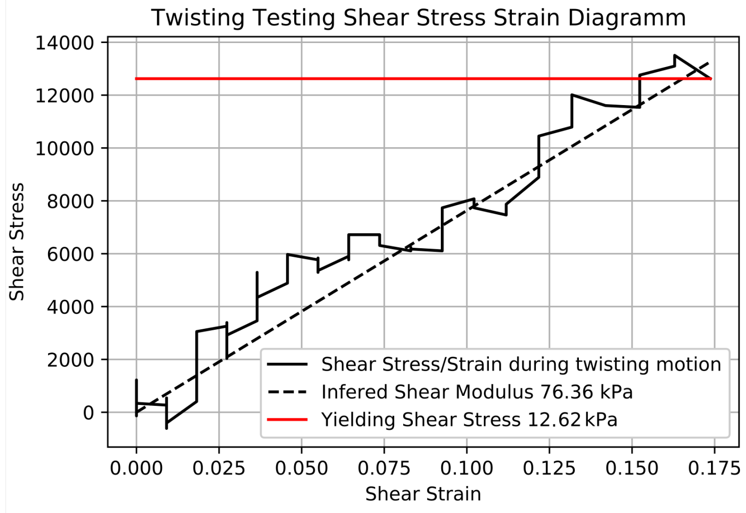
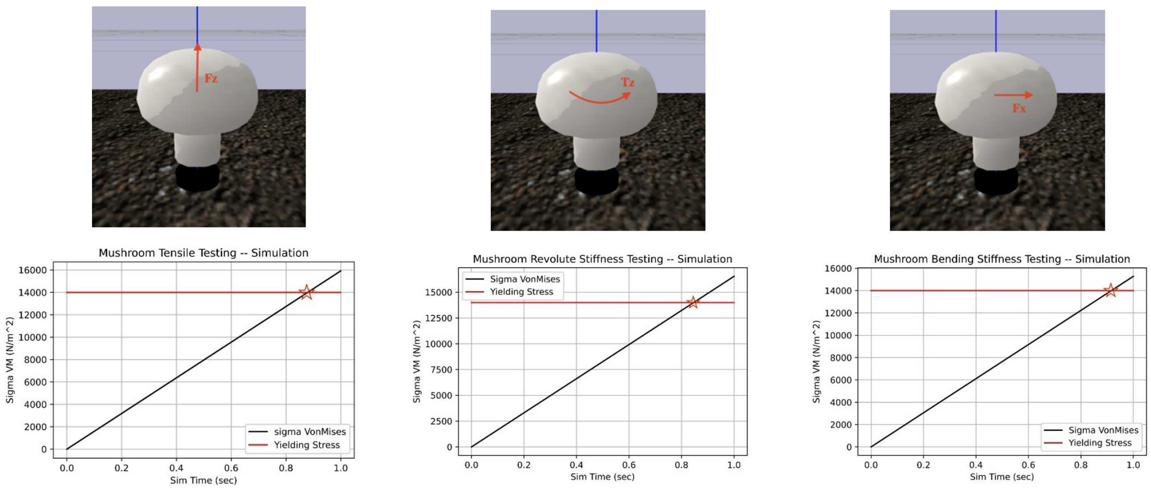


Figure 3.14 Mushroom twisting shear stress strain diagram

In practice, we found that by setting values for the Young Modulus that resemble the real world, namely, on the order of magnitude of 0.5 MPa, the system becomes too stiff, which in turn requires a sample rate dt for the physics engine at 0.1 ms for ensuring stability, rendering the simulation 2 times slower than the real world clock time. As the design purpose of this simulation as a dynamic model is to derive a kinodynamic plan with the motion primitives for mushroom picking, we can scale the stiffness to the order of magnitude of 10kPa, resulting in a kinodynamically equivalent model that is $1/10$ faster than the real-world clock time.

Lastly, we found that the bending action resulting from the lineal isotropic model defined above is too stiff compared to the experimental results, as also confirmed in [167]. We simply readjust to emulate the anisotropic properties as $K_{\text{bend adj}} = 1/5K_{\text{bend}}$.



3.3 Soft Gripper Modeling

For the needs of the EU Project "SoftGrip" [11] we need to incorporate a soft gripper (see fig. 3.15) in our simulation framework for the task of mushroom harvesting.

Modelling of soft, deformable elements within the robotic manipulation context is a complex tasks, as usually for an accurate modeling of such elements analytical Finite Element

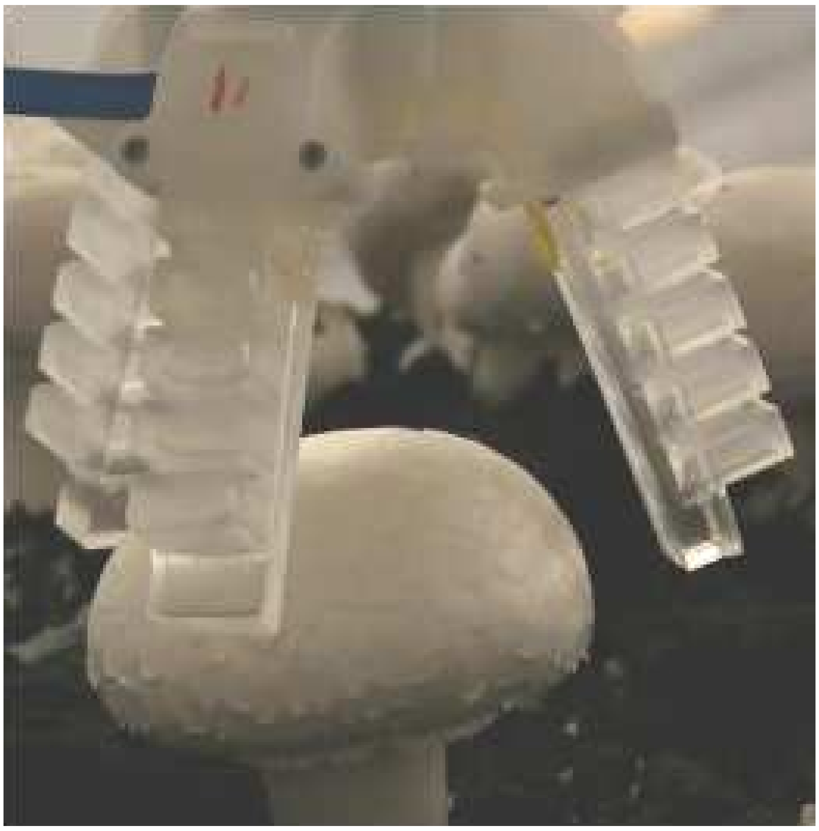


Figure 3.15 Soft Gripper Real-World Hardware Prototype

Analysis (FEA) methodologies are incorporated. As these models offer high-fidelity they are usually slow and require significant domain expertise in the development stage. Similarly to the mushroom modeling, we proceed with a “Line-Segement Model” [177], which serves as a first order approximation of the equivalent continuum dynamics model [176]. We are able to readily design the soft fingers URDF files and incroprorate them in PyBullet using the SoMoGym Framework [157]

3.3.1 Soft Gripper Dynamics with a Line-Segment Model

We proceed with modeling of the soft fingers using the line-segment model [177] (see fig. 3.16) as a first order approximation of the continuum mechanics equivalent model [176].

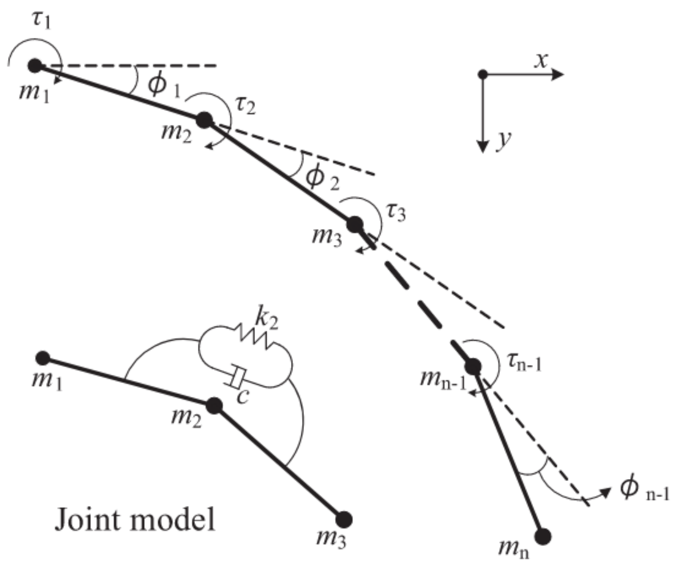


Figure 3.16 Soft Gripper Dynamics Modeling Using a Line-Segment Model [177]

We use SoMoGym [157] for generating the URDF file and for controlling the soft finger, inspired by the "antipodal gripper" example (see fig. 3.17) inside the code base.

We test the soft finger by applying a ramp function as an input (see fig. 3.18) in order to test the system response as a pressure vs angle (fingertip) (see fig. 3.19).

We proceed with a final integration of the soft fingers at the rigid gripper base (see fig. 3.20). The URDF file for the complete soft gripper is manually crafted, with a manual integration of the auto-generated URDF files of the soft-fingers as described above.

3.3.2 FEM Finger Models and PyBullet Testing

It is important to mention that although PyBullet does incorporate the ability to import meshes for the simulation of deformable objects, initial tests of ours indicated that it was not possible to proceed by this approach. We found the simulation time to be extremely slow for practical control purposes in a MPC or a RL context and additionally it proved impossible to apply grasping with soft fingers to any type of object, as this would result in penetration or

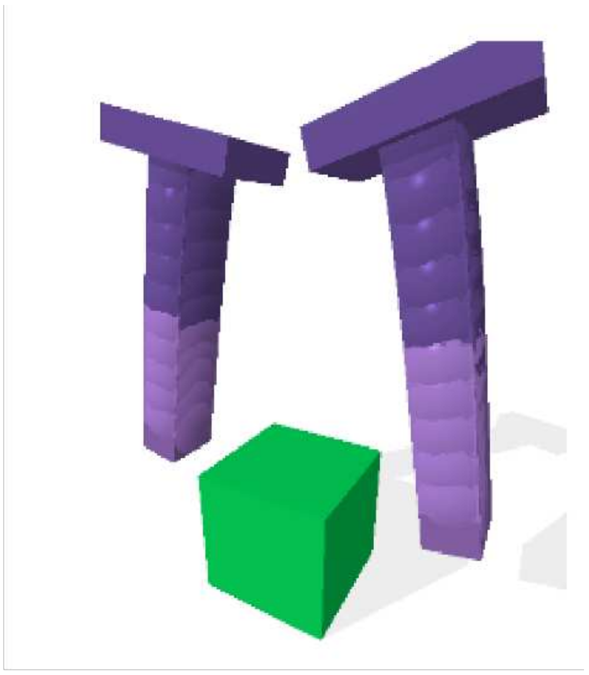


Figure 3.17 Somogym anti-podal gripper example [157].

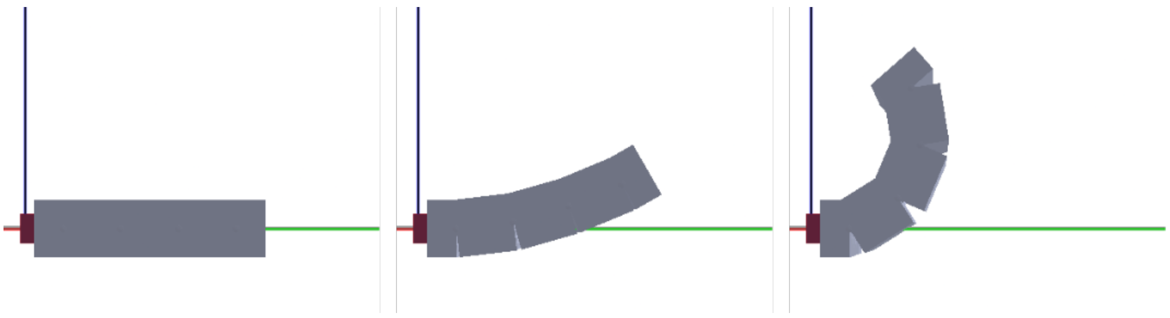


Figure 3.18 Soft finger step response in simulation.

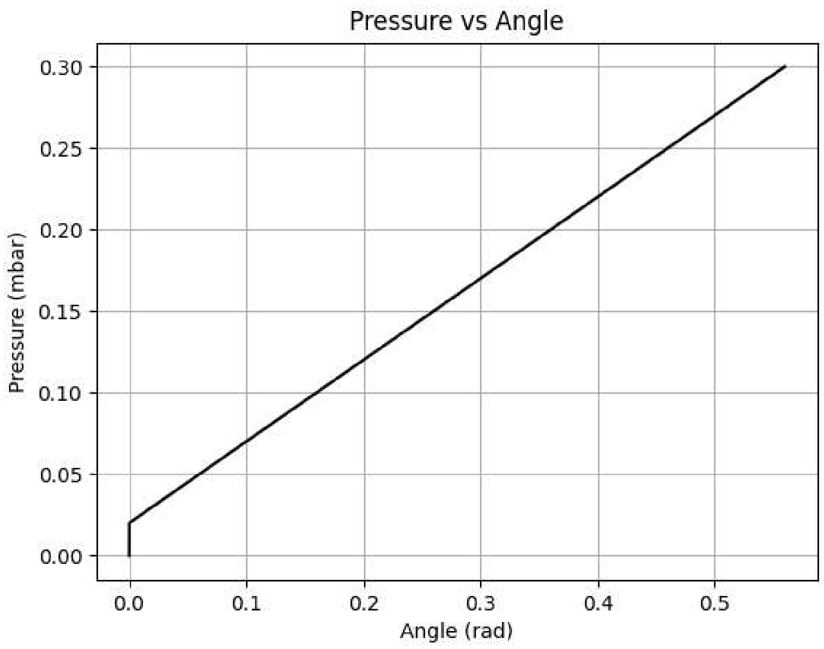


Figure 3.19 Soft finger step response in simulation. Pressure vs Angle

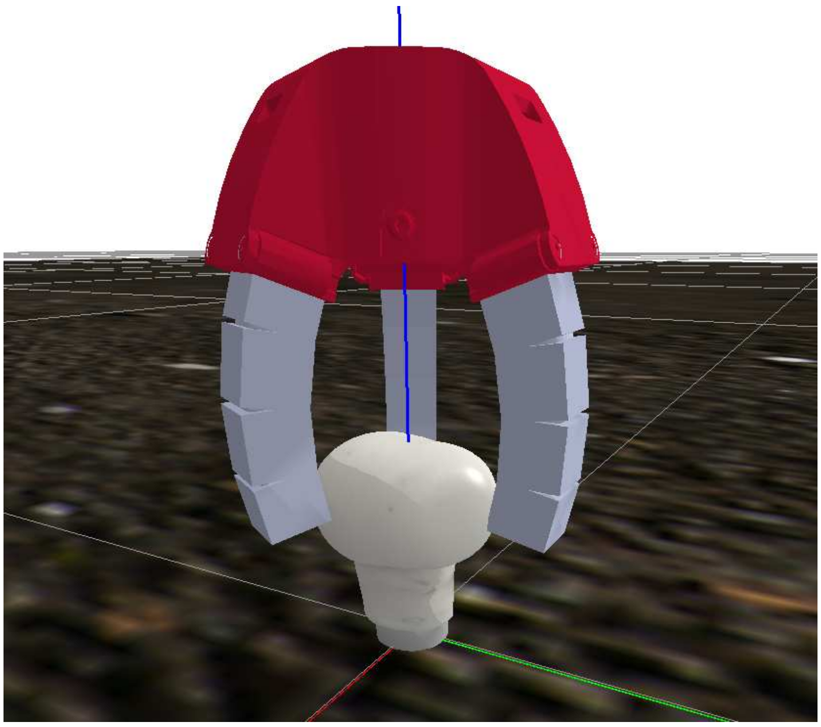


Figure 3.20 Complete System Soft Gripper and Mushroom Model in PyBullet simulation

the object slipping even when high closure forces were applied. This finding seems also to be supported in [54].

3.3.3 Potential Future Directions

3.3.3.1 System Identification

Further experiments with the real-world setup should be conducted for validating the model design. Black-box techniques or differentiable end-to-end architectures [153] for system parameters identifications could be utilized in that case.

3.3.3.2 Physics Engines alternatives

We could experiment with different physics engines, specialized for simulating deformable objects for robotic manipulation, like [155].

3.4 Real-World Hardware & Software Setup

3.4.1 Robot Manipulator

We use the Franka Robot [178] for conducting our real-world experiments (see fig. 3.21) as with it's 7 DOFs allows for a redundant 6DOF control of it's end-effector, and it is additionally equipped with a simple 1 DOF rigid gripper. Moreover, the Franka Robot is equipped with torque sensors in each joint and thus allowing for direct impedance control schemes as operational space controller (OSC) which expands significantly the envelop for forceful robotic manipulation. Lastly it's API's in C++ and consequently in ROS (and Simulink/MATLAB), allow for direct and seamless software integration.

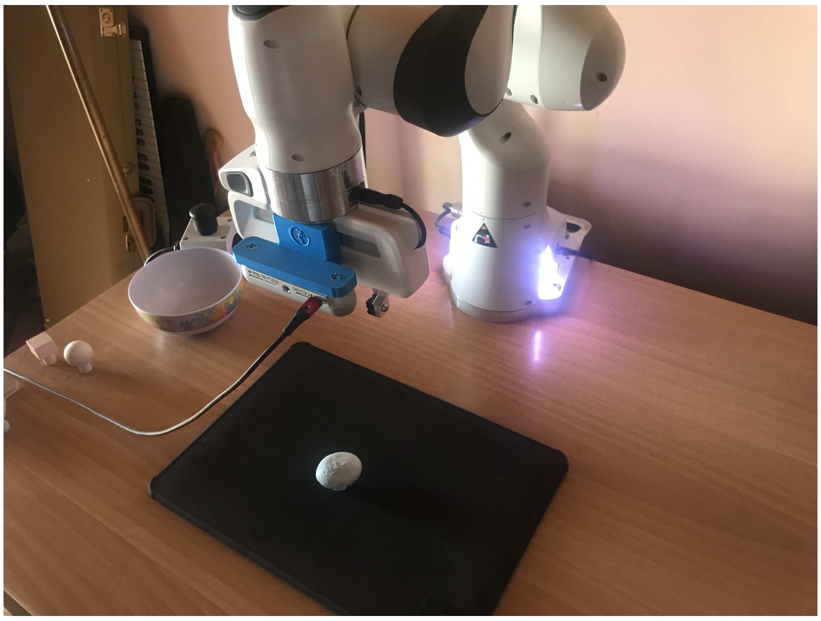


Figure 3.21 Franka robot real-world setup.

3.4.2 Vision System

For the vision system we rely on Intel's real-sense cameras, namely the 455 [179] and the 405 [180] models, for complete 3D workspace reconstruction (see fig. 3.21) and fine-manipulation in close proximity (see fig. 3.22) respectively.

3.4.3 Computation

As the Franka Robot requires Real-Time Linux Kernel for control in the 1kHz loop, we use a NVIDIA's Jetson Nano [181] with RT_PREEMPT Kernel patch for running the Franka ROS [182] control node.

The main workstation PC is a generic linux laptop or PC with intel cpu and an NVIDIA's GTX graphics card.

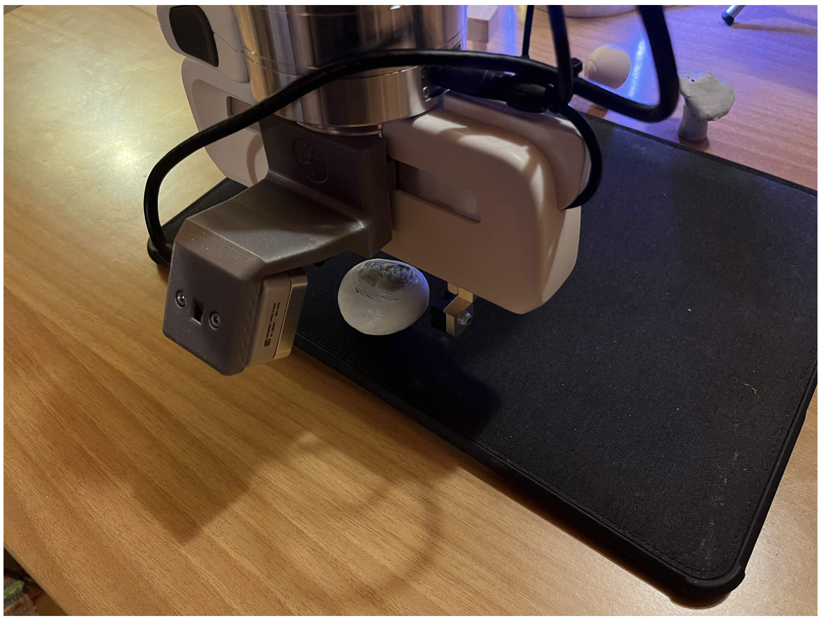


Figure 3.22 Franka Robot with the Intel Real Sense 405 mounted.

3.4.4 Software Architecture

We build the complete software stack around the Robotic Operation System (ROS) and we utilize the MoveIt! motion planning framework for [183] generating robot motions and to perform hand-eye calibration for determining the camera extrinsic properties.

3.5 Conclusion

3.5.1 Limitations & Future Work

The proposed simulation framework, built on first-order approximations and a rigid multi-body physics engine, offers a balance between computational efficiency and policy training effectiveness. However, this approach introduces inherent trade-offs in fidelity and generalization. The framework's complexity stems from model simplification, which linearizes system dynamics. While computationally efficient, this method restricts applicability to

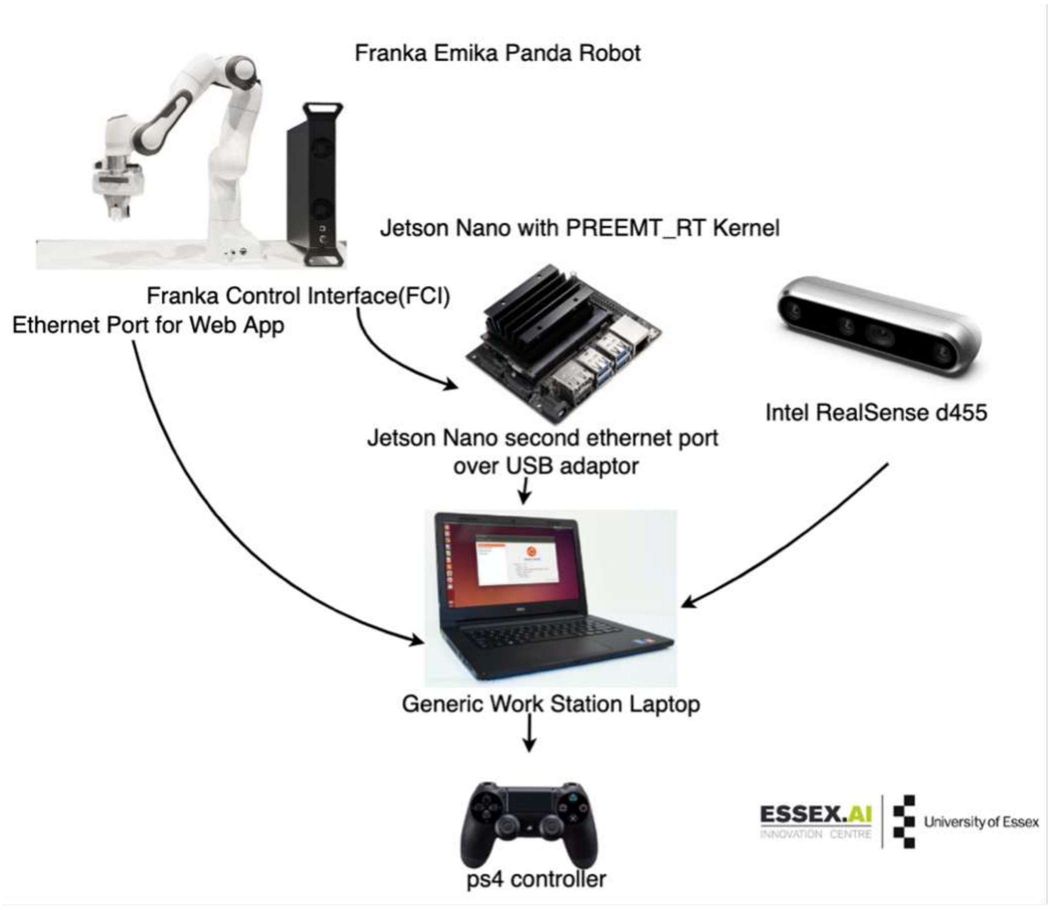

Franka Emika Panda Robot
Jetson Nano with PREEMT_RT Kernel
Franka Control Interface(FCI)
Ethernet Port for Web App
Intel RealSense d455
Jetson Nano second ethernet port over USB adaptor
Generic Work Station Laptop
ps4 controller
ESSEX.AI INNOVATION CENTRE
University of Essex

Figure 3.23 Hardware configuration for the real-world experiments.

regimes where nonlinearities and inertial coupling remain negligible, and limits parameter diversity primarily to geometric and kinematic variations.

Key limitations of the framework include reduced dynamic fidelity in high-acceleration or contact-rich tasks, narrowed randomization scope for deformable objects, and potential exploration bias in complex scenes.

Future work will focus on integrating second-order sensitivity analysis and continuum mechanics priors to address these limitations while maintaining real-time performance [184, 185]. , thereby enhancing the framework's applicability to a broader range of robotic tasks and environments.

3.5.2 Generalization. Beyond Mushroom Harvesting.

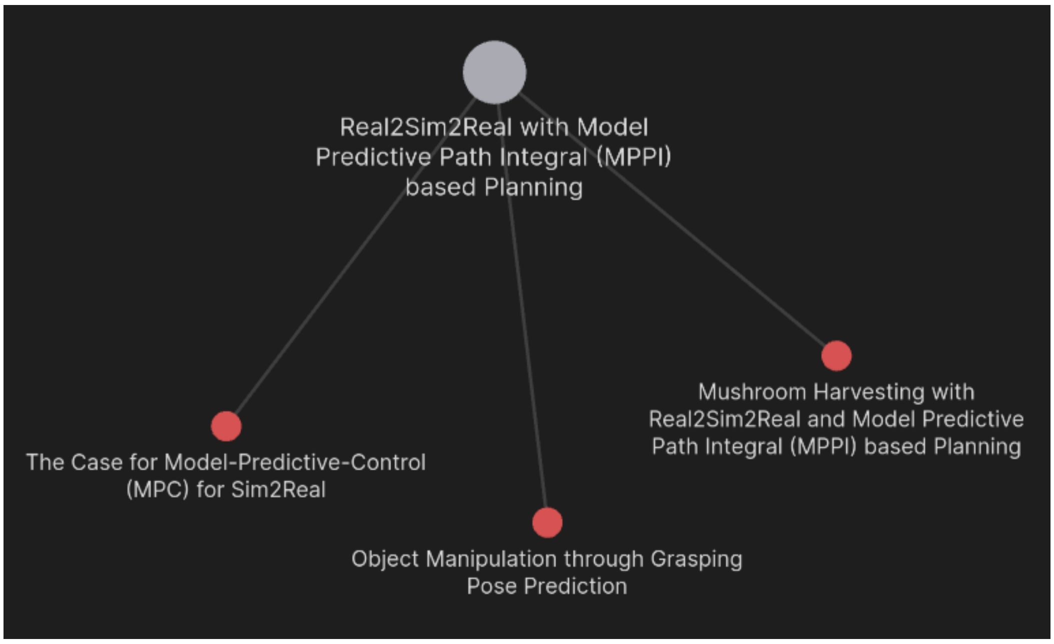
The analytical framework employing first-order approximations and adaptive domain randomization for rigid multibody systems exhibits broad applicability beyond agricultural robotics. In **soft robotics**, where high damping and low inertia dominate dynamics [186], this approach could streamline real-time control of deformable actuators while maintaining computational efficiency—critical for applications like minimally invasive surgical tools or underwater exploration systems. Similarly, **precision agriculture** could leverage these methods for digital twins simulating soil–root interactions or crop responses to environmental stressors, enabling predictive analytics for irrigation and nutrient management [187]. The framework's ability to balance accuracy with computational cost also benefits **industrial automation**, particularly in handling delicate objects through soft grippers that require rapid adaptation to material variability [188, 189].

Chapter 4

Real2Sim2Real with Model Predictive Path Integral (MPPI) based Planning

"No man ever steps in the same river twice, for it's not the same river and he's not the same man."

—Heraclitus, *Fragments*


Real2Sim2Real with Model Predictive Path Integral (MPPI) based Planning
The Case for Model-Predictive-Control (MPC) for Sim2Real
Object Manipulation through Grasping Pose Prediction
Mushroom Harvesting with Real2Sim2Real and Model Predictive Path Integral (MPPI) based Planning

4.1 Grasping Pose Prediction for Object Manipulation

In an attempt to approach the problem of mushroom uprooting with a robotic manipulator we analyse the well-established field of “Grasping Pose Prediction” with robotic grippers. This sub-field of robotic manipulation deals with the problem of an automatic prediction and evaluation of an optimal grasp pose given an object or a complete scene and a gripper. The complete task of robotic manipulation is then based on this predicted grasping pose and by usually employing traditional motion planning.

Grasping Pose Prediction is useful in pick-and-place and scene decluttering scenarios. For more dexterous and delicate manipulation tasks which impose strict dynamics constraints and require complex kinodynamic planing in hybrid dynamic spaces more elaborate robotic manipulation approaches should be employed, which usually stem from the disciplines of “Task and Motion Planing” (TAMP), “Forceful Robotic Manipulation” or more lately end-to-end visuomotor policies with Reinforcement Learning.

4.1.1 Deep-Learning for Grasp Prediction

Optimal Grasping Prediction has been a long standing research topic in robotics and still remains an open research topic. Till recently the predominant approach in grasping have been analytical and sampling based approaches based on optimization of some force closure quality evaluation metric. The GraspIt! framework [190] encapsulates the core results of this era and still remains today a golden standard and a commonly used baseline.

A combination of major breakthroughs in data-driven, end-to-end, deep architectures in Machine and Reinforcement Learning, the introduction of advanced perception and vision sensors and apparatuses in combination with powerful computational resources have naturally brought great research interest in grasping prediction as well.

Levine et al. [191] by purely utilizing an end-to-end, pixels-to-velocities approach based on a Convolutional Neural Network (CNN) and gathering a large number of trials from real-world robots they train an agent able to autonomously grasp random objects from a bin without then need of a manual hand-eye camera calibration.

Bousmalis et al. [75] as an attempt to mitigation the issue of the extreme cost of real-world data sampling, they employ domain adaptation with Generative Adversarial Network (GAN) for realistically rendering synthetic images for a consequent successful sim2real transfer. Their pipeline is based in observation from a single monocular raw camera.

Pas et al. [192] are able to work on the point cloud generated from a cluttered scene, without the need for an explicit object segmentation, by extracting a "Region of Interest (ROI)" based on heuristics, sampling several thousand grasp candidates on that region, and finally assigning a grasping score on each grasp with the help of a convolution neural network.

One of the most concrete lines of work in the field and as an attempt to establish a foundational model for grasping pose prediction is the case of the "Dex-Net" [193–197] which makes again heavy usage of training in simulation, where by utilizing the privileged information, a large data of training data can be generated with expert grasping demonstrations.

Dyrstad et al. [49] showcase how human input can help in generate grasping samples which can then be further enriched with data enhancement methodologies for training a deep grasping network.

Sundermeyer et al. [198] with the Contact-Grasp-Network learn to predict grasping poses based on the point cloud of the cluttered scene. The training is taking place in simulation where multiple random, but stable, cluttered scenes are generated with the help of the ACRONYM dataset [199].

Weng et al. [200] with the Neural Grasp Distance Fields (NGDF) for grasping instead of a discrete set of grasping poses they predict a distance of the gripper to valid set of continuous grasping poses on the object. This representation, which can be interpreted as a cost, it allows for a conjoint overall manipulation planning, which is a unique element.

At the core of most of the aforementioned techniques lie the simulated datasets, such as [199], which contain grasping picking demonstrations. Want et al. [201] expand the envelope with the DexGraspNet, which contains “1.32 million dexterous grasps for the Shadow Hand on 5355 objects”.

4.1.2 An Evaluation of The Volumetric Grasping Network (VGN)

We choose to evaluate the Volumetric Grasping Network [50] as a candidate for grasping pose generation for mushroom picking. This approach has some key properties that render it a a good candidate for our intents and purposes, namely:

1. Direct Grasping Pose Generation from a Point Cloud without the need for an explicit object detection or segmentation in the clutter, which in our case is the mushroom crop.
2. The ability to evaluate a case of a direct, zero-effort Sim2Real approach due the explicit transferable abstraction through this pure geometric nature.

We start by evaluating the VGN in simulation, with the pretrained network, for the case of a single mushroom at first (see fig. 4.1, fig. 4.2). The pretrained network, even though it hasn’t been trained for single mushrooms, or mushroom crops, seems to be able to generate a grasping pose (see fig. 4.3), sub-optimal though as it may be.

We proceed with an evaluation on a real-world setup, again with the pre-trained VGN, in a scene which includes 3 mushrooms (see fig. 4.4).

VGN was able to generate a quite convincing set of grasping poses for each mushroom, assigning the best grasping score to an individual one (see fig. 4.5). Again this is quire remarkable, given that the network has only been trained in simulation only, with a quite different set of objects and cluttered scenes.

VGN, in it’s pre-trained form at least, does not seem to generate accurate enough grasping poses for mushroom grasping. These results nevertheless, allow us to exploit the VGN in

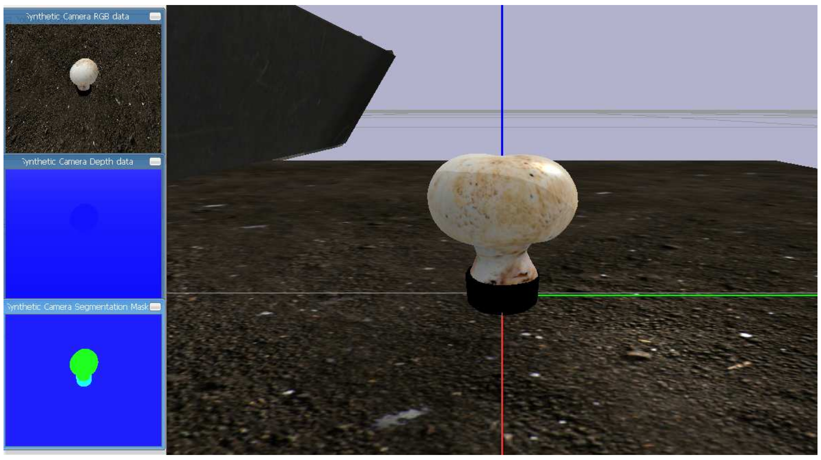

Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask

Figure 4.1 3D scene in simulation.

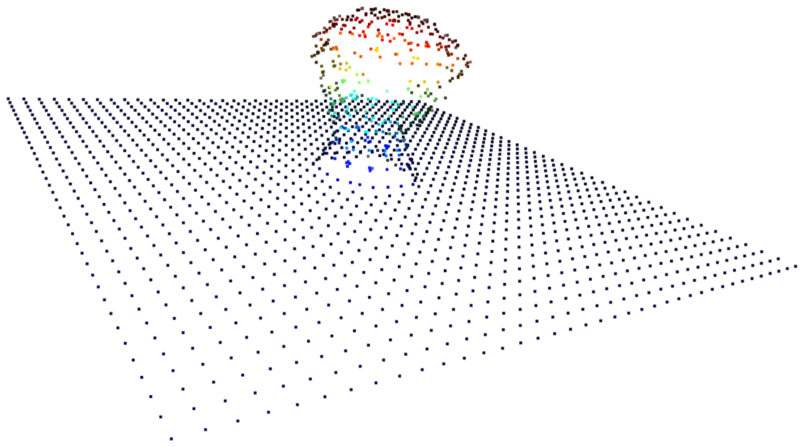


Figure 4.2 Point Cloud generated after scanning the scene and utilizing the Volumetric Grasping Network.

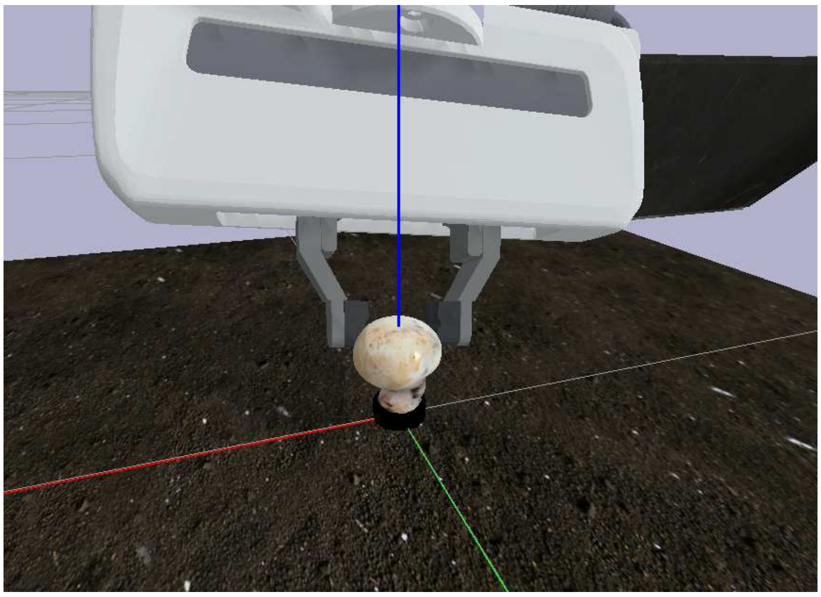


Figure 4.3 Grasping pose as predicted from VGN applied in Simulation.

our pipeline for mushroom harvesting, as we can select a good mushroom candidate for grasping within a surrounding mushroom cluster, something usually really hard to approach analytically.

4.1.3 Future Directions

An immediate improvement would be a retraining of the VGN, by skewing the data-set towards mushroom like crop scenarios. This should allow for the generation of precise and robust grasping poses for mushroom harvesting.

As we are dealing with a highly dynamic task, that includes strict constraints on allowed force application, in combination with dexterous forceful manipulation, within hard optimization in overall time and total yield production for a practical industrial application the grasping pose prediction is heavily conditioned and cannot be examined independently. More holistic frameworks, such as Task and Manipulation Planning for Forceful Manipulation or end-to-end RL schemes should be employed in that sense like in the cases of [202, 200].

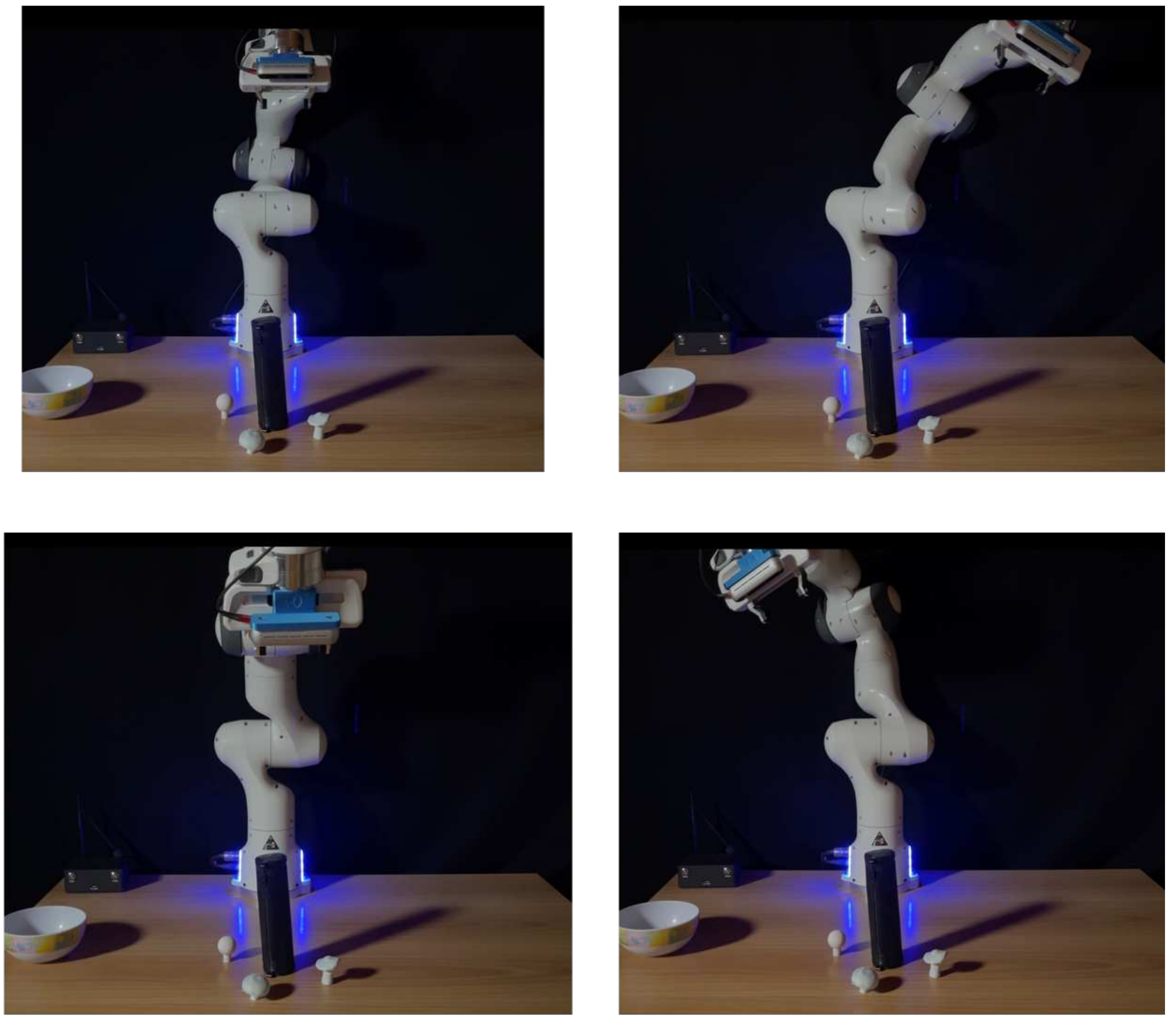


Figure 4.4 Poses for 3D Scene Reconstruction as Truncated Signed Distance Function (TSDF).

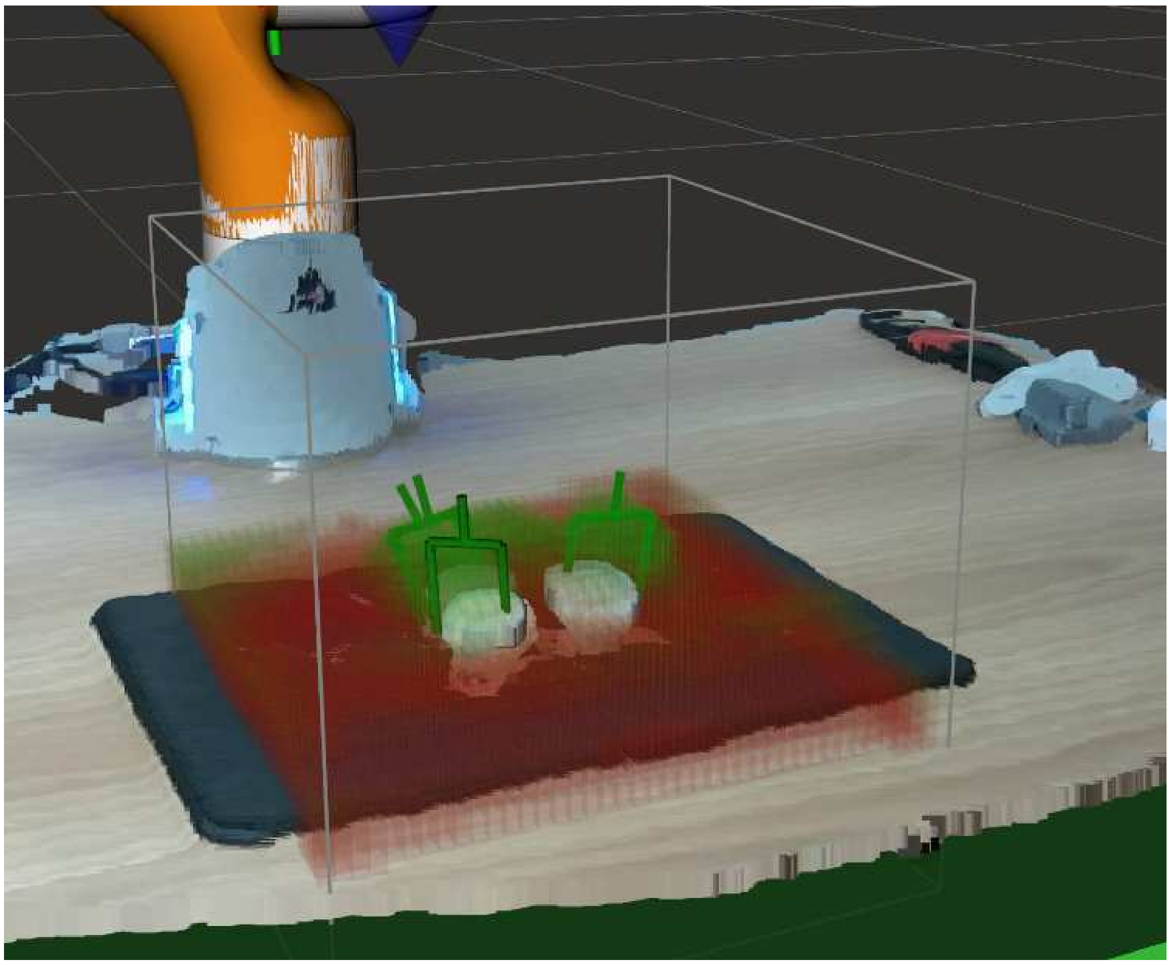


Figure 4.5 Predicted grasping poses with VGN as rendered in rviz.

4.2 Mushroom Harvesting with Real2Sim2Real and Model Predictive Path Integral (MPPI) based Planning

To extract the necessary scene semantics for an on-the-fly physics simulation scene instantiation (Real2Sim) we capture a set of depth images from a set of different poses which we then integrate with the Iterative Closest Point (ICP) for generating a Point Cloud (PC) and a Truncated Signed Function (TSDF) equivalent representation. We utilize these representations twofold, first to estimate the mushroom poses, using the pipeline developed by [203] and second for selecting a good mushroom candidate for grasping together with its close surrounding clutter for an on-the-fly generation of a simulation instance which acts as scene reconstruction and a model for our system where we perform the planning reasoning. For the mushroom & the consequent region selection, we make use of the pre-trained Volumetric Grasping Network (VGN) [50] which can generate a set of grasping pose candidates for scene decluttering.

Based on the scene reconstruction with the simulation framework, we can now reason in an offline manner to generate an optimal mushroom uprooting strategy. We utilize the MPPI control framework as a simple, yet effective [204–206] framework to solve the task of optimal mushroom picking in simulation and to carry out subsequent rollouts of the set of critical waypoints of the end-effector in the real-world setup. This framework allows for a definition of cost objectives for task shaping together with collision avoidance and grasping force minimization. It is also gradient-free, which fits well in the context of the inherent hybrid dynamic modalities involved in grasping & manipulation dynamics.

An overview of the overall system architecture is presented in fig. 4.6 fig. 4.7 [1].

During the first step of the pipeline, a sub-region of the mushroom crop is scanned to obtain a set of depth images $\{ \mathrm{D}_{\mathrm{imgs}} \}$ from various poses. The 3D point cloud and equivalent

[1]A video explaining the complete pipeline can be viewed at youtube.com/watch?v=k38ePBsBego

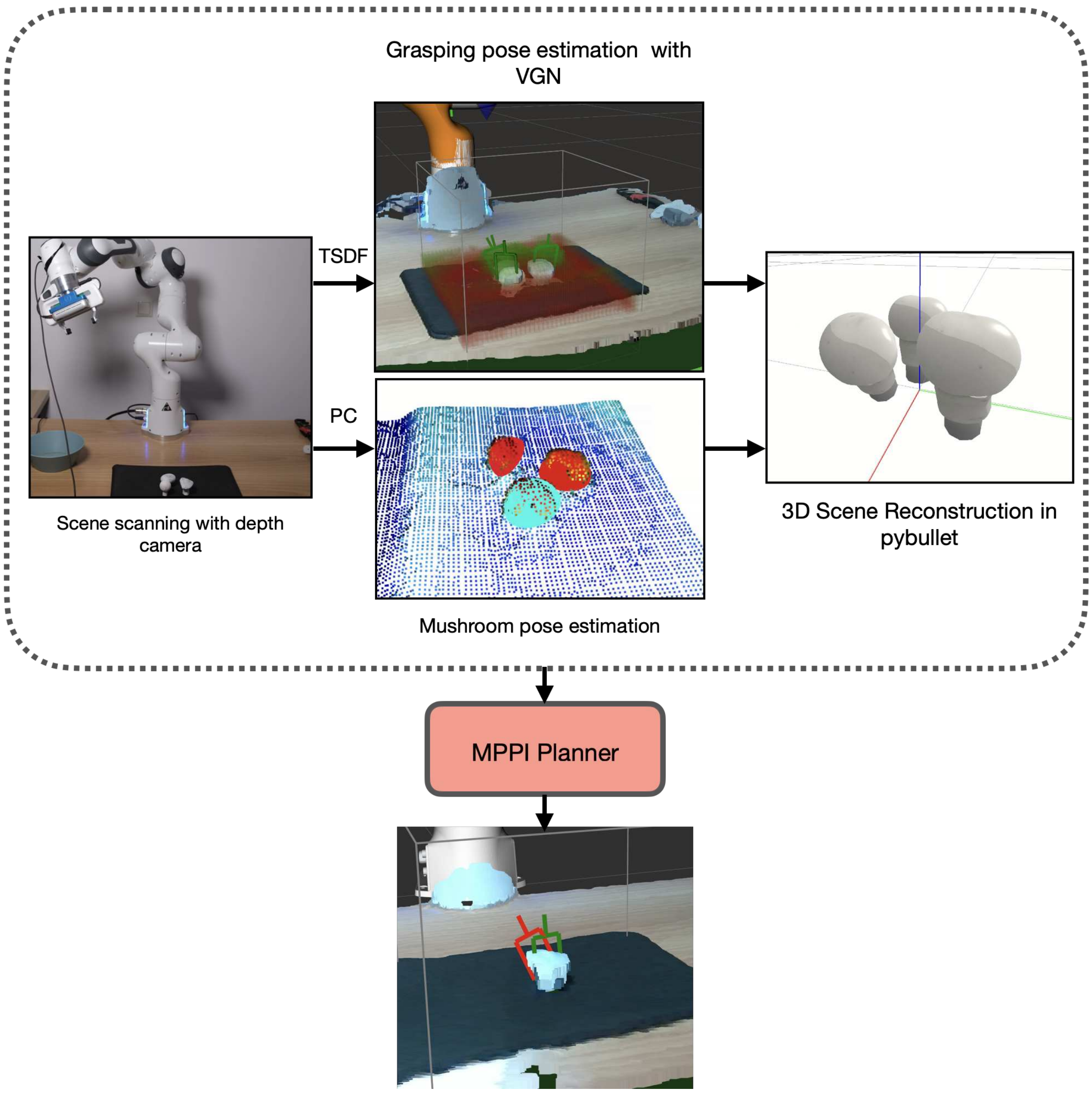

Grasping pose estimation with VGN
TSDF
PC
Scene scanning with depth camera
3D Scene Reconstruction in pybullet
Mushroom pose estimation
MPPI Planner

Figure 4.6 Mushroom harvesting with Real2Sim2Real and Model Predictive Path Integral (MPPI) overall system architecture.

TSDF representations are then obtained with the IPC method and the open3D library. We can then obtain a set of mushroom pose estimations with [203] and a set of grasping pose candidates with respective VGN score rankings [50]. We select the mushroom with the highest grasping score to be picked on the basis of the Euclidean distance between the grasping candidate from VGN and the mushroom pose estimates. By instantiating the "master simulation" scene equivalent and the MPPI planner we can extract a set of waypoints for the gripper which are descriptive of the optimal picking maneuver. We finally deploy to the real Franka robot by feeding the generated waypoints to the MoveIt! library which generates the final joint trajectories with RRT and inverse kinematics.

Algorithm 1 Autonomous Mushroom Harvesting

```
 1: while mushroomsPosesDetected do
 2:     { D_imgs} ← sceneScan()
 3:     PC, TSDF ← get3DPointCloudWithIPC({ D_imgs})
 4:     MP, MS ← mushroomPoseEstimation( PC)
 5:     GP, GPR ← graspingPosesEstimation( TSDF)
 6:     TMP, SMP ← selectTargetMushroom( MP, GP)
 7:     Sim ← instantiateMasterSimulation( TMP, SMP)
 8:     GWP ← graspingManueverWithMPPI( Sim)
 9:     TRJ ← trajectoryGenWithRRTandIK( GWP)
10:     rolloutTrajectoryToTheRealWorld( GTR)
11: end while
```

4.2.1 Optimal uprooting planning with MPPI

To generate an optimal mushroom picking planning, we base our implementation on the MPPI controller in [205]. Our simulation framework is implemented in PyBullet instead of NVIDIA's Isaac gym, as we make use of the active constraint removal (for mushroom detachment) during the simulation run-time, which is not available in the latter. We modify therefore the implementation for sampling from our simulation environment in parallel in CPU as pybullet is not GPU parallelizable. This modification does not allow for real-time control; therefore, we do use a "master" simulator where we can extract a set of waypoints,

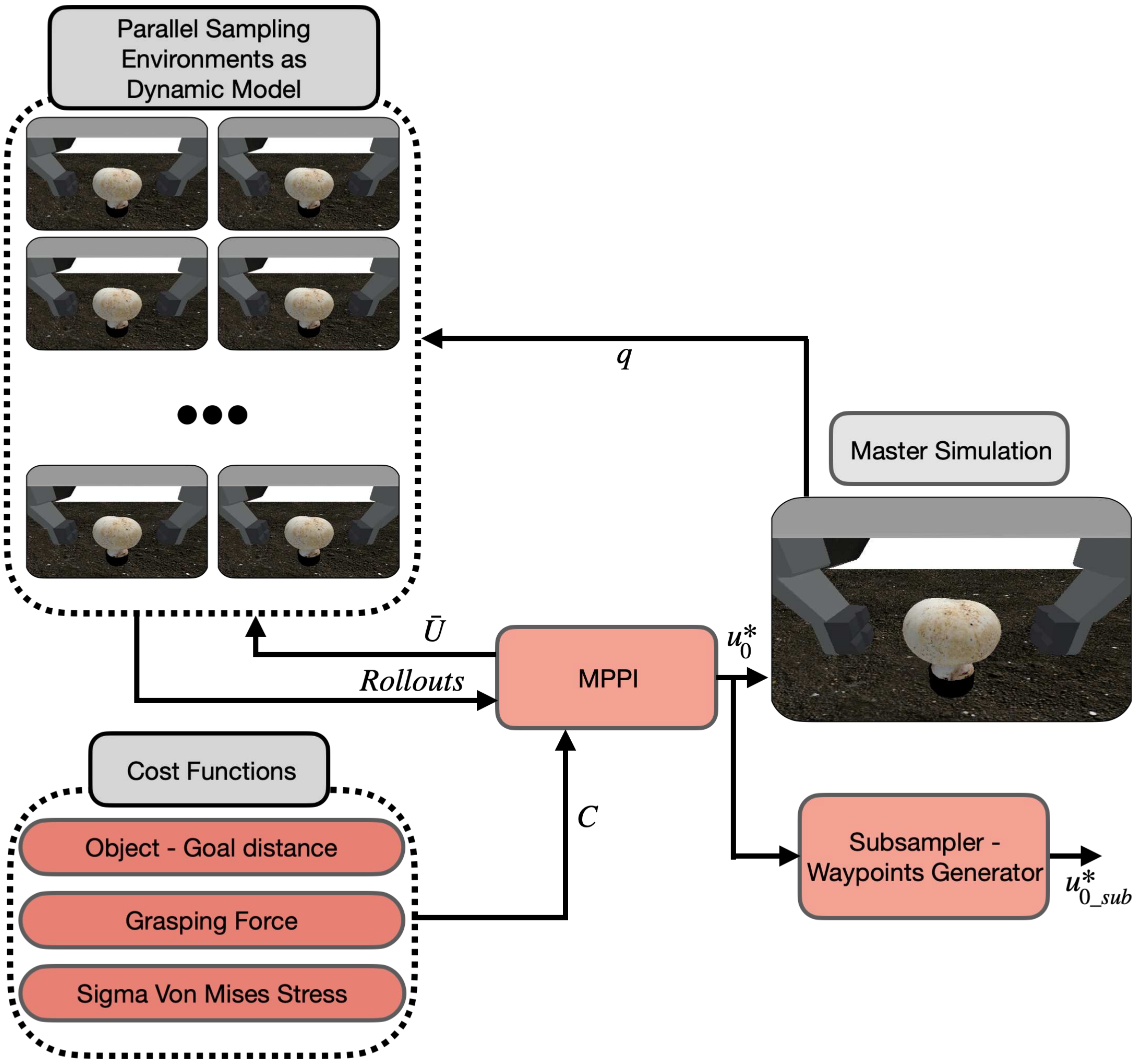


Figure 4.7 MPPI planner - system overview

rendering the system, in essence, an offline planner. This subsampling methodology has the added benefit of denoising the trajectory, which is a known issue with MPPI controllers. For an overview of the proposed implementation, see fig. 4.7.

To describe the mushroom picking task, we define a set of cost functions.

$$C_{\text{pick}} = C_{\text{dist}} + C_{\text{force}} + C_{\text{stress}}$$

$$C_{dist} = \omega_d \|p_{EE} - p_M\| + \omega_{M_p} \|p_G - p_M\|$$

The C_{dist} incentivizes the interaction between the gripper and the mushroom cap and describes its desired goal position, which in our case is just an offset by the z-axis.

$$C_{\text{force}} = \begin{cases} \omega_f \sum f_{fingers}, & \text{if } \sum f_{fingers} < f_{\text{safe}} \\ 0, & \text{otherwise} \end{cases}$$

C_{force} should penalize actions that are outside the safe limits in an attempt to avoid bruising and, consequently, destruction of the produce. In practice $f_{\text{safe}} = 3$ N.

$$C_{\text{stress}} = -\omega_\sigma \sigma_{VM}$$

The C_{stress} cost term incentivizes actions that will cause stress on the root and ultimately failure for the mushroom to be picked. We found this cost term to be necessary to allow the MPPI controller to discover the bending action, as it would otherwise be stuck to local minima.

For the current demonstrator, we have avoided adding a collision-cost term, as this would be necessary in a cluttered environment. Given the aforementioned compromise of a CPU parallelization instead of a GPU one, we found in practice that including collision detection between multiple mushroom instances was prohibitively expensive and impractical in terms of computation. This is within the scope of future work, which should include a GPU parallelizable physics engine that also allows for the removal of fixed constraints during runtime for material failure-mode emulation.

4.2.2 Experiments

We start by testing the efficacy of the simulation framework in combination with the MPPI planner with an AB test purely in simulation. In case A, we set the friction c_c of the mushroom cap at an abnormally high value $c_c = 1.0$ where the development of high shear

Table 4.1 Cost function weights

Cost function weights	**Values**
ω_d	8
ω_{M_p}	12
ω_f	0.005
ω_σ	0.0001

Table 4.2 MPPI parameters tuning

MPPI parameters	**Values**
Sampling Environments Number	60
Planning Horizon	20 steps
Sampling Method	Halton Spline
Sampling Noise σ	$[diag(0.1), 0.05]$
Rollout var discount λ	0.95

Table 4.3 PyBullet simulation parameters tuning

Simulation Parameters	**Values**
Controller sampling time	0.01 sec
Physics engine *dt*	0.001 sec
Solver iterations	10
Friction ERP	0.02

grasping forces should overcome the axial root resistance, allowing for a direct pulling motion within the desired constraints. We repeat the experiment in case In case B, under the same configurations, only this time setting the friction of the mushroom cap to a more

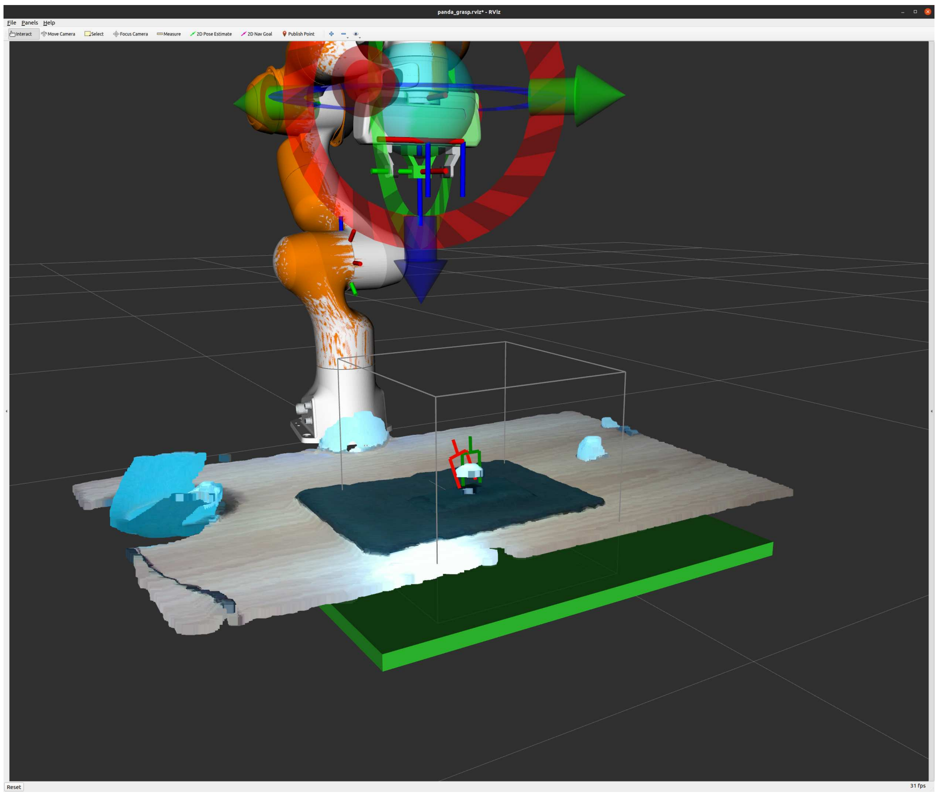


Figure 4.8 The key way-points poses generated from the MPPI planner describing the “bending motion” as optimal for mushroom picking.

realistic value $c_c = 0.3$, where a pure pulling motion would only be possible by exerting high lateral grasping forces, which would exceed the safe thresholds or would even be impossible to develop. As we see in fig. 4.9 our hypothesis is satisfied and the MPPI planner discovers autonomously the bending action to successfully detach the mushroom within the desired constraints. The total planning time with an Intel i5 13600K CPU and the configurations as described in Tables was in both cases 70 real world secs which corresponds to 1.4 ‘master’ simulation *secs*.

In the first row we observe the picking attempt of the MPPI planner when we set an abnormally high friction value at $c_c = 1.0$ at the mushroom cap. In that case, a direct pull motion is emerging because the necessary high shear forces can be easily generated. In the second row, we observe the result where we set a lower friction value $c_c = 0.3$ where the MPPI planner autonomously discovers the bending motion as the optimal picking strategy

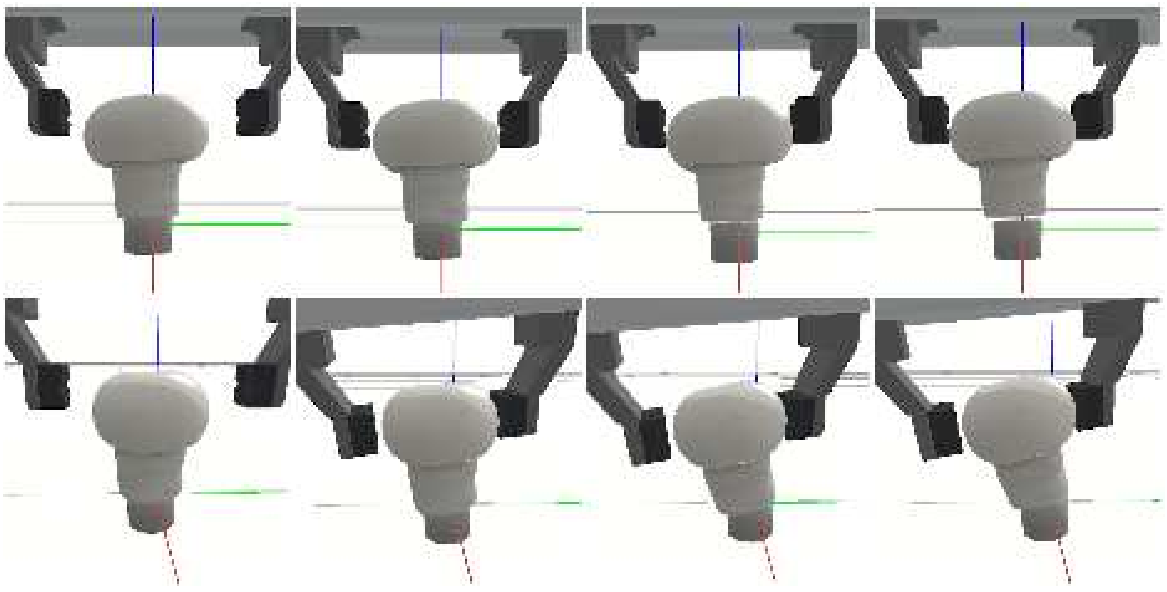


Figure 4.9 MPPI Mushroom harvesting AB cap friction testing resulting in alternative picking strategies by the MPPI planner.

We finally deploy the generated trajectory of the B test by first subsampling, in order to generate a set of key-waypoints for the gripper pose. We manually choose the gripper pose when contact of both fingers is initiated with the mushroom cap, and the pose at the precise moment when the mushroom is detached. We then feed the waypoints directly to the MoveIt! group commander for generating the final trajectory with RRT and handling the inverse kinematics. Automated extraction of an optimal set of waypoints to accurately describe the picking maneuver is within the scope of future work.

The bending action as transferred to the real-world setup with the Franka Robot and a 3D printed mushroom-like shape.

4.2.3 Future Directions

We have demonstrated a pipeline for mushroom harvesting with a robotic gripper, based on on-the-fly 3D scene reconstruction with a physical simulation and an offline MPPI planner for generating optimal uprooting manipulation maneuvers. We show that despite the lumped system dynamic model and the CPU parallelization limitations of the MPPI planner, the

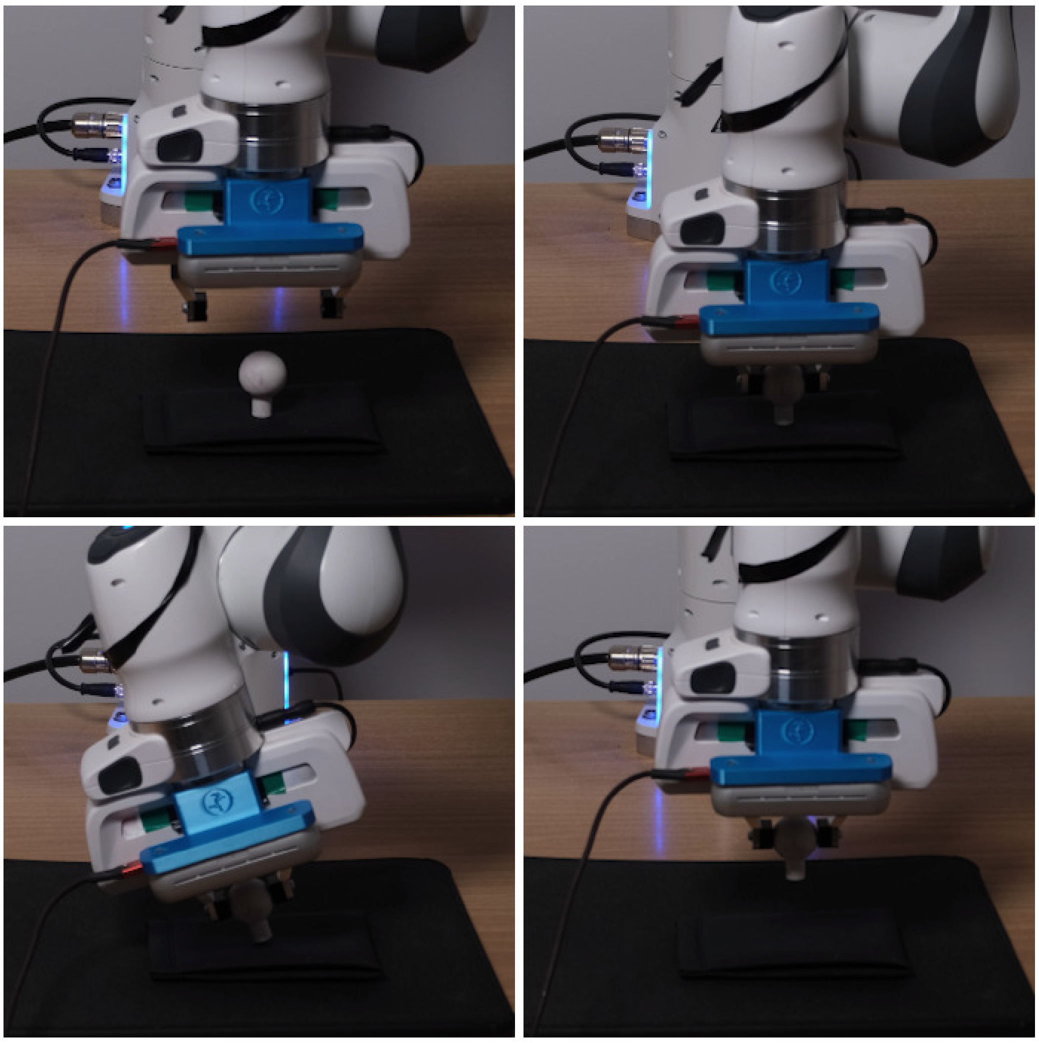


Figure 4.10 MPPI - mushroom harvesting real-world deployment.

system manages to discover optimal actions for mushroom picking. This is a sense-plan-act open-loop architecture; namely, it is not reactive to unpredictable events due to environmental changes and failures due to modeling discrepancies. Therefore, the next step is to close the loop, which means that the pass-through computation should be achieved in real time. We can do so through GPU parallelization [205] and an on-the-fly perception module for mushroom state estimation that includes pose and root stiffness. This would also allow for a significant increase in the total number of sampling environments and the planning horizon, resulting in more complex and robust grasping strategies.

As mentioned above, the MPPI-based planning strategy is prone to get stuck in local minima and additionally requires a significant engineering effort for the cost function shaping. Potential extensions to long-term planning could include MPC-like approaches [207, 208] or more analytical discrete/continuous TAMP frameworks such as PDDLStream for multi-stage forceful manipulation [209] or optimization over graphs of convex sets [210]. Finally, real-world analytical experiments in industrial settings with the incorporation of a softgripper into actual mushroom crops are necessary to assess the effectiveness of the method and any potential commercial viability.

4.3 Conclusions

4.3.1 MPPI-Based Planning in Broader Robotic Contexts

Model Predictive Path Integral (MPPI) control provides a unifying framework for addressing diverse robotics challenges, particularly in dynamic and uncertain environments. Its sampling-based nature enables robustness to both parametric and structural uncertainties—a critical requirement for bridging the simulation-to-reality (sim2real) gap. By leveraging Monte Carlo trajectory sampling and parallelized cost evaluation, MPPI inherently accommodates nonlinear dynamics, non-convex constraints, and stochastic disturbances, making it applicable beyond robotic arm manipulation to tasks such as autonomous navigation, multi-agent coordination, and contact-rich object interaction [211–213].

In the context of domain randomization (DR), MPPI's adaptability aligns with the need for policies robust to variations in dynamics, sensor noise, and environmental geometry. For example, space manipulators for debris removal [213] and autonomous vehicles navigating cluttered urban environments [212] rely on MPPI's ability to iteratively refine on-board models using real-time parameter estimation. Furthermore, MPPI's capacity to handle discontinuous cost functions—such as collision penalties and task-specific rewards—enables seamless integration with reinforcement learning pipelines that employ DR for policy transfer [211].

MPPI's utility can extend to:

- **Multi-robot systems**: Distributed collision avoidance through coupled cost functions [211]
- **Legged locomotion**: Terrain adaptation via online inertia matrix estimation [211]
- **Human-robot collaboration**: Safe interaction through stochastic impedance tuning [214, 215]

4.3.2 Potential Alternatives to MPPI for Sim2Real

While MPPI excels in handling nonlinear dynamics and parametric uncertainties, alternative planning frameworks offer complementary strengths for specific robotic applications. Below, we present three prominent approaches as an alternative to MPPI based planning:

Table 4.4 Predominant alternatives to MPPI planners for Sim2Real

Method	**Key Features**	**Sim2real Relevance**	**Limitations**
D* Lite	Incremental replanning for dynamic environments [216]	Efficient adaptation to real-world map changes (e.g., movable obstacles)	Limited to geometric constraints; struggles with high-dimensional action spaces
APP (Alternative Paths Planner)	Precomputed path sets for semi-structured environments [217]	Fixed-time guarantees enhance predictability for repetitive industrial tasks	Requires offline pre-processing; less adaptable to novel perturbations

Method	Key Features	Sim2real Relevance	Limitations
RU-PRM	Reusable roadmap components for similar obstacle configurations [218]	Accelerates planning in partially randomized environments via memory reuse	Performance degrades with extreme domain shifts in DR training

4.3.2.1 Why MPPI Remains Central

MPPI's sampling-based formulation provides inherent advantages for sim2real transfer that discrete or precomputed methods lack:

1. **Unified handling of parametric and structural uncertainties** through parallelized Monte Carlo rollouts [213]
2. **Seamless integration with DR-trained neural policies** via differentiable cost formulations [211]
3. **Real-time adaptation** to residual reality gaps through online covariance tuning [211]

While the alternatives excel in structured subproblems, none match MPPI's generalizability across manipulation, navigation, and human-robot interaction domains under randomized training regimes [211].

4.3.3 Scalability and Industrial Viability

MPPI's sampling-based architecture exhibits strong scalability for industrial applications due to its parallelizable computation and adaptability to high-dimensional systems. Through GPU acceleration, MPPI achieves real-time performance even in complex scenarios like multi-robot coordination (48–144 state dimensions) [219] and 7-DoF manipulator control [206], making it viable for large-scale manufacturing lines requiring synchronized robotic cells. Its derivative-free optimization bypasses gradient calculations, enabling efficient scaling across tasks with nonlinear dynamics, such as contact-rich assembly or precision grasping under variable friction [220, 221]. Commercial adoption is further bolstered by MPPI's compatibility with modular safety frameworks like Control Barrier Functions (CBFs) [222, 223], which enforce collision avoidance and joint limit constraints without compromising planning speed—critical for ISO-certified industrial environments. For instance, MPPI-based systems have demonstrated 125Hz control rates in joint-space trajectory optimization [206].

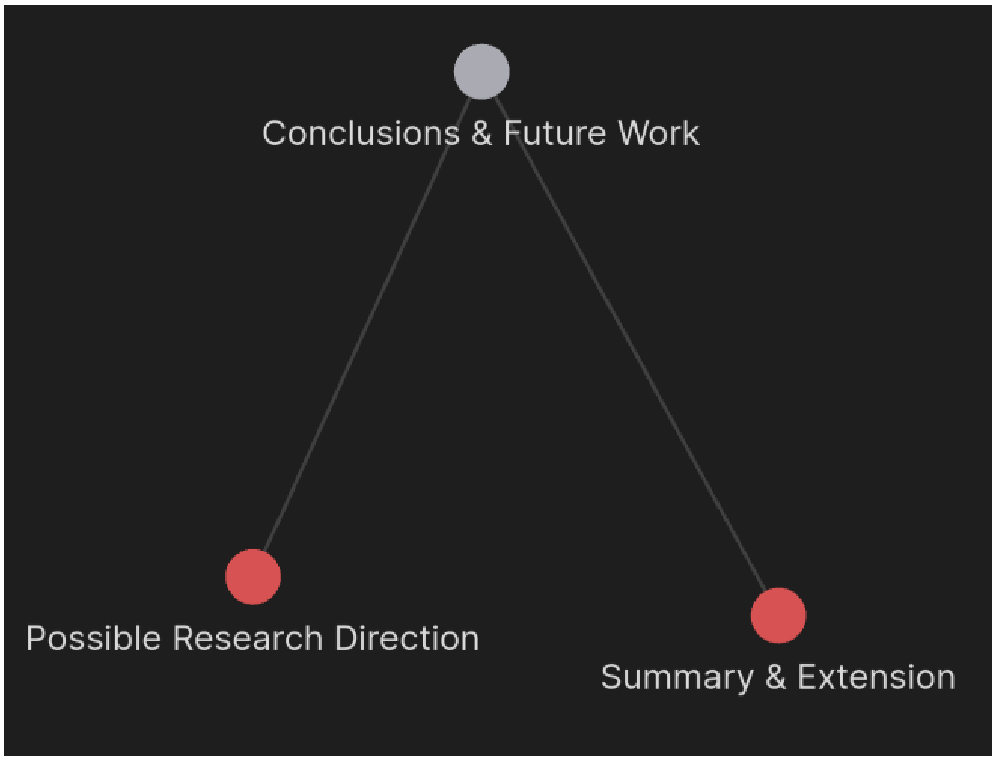
While computational demands historically limited deployment, advances in edge-computing hardware and distributed sampling [224] now enable cost-effective integration into collaborative robots (cobots) and autonomous guided vehicles (AGVs). Hybrid architectures combining MPPI with learned cost functions [206] further enhance commercial appeal by reducing manual tuning for tasks like PCB soldering or battery module handling, where domain randomization prepares controllers for component tolerances. These attributes position MPPI as a scalable, safety-certifiable solution for Industry 4.0 applications requiring both precision and rapid reconfigurability [225].

Chapter 5

Conclusions & Future Work

”The real voyage of discovery consists not in seeking new landscapes, but in having new eyes.”

—Marcel Proust, *In Search of Lost Time*



5.1 Summary & Extension

This work has studied the current state-of-the-art on the Simulation to Reality Gap for Robotics through an extensive literature review in an attempt to provide a set of concrete definitions, a taxonomy on sim2real techniques and methodologies and simulation tools, as well as the correlation to adjacent scientific principles and fields. This broad and holistic methodological approach, reveals that Sim2Real cannot be simply reduced to isolated, ad-hoc techniques applied to an already predefined system architecture, but on the contrary it impacts all aspects of a visuomotor policy design process—from the observation space and the representation of the perception module, the controller-agent architecture and training methodology, to the action space design, all of which are task conditioned and are affecting a sim2real transfer as we establish in Chapter 2. Based on the current research trends, usually sim2real seems to be marginally treated in the context of the learning and control disciplines. Our literature review and subsequent implementation offer valuable insights and can serve as a reference for the design of visuomotor policies.

A future extension of the literature review should include an exhaustive coverage of the model-based Reinforcement Learning and it's interception with the Model Predictive Control(MPC) methodologies with respect to life-long learning and adaptive, ad-hoc, techniques respectively. Based on the newly gained insights on sim2real, a manual design of simulation frameworks is a significant bottleneck in the typically imposed sim2real workflow and it can therefore not be a viable scalable way forward. It is crucial that the world model is actively built by the agent itself as it goes, in an active inference [226] way, where the agent utilizes the prior information and an inherit curiosity in order to further interact with the world in order to autonomously discover it's underlying ontologies and semantics. Lastly with regards to the literature survey, the rapid advancements and mass adoption of Large Language Models (LLMs) and Vision Language Models should be also examined with respect to implications on sim2real for robotics.

Our subsequent implementation for designing such a visuomotor policy for the hard automation problem of mushroom harvesting with a robotic gripper allowed us to analytically evaluate a set of key sim2real techniques, the impact of the overall agent architecture design parameters and methodologies and most importantly has revealed the key weak points of a "conventional" sim2real pipeline. More concretely by designing a fist order approximation model, as a surrogate model for our simulation framework, which involves approximations for the material elastic deformations and failure modes we were able to answer whether and how such approximations, often-times crude, can be useful for effectively solving the task in a sim2real context. We've shown that such 1^{st} order modeling approximations for elastic deformations and failure modes are possible in the context of a traditional rigid-body physics engine in order to provide the necessary simulation runtime speed, which is necessary in a RL training context as well as an MPC context.

Designing such a simulation framework for mushroom harvesting has proven to far more challenging than initially anticipated, particularly in terms of development effort and real-world data collection for fine-tuning in order to match the real-world dynamics through system identification. **Given the amount of development effort required to create a simulation framework with adequate predictive accuracy and runtime speed, using it solely for training seemed inefficient and underutilized**. This led us to an alternative approach beyond Reinforcement Learning, aiming to directly harness the predictive power of our model as core component of the prediction and planning modules. We adopted a sampling based MPC methodology, specifically the Model Predictive Path Integral (MPPI) control framework as it is gradient-free and therefore is able to handle the inherit hybrid dynamics nature of the robotic manipulation. The consequent transfer from simulation to the real-world comes as a result of our modeling methodologies, the system identification procedures and an architecture that renders the simulation a perception and planning module by effectively transferring in the end a task plan description which can be handled by conventional planners and operational controllers into final torques to the joints. This showcases the power of explicit transferable abstractions which allow even for relatively crude models to generate accurate task descriptions involving complex dynamics, which is our main key insight and contribution.

A first obvious potential future extension on this existing research work should include a close-loop architecture which would necessitate faster simulation inference time. This could be achieved by relaxing the stiffness conditions of our problem in combination with a more efficient source code base with potentially an alternative physics engine. This would render the pipeline reactive to potential world scene alterations during execution.

Another important note is that the sim2real transfer could be compromised occasionally due to simulation optimization bias (SOB) as the planner tends to exploit simulation imperfections generating infeasible and non-realistic plans. Extending the current framework with higher level abstraction symbolic planners [227] could offer a viable solution for long-horizon planning. Xue et al. [44] explore the parallels between sampling based MPC and a Diffusion-Style Annealing as an attempt to robustify the more vanilla MPPI, which could also be an excellent research direction for our architecture. Domain Randomization is another interesting aspect with respect to a MPPI style planner where the parallel simulations are varying in key parameters which should potentially increase the overall robustness.

Sensitivity to local minima and the laborious nature of fine-tuning the cost functions for each task are also significant drawbacks of this approach. Generative methodologies, leveraging the power of recent Large Language Models and Vision Language models, potentially in combination with human-demonstrations, could automatically give rise to complex cost functions as task descriptors [2, 228] with the additional benefit of increasing the visual perception characteristics.

In the context of bridging the simulation-to-reality gap in robotics, the interpretability of decisions made by robotic systems using Model Predictive Path Integral (MPPI) control requires also some rigorous attention, given the inherently stochastic nature of the method. While MPPI's stochastic nature can make it less immediately interpretable than traditional deterministic methods, strategies such as **visualizing sampled trajectories**, **decomposing cost functions**, and **analyzing importance sampling** can enhance human understanding of the system's decision-making process. However, **a notable trade-off exists between real-time performance and explainability**, particularly in time-critical robotic manipulation tasks. Enhancing explainability often involves additional computational steps that can impact real-time performance, necessitating careful balancing through techniques like asynchronous explainability analysis, adaptive sampling, and hierarchical interpretation systems.

Drawing parallels between physics modeling and mechanistic interpretability offers further insights into this challenge. **Physics models, with their well-defined coordinates and parameters, can be viewed as monosemantic interpretations, providing clear physical meanings and consistent interpretations.** This approach aligns with the goals of mechanistic interpretability in machine learning, where both seek to decompose complex systems into interpretable components and establish causal relationships. However, the complexity and non-linearity of real-world robotic systems often surpass simple physics models, leading to emergent behaviors that challenge straightforward interpretations. Recent approaches, such as Physics-Informed Neural Networks (PINNs) and hybrid modeling, aim to bridge this gap by combining the interpretability of physics models with the flexibility of machine learning. These advancements point towards a future where robotic systems can achieve both high performance and interpretability, crucial for their safe and effective deployment in real-world scenarios.

Lastly extensive testing and experimentation in real-world mushroom fields is necessary in order to evaluate all these potential directions and assess their commercialization potential.

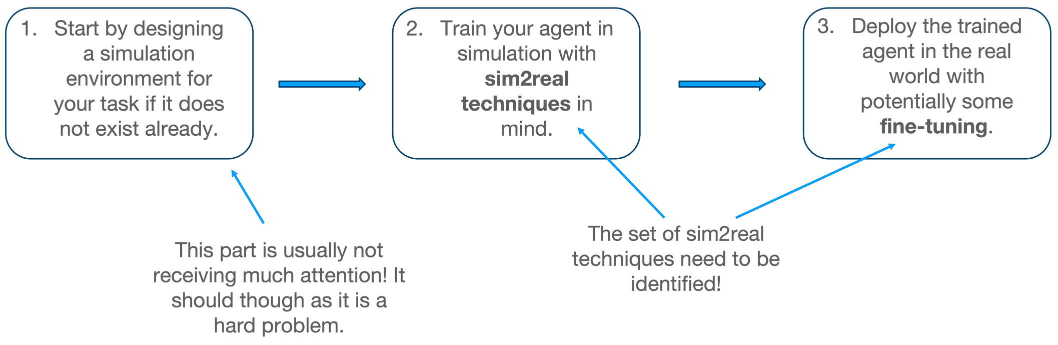

1. Start by designing a simulation environment for your task if it does not exist already.
2. Train your agent in simulation with **sim2real techniques** in mind.
3. Deploy the trained agent in the real world with potentially some **fine-tuning**.
This part is usually not receiving much attention! It should though as it is a hard problem.
The set of sim2real techniques need to be identified!

Figure 5.1 The cost of developing a simulation framework for real-world robotic application is most often underestimated.

5.2 Possible Research Direction

At the core of our research thematic lies the development of a simulation framework for a hard real-world automation problem. We firmly believe that simulation as a solely training artifact is a heavy under-utilization of a valuable asset with multiple underscored benefits. Simulation should be seen as a human made "real-world embedding" based on spatial and temporal transients of human scale, namely Newton-Euler dynamics (at least in case of rigid body dynamics). This embedding space as perception module offers an explainable scene reconstruction with obvious task planning possibilities. Even in the context learning based methodologies and the "model-based" vs "model-free" (pseudo) dilemma, data efficiency can achieved basically given a world model. A world model of course comes at a cost, either development cost or training cost with the additional complexity potentially jeopardizing convergence.

The future research direction should focus heavily in the active inference [226] framework, where the agent should purposefully act in order to gain more information about the world and update its current model. The robot agent should be seen as an "active world explorer" and not purely the end deployment goal of static activation weights.

The concept of active inference aligns well also with addressing the issues of explainability and interpretability as the agent should be able to embed autonomously and spontaneously the world model in a monosemantic base [229] that is also mechanistically interpretable [230, 231] as an optimal encoding and translation for the action space as conditioned for the task completion.

An interesting research direction could therefore be an attempt for a “dynamic embedding space”, namely a learned dynamic world model as a perception module, namely 3D scene reconstruction with physics embedding. The ability to predict the future world state should drive the exploration and a consequent update of the world model [232].

Bibliography

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. Comment: 15 pages, 5 figures.

[2] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-Level Reward Design via Coding Large Language Models. Comment: ICLR 2024. Project website and open-source code: https://eureka-research.github.io/.

[3] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control.

[4] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. GenSim: Generating Robotic Simulation Tasks via Large Language Models. Comment: See our project website (https://liruiw.github.io/gensim), demo (https://huggingface.co/spaces/Gen-Sim/Gen-Sim), and code (https://github.com/liruiw/GenSim) for visualizations and open-source models and datasets.

[5] Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Abhiram, and useprefix=true family=al., prefix=et. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. Comment: Project website: https://robotics-transformer-x.github.io.

[6] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. Comment: Website: https://openvla.github.io/.

[7] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with Language Model is Planning with World Model.

[8] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. Comment: Project website at https://huangwl18.github.io/language-planner.

[9] Leonidas Droukas, Zoe Doulgeri, Nikolaos L. Tsakiridis, Dimitra Triantafyllou, Ioannis Kleitsiotis, Ioannis Mariolis, Dimitrios Giakoumis, Dimitrios Tzovaras, Dimitrios Kateris, and Dionysis Bochtis. A Survey of Robotic Harvesting Systems and Enabling Technologies. 107(2):21.

[10] Luiz F. P. Oliveira, António P. Moreira, and Manuel F. Silva. Advances in Agriculture Robotics: A State-of-the-Art Review and Challenges Ahead. 10(2):52.

[11] Softgrip Project.

[12] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. 521(7553):436–444.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. 518(7540):529–533.

[15] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real Single-Image Flight without a Single Real Image. Comment: To appear at Robotics: Science and Systems Conference (R:SS), 2017. Supplementary video: https://www.youtube.com/watch?v=nXBWmzFrj5s.

[16] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand.

[17] Konstantinos Dimitropoulos, Ioannis Hatzilygeroudis, and Konstantinos Chatzilygeroudis. A Brief Survey of Sim2Real Methods for Robot Learning. pages 133–140.

[18] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. 9:153171–153187.

[19] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Florian Golemo, Melissa Mozifian, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, C. Karen Liu, Jan Peters, Shuran Song, Peter Welinder, and Martha White. Perspectives on Sim2Real Transfer for Robotics: A Summary of the R:SS 2020 Workshop. Comment: Summary of the "2nd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics" held in conjunction with "Robotics: Science and System 2020". Website: https://sim2real.github.io/.

[20] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. Comment: Accepted to the 2020 IEEE Symposium Series on Computational Intelligence.

[21] Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot Learning from Randomized Simulations: A Review. Comment: submitted to Frontiers in Robotics and AI.

[22] Liam Paull and Anthony Courchesne. On Assessing the Value of Simulation for Robotics.

[23] Lucy Xiaoyang Shi, Archit Sharma, Tony Z. Zhao, and Chelsea Finn. Waypoint-Based Imitation Learning for Robotic Manipulation. Comment: The first two authors contributed equally.

[24] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. Comment: ICML 2017. Code at https://github.com/cbfinn/maml, Videos of RL results at https://sites.google.com/view/maml, Blog post at http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/.

[25] Fabio Muratore and Michael Gienger. Robot learning from randomized simulations: A review. 9:799893.

[26] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J. Pal, and Liam Paull. Active Domain Randomization. Comment: Code available at https://github.com/montrealrobotics/active-domainrand.

[27] John Smith and Jane Doe. Continual domain randomization.

[28] Gabriele Tiboni, Andrea Protopapa, Tatiana Tommasi, and Giuseppe Averta. Domain Randomization for Robust, Affordable and Effective Closed-loop Control of Soft Robots. Comment: Presented as conference paper at IEEE/RSJ IROS 2023, Detroit, USA. Project website at https://andreaprotopapa.github.io/dr-soro/.

[29] Fabio Ramos, Rafael Possas, and Dieter Fox. BayesSim: Adaptive Domain Randomization Via Probabilistic Inference for Robotics Simulators. In *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation.

[30] Theo Jaunet, Guillaume Bono, Romain Vuillemot, and Christian Wolf. SIM2REALVIZ: Visualizing the Sim2Real Gap in Robot Ego-Pose Estimation.

[31] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, C. Karen Liu, Jan Peters, Shuran Song, Peter Welinder, and Martha White. Sim2Real in Robotics and Automation: Applications and Challenges. 18(2):398–400.

[32] Fabio Muratore, Michael Gienger, and Jan Peters. Assessing Transferability from Simulation to Reality for Reinforcement Learning. 43(4):1172–1183.

[33] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? 5(4):6670–6677.

[34] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience.

[35] Elie Aljalbout, Felix Frank, Maximilian Karl, and prefix=van der useprefix=true family=Smagt, given=Patrick. On the Role of the Action Space in Robot Manipulation Learning and Sim-to-Real Transfer.

[36] Jack Collins, Ross Brown, Jurgen Leitner, and David Howard. Traversing the Reality Gap via Simulator Tuning. Comment: 8 Pages, Submitted to IROS2020.

[37] David Blanco-Mulero, Oriol Barbany, Gokhan Alcan, Adrià Colomé, Carme Torras, and Ville Kyrki. Benchmarking the Sim-to-Real Gap in Cloth Manipulation. Comment: Accepted to IEEE Robotics and Automation Letters (RA-L). 8 pages, 6 figures. Supplementary material available at https://sites.google.com/view/cloth-sim2real-benchmark.

[38] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810.

[39] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning Dexterous In-Hand Manipulation. Comment: Making OpenAI the first author. We wish this paper to be cited as "Learning Dexterous In-Hand Manipulation" by OpenAI et al. We are replicating the approach from the physics community: arXiv:1812.06489.

[40] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion.

[41] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual Dexterity: In-hand Dexterous Manipulation from Depth.

[42] Zhao-Heng Yin, Binghao Huang, Yuzhe Qin, Qifeng Chen, and Xiaolong Wang. Rotating without Seeing: Towards In-hand Dexterity through Touch. Comment: Project page: https://touchdexterity.github.io.

[43] Luke Beddow, Helge Wurdemann, and Dimitrios Kanoulas. Reinforcement Learning Grasping With Force Feedback From Modeling of Compliant Fingers. pages 1–12.

[44] Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-Order Sampling-Based MPC for Torque-Level Locomotion Control via Diffusion-Style Annealing. Comment: 9 pages, 9 figures, submitted to ICRA2025.

[45] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to Rewards for Robotic Skill Synthesis. Comment: https://language-to-reward.github.io/.

[46] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H. Huang, Dhruva Tirumala, Markus Wulfmeier, Jan Humplik, Saran Tunyasuvunakool, Noah Y. Siegel, Roland Hafner, Michael Bloesch, Kristian Hartikainen, Arunkumar Byravan, Leonard Hasenclever,

Yuval Tassa, Fereshteh Sadeghi, Nathan Batchelor, Federico Casarini, Stefano Saliceti, Charles Game, Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber, Nicole Hurley, Francesco Nori, Raia Hadsell, and Nicolas Heess. Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning. Comment: Project website: https://sites.google.com/view/op3-soccer.

[47] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In Federico Morán, Alvaro Moreno, Juan Julián Merelo, and Pablo Chacón, editors, *Advances in Artificial Life*, pages 704–720. Springer.

[48] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. Comment: 8 pages, 7 figures. Submitted to 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017).

[49] Jonatan S. Dyrstad and John Reidar Mathiassen. Grasping virtual fish: A step towards robotic deep learning from demonstration in virtual reality. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1181–1187.

[50] Michel Breyer, Jen Jen Chung, Lionel Ott, Roland Siegwart, and Juan Nieto. Volumetric Grasping Network: Real-time 6 DOF Grasp Detection in Clutter. Comment: Conference on Robot Learning (CoRL), 2020.

[51] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. GenSim: Generating Robotic Simulation Tasks via Large Language Models. Comment: See our project website (https://liruiw.github.io/gensim), demo (https://huggingface.co/spaces/Gen-Sim/Gen-Sim), and code (https://github.com/liruiw/GenSim) for visualizations and open-source models and datasets.

[52] Yao Zhao, Tao Wu, Yijie Zhu, Xiang Lu, Jun Wang, Haitham Bou-Ammar, Xinyu Zhang, and Peng Du. ZSL-RPPO: Zero-Shot Learning for Quadrupedal Locomotion in Challenging Terrains using Recurrent Proximal Policy Optimization.

[53] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric Actor Critic for Image-Based Robot Learning. Comment: Videos of experiments can be found at http://www.goo.gl/b57WTs.

[54] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-Real Reinforcement Learning for Deformable Object Manipulation.

[55] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. Comment: Project website: https://sites.google.com/view/alternating-pick-and-place.

[56] Helei Duan, Bikram Pandit, Mohitvishnu S. Gadde, prefix=van useprefix=true family=Marum, given=Bart Jaap, Jeremy Dao, Chanho Kim, and Alan Fern. Learning Vision-Based Bipedal Locomotion for Challenging Terrain.

[57] Raghad Alghonaim and Edward Johns. Benchmarking Domain Randomisation for Visual Sim-to-Real Transfer. Comment: Published at ICRA 2021. For project page, please visit: https://www.robot-learning.uk/benchmarking-domain-randomisation.

[58] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation.

[59] Zihan Ding, Ya-Yen Tsai, Wang Wei Lee, and Bidan Huang. Sim-to-Real Transfer for Robotic Manipulation with Tactile Sensory. Comment: Accepted for publication at IROS 2021.

[60] Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. VMP: Versatile Motion Priors for Robustly Tracking Motion on Physical Characters.

[61] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. 4(26):eaau5872.

[62] Chenhao Li, Marin Vlastelica, Sebastian Blaes, Jonas Frey, Felix Grimminger, and Georg Martius. Learning Agile Skills via Adversarial Imitation of Rough Partial Demonstrations.

[63] Gabriele Tiboni, Pascal Klink, Jan Peters, Tatiana Tommasi, Carlo D'Eramo, and Georgia Chalvatzaki. Domain Randomization via Entropy Maximization. Comment: Published as a conference paper at ICLR 2024. Project website at https://gabrieletiboni.github.io/doraemon/.

[64] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein Variational Policy Gradient.

[65] Lerrel Pinto, James Davidson, and Abhinav Gupta. Supervision via Competition: Robot Adversaries for Learning Tasks. Comment: Submission to ICRA 2017.

[66] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust Adversarial Reinforcement Learning. Comment: 10 pages.

[67] Julius Hietala, David Blanco-Mulero, Gokhan Alcan, and Ville Kyrki. Closing the Sim2Real Gap in Dynamic Cloth Manipulation. Comment: 8 pages, 8 figures. This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

[68] Zhaoming Xie, Xingye Da, prefix=van de useprefix=true family=Panne, given=Michiel, Buck Babich, and Animesh Garg. Dynamics Randomization Revisited:A Case Study for Quadrupedal Locomotion.

[69] Manuel Kaspar, Juan D. Muñoz Osorio, and Juergen Bock. Sim2Real Transfer for Reinforcement Learning without Dynamics Randomization. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4383–4388.

[70] Jeremy Dao, Helei Duan, Kevin Green, Jonathan Hurst, and Alan Fern. Learning to Walk without Dynamics Randomization. page 3.

[71] Fabio Muratore. Data-efficient domain randomization with bayesian optimization. ICRA 2021 Presentation.

[72] Xiaoyu Chen, Jiachen Hu, Chi Jin, Lihong Li, and Liwei Wang. UNDERSTANDING DOMAIN RANDOMIZATION FOR SIM-TO-REAL TRANSFER.

[73] Cheongwoong Kang, Wonjoon Chang, and Jaesik Choi. Balanced Domain Randomization for Safe Reinforcement Learning. 14(21):9710.

[74] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain Separation Networks. Comment: This work will be presented at NIPS 2016.

[75] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. Comment: 9 pages, 5 figures, 3 tables.

[76] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. Comment: Final CVPR 2017 paper and supplementary material.

[77] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. Comment: Published in JMLR: http://jmlr.org/papers/v17/15-239.html.

[78] Youngbin Park, Sang Hyoung Lee, and Il Hong Suh. Sim-to-Real Visual Grasping via State Representation Learning Based on Combining Pixel-Level and Feature-Level Domain Adaptation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6300–6307. IEEE.

[79] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks. Comment: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019).

[80] Homanga Bharadhwaj, Zihan Wang, Yoshua Bengio, and Liam Paull. A Data-Efficient Framework for Training and Sim-to-Real Transfer of Navigation Policies. Comment: Under review in ICRA 2019.

[81] Fangyi Zhang, Jürgen Leitner, Zongyuan Ge, Michael Milford, and Peter Corke. Adversarial Discriminative Sim-to-real Transfer of Visuo-motor Policies. Comment: Under review for the International Journal of Robotics Research.

[82] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. RL-CycleGAN: Reinforcement Learning Aware Simulation-To-Real. Comment: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2020).

[83] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots. Comment: RSS 2021. Webpage at https://ashish-kmr.github.io/rma-legged-robots/.

[84] Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the Unknown: Learning a Universal Policy with Online System Identification. Comment: Accepted as a conference paper at RSS 2017.

[85] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-Hand Object Rotation via Rapid Motor Adaptation. Comment: CoRL 2022. Code and Website: https://haozhi.io/hora.

[86] Andrei A. Rusu, Mel Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-Real Robot Learning from Pixels with Progressive Nets.

[87] Yunchu Zhang, Liyiming Ke, Abhay Deshpande, Abhishek Gupta, and Siddhartha Srinivasa. Cherry-Picking with Reinforcement Learning.

[88] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). 11(1):1–18.

[89] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-Tuned Sim-to-Real Transfer. Comment: ICRA 2021. First two authors contributed equally. Website at https://yuqingd.github.io/autotuned-sim2real/.

[90] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through Simulation: Integrating Simulated and Real Data into Deep Reinforcement Learning for Vision-Based Autonomous Flight. Comment: First three authors contributed equally. Accepted to ICRA 2019.

[91] Rafael Possas, Lucas Barcelos, Rafael Oliveira, Dieter Fox, and Fabio Ramos. Online BayesSim for Combined Simulator Parameter Inference and Policy Improvement. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5445–5452. IEEE.

[92] Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone. Humanoid Robots Learning to Walk Faster: From the Real World to Simulation and Back.

[93] Josiah P. Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 4931–4932. AAAI Press.

[94] Siddharth Desai, Haresh Karnan, Josiah P. Hanna, Garrett Warnell, and Peter Stone. Stochastic Grounded Action Transformation for Robot Learning in Simulation. Comment: Accepted at 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020.

[95] Markus Wulfmeier, Ingmar Posner, and Pieter Abbeel. Mutual Alignment Transfer Learning.

[96] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model.

[97] Mohammadhossein Malmir, Josip Josifovski, Noah Klarmann, and Alois Knoll. Robust Sim2Real Transfer by Learning Inverse Dynamics of Simulated Systems.

[98] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Dmitry Olefir, Tomas Hodan, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Teja Katam, and Ahsan Lodhi. BlenderProc: Reducing the Reality Gap with Photorealistic Rendering.

[99] Joanne Truong, Max Rudolph, Naoki Yokoyama, Sonia Chernova, Dhruv Batra, and Akshara Rai. Rethinking Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation.

[100] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and Interpretable Skill Acquisition in Multi-task Reinforcement Learning. Comment: 14 pages, 6 figures.

[101] Ryan Julian, Eric Heiden, Zhanpeng He, Hejia Zhang, Stefan Schaal, Joseph Lim, Gaurav Sukhatme, and Karol Hausman. *Scaling Simulation-to-Real Transfer by Learning Composable Robot Skills.*

[102] Yuanpei Chen, Chen Wang, Li Fei-Fei, and C. Karen Liu. Sequential Dexterity: Chaining Dexterous Policies for Long-Horizon Manipulation. Comment: CoRL 2023.

[103] Kai Lu, Bo Yang, Bing Wang, and Andrew Markham. Decoupling Skill Learning from Robotic Control for Generalizable Object Manipulation. Comment: Accepted to IEEE International Conference on Robotics and Automation (ICRA) 2023.

[104] Mengyuan Yan, Iuri Frosio, Stephen Tyree, and Jan Kautz. Sim-to-Real Transfer of Accurate Grasping with Eye-In-Hand Observations and Continuous Control. Comment: Neural Information Processing Systems (NIPS) 2017 Workshop on Acting and Interacting in the Real World: Challenges in Robot Learning.

[105] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual Dexterity: In-hand Dexterous Manipulation from Depth.

[106] Kefang Zhang, Jiatao Lin, Lv Bi, and Tan Zhang. Sim2Real Learning of Vision-Based Obstacle Avoidance for Robotic Manipulators.

[107] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep Drone Acrobatics. Comment: 8 pages + 2 pages references. Video: https://youtu.be/2N_wKXQ6MXA. Code: https://github.com/uzh-rpg/deep_drone_acrobatics.

[108] Wenxuan Zhou, Bowen Jiang, Fan Yang, Chris Paxton, and David Held. Learning Hybrid Actor-Critic Maps for 6D Non-Prehensile Manipulation.

[109] Panagiotis Petropoulakis, Ludwig Gräf, Mohammadhossein Malmir, Josip Josifovski, and Alois Knoll. State Representations as Incentives for Reinforcement Learning Agents: A Sim2Real Analysis on Robotic Grasping. Comment: Accepted to IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2024.

[110] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. 5(47):eabc5986.

[111] Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General In-Hand Object Rotation with Vision and Touch. Comment: CoRL 2023; Website: https://haozhi.io/rotateit/.

[112] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, prefix=de useprefix=true family=Freitas, given=Nando, and Nicolas Heess. Reinforcement and Imitation Learning for Diverse Visuomotor Skills. Comment: 13 pages, 6 figures, Published in RSS 2018.

[113] Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. TRANSIC: Sim-to-Real Policy Transfer by Learning from Online Correction. Comment: Project website: https://transic-robot.github.io/.

[114] Rich Caruana. Multitask Learning. 28(1):41–75.

[115] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning. Comment: First 2 authors contributed equally. Website: https://sites.google.com/berkeley.edu/metaadaptivecontrol.

[116] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and prefix=de useprefix=true family=Freitas, given=Nando. A Generalist Agent. Comment: Published at TMLR, 42 pages.

[117] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. RoboCat: A Self-Improving Foundation Agent for Robotic Manipulation.

[118] Josip Josifovski, Mohammadhossein Malmir, Noah Klarmann, and Alois Knoll. Continual Learning on Incremental Simulations for Real-World Robotic Manipulation Tasks.

[119] Ken Kansky, Tom Silver, David A. Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema Networks: Zero-shot Transfer with a Generative Causal Model of Intuitive Physics.

[120] Eugene Valassakis, Zihan Ding, and Edward Johns. Crossing The Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics. Comment: To be published at IROS 2020. 8 pages, 6 figures. For supplementary material and code, please visit : https://www.robot-learning.uk/crossing-the-gap.

[121] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-Contrastive Networks: Self-Supervised Learning from Video.

[122] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning. Comment: First two authors contributed equally. Video available at https://sites.google.com/view/daml.

[123] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from Human Videos as a Versatile Representation for Robotics. Comment: Accepted at CVPR 2023. Website at https://robo-affordances.github.io/.

[124] Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns. DALL-E-Bot: Introducing Web-Scale Diffusion Models to Robotics. 8(7):3956–3963.

[125] Chao Zhao, Chunli Jiang, Junhao Cai, Michael Yu Wang, Hongyu Yu, and Qifeng Chen. Flipbot: Learning Continuous Paper Flipping via Coarse-to-Fine Exteroceptive-Proprioceptive Exploration. Comment: Accepted to International Conference on Robotics and Automation (ICRA) 2023.

[126] Open Dynamics Engine.

[127] Bullet Real-Time Physics Simulation | Home of Bullet and PyBullet: Physics simulation for games, visual effects, robotics and reinforcement learning.

[128] Fabio Muratore, Felix Treede, Michael Gienger, and Jan Peters. Domain Randomization for Simulation-Based Policy Optimization with Transferability Assessment. In *Proceedings of The 2nd Conference on Robot Learning*, pages 700–713. PMLR.

[129] Serena Ivaldi, Jan Peters, Vincent Padois, and Francesco Nori. Tools for simulating humanoid robot dynamics: A survey based on user feedback. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*.

[130] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4397–4404.

[131] Jack Collins, David Howard, and Jürgen Leitner. Quantifying the Reality Gap in Robotic Manipulation Tasks. Comment: Submitted to ICRA 2019 (Under Review).

[132] MuJoCo — Advanced Physics Simulation.

[133] Robot simulator CoppeliaSim: Create, compose, simulate, any robot - Coppelia Robotics.

[134] Marian Körber, Johann Lange, Stephan Rediske, Simon Steinmann, and Roland Glück. Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning.

[135] Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. Sim-to-Real Transfer with Neural-Augmented Robot Simulation. In *Proceedings of The 2nd Conference on Robot Learning*, pages 817–828. PMLR.

[136] Anurag Ajay, Maria Bauza, Jiajun Wu, Nima Fazeli, Joshua B. Tenenbaum, Alberto Rodriguez, and Leslie P. Kaelbling. Combining Physical Simulators and Object-Based Networks for Control. Comment: ICRA 2019; Project page: http://sain.csail.mit.edu.

[137] Eric Heiden, David Millard, Erwin Coumans, and Gaurav S. Sukhatme. Augmenting Differentiable Simulators with Neural Networks to Close the Sim2Real Gap.

[138] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S. Sukhatme. NeuralSim: Augmenting Differentiable Simulators with Neural Networks. Comment: Accepted at IEEE International Conference on Robotics and Automation (ICRA) 2021.

[139] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. TossingBot: Learning to Throw Arbitrary Objects with Residual Physics. Comment: Summary Video: https://youtu.be/f5Zn2Up2RjQ Project webpage: https://tossingbot.cs.princeton.edu.

[140] Kei Ota, Devesh K. Jha, Diego Romeres, prefix=van useprefix=true family=Baar, given=Jeroen, Kevin A. Smith, Takayuki Semitsu, Tomoaki Oiki, Alan Sullivan, Daniel Nikovski, and Joshua B. Tenenbaum. Data-Efficient Learning for Complex and Real-Time Physical Problem Solving using Augmented Simulation. Comment: Under submission.

[141] Michael Lutter, Christian Ritter, and Jan Peters. Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning. Comment: Published at ICLR 2019.

[142] Miles Cranmer. Lagrangian Neural Networks.

[143] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian Neural Networks. Comment: Conference paper at NeurIPS 2019. Main paper has 8 pages and 5 figures.

[144] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C. Karen Liu. Fast and Feature-Complete Differentiable Physics for Articulated Rigid Bodies with Contact.

[145] Eric Heiden, David Millard, Hejia Zhang, and Gaurav S. Sukhatme. Interactive Differentiable Simulation.

[146] Michael Lutter, Johannes Silberbauer, Joe Watson, and Jan Peters. Differentiable Physics Models for Real-world Offline Model-based Reinforcement Learning.

[147] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C. Lin. Scalable Differentiable Physics for Learning and Control.

[148] Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting. Comment: Accepted at Robotics: Science and Systems 2021.

[149] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – A Differentiable Physics Engine for Large Scale Rigid Body Simulation. Comment: 9 pages + 12 pages of appendices and references. In submission at NeurIPS 2021 Datasets and Benchmarks Track.

[150] Jack Collins, Ross Brown, Jürgen Leitner, and David Howard. Follow the Gradient: Crossing the Reality Gap using Differentiable Physics (RealityGrad). Comment: 8 Pages.

[151] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable Programming for Physical Simulation. Comment: Published at ICLR 2020.

[152] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradSim: Differentiable simulation for system identification and visuomotor control. Comment: ICLR 2021. Project page (and a dynamic web version of the article): https://gradsim.github.io.

[153] K. Arnavaz, M. Kragballe Nielsen, P. G. Kry, M. Macklin, and K. Erleben. Differentiable Depth for Real2Sim Calibration of Soft Body Simulations. 42(1):277–289.

[154] Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated Policy Learning with Parallel Differentiable Simulation. Comment: International Conference on Learning Representations.

[155] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation. Comment: Conference on Robot Learning, 2020.

[156] Fei Liu, Zihan Li, Yunhai Han, Jingpei Lu, Florian Richter, and Michael C. Yip. Real-to-Sim Registration of Deformable Soft Tissue with Position-Based Dynamics for Surgical Robot Autonomy. Comment: Fei Liu and Zihan Li contributed equally to this work. Final version of this paper is going to be published in ICRA 2021.

[157] Moritz A. Graule, Thomas P. McCarthy, Clark B. Teeple, Justin Werfel, and Robert J. Wood. SoMoGym: A Toolkit for Developing and Evaluating Controllers and Reinforcement Learning Algorithms for Soft Robots. 7(2):4071–4078.

[158] Zhou Xian, Bo Zhu, Hsiao-Yu Tung, and Antonio Torralba. FLUIDLAB: A DIFFERENTIABLE ENVIRONMENT FOR BENCHMARKING COMPLEX FLUID MANIPULATION.

[159] Featherstone Roy. *Rigid Body Dynamics Algorithms*. Springer New York, NY.

[160] Haining Luo and Yiannis Demiris. Benchmarking and Simulating Bimanual Robot Shoe Lacing. 9(10):8202–8209.

[161] Adam Smith. HOW-TO: Motion Tracking with PhaseSpace.

[162] J. N. Reed and R. D. Tillett. Initial experiments in robotic mushroom harvesting. 4(3):265–279.

[163] R. Noble, J. N. Reed, S. Miles, A. F. Jackson, and J. Butler. Influence of Mushroom Strains and Population Density on the Performance of a Robotic Harvester. 68(3):215–222.

[164] J.N. Reed, S.J. Miles, J. Butler, M. Baldwin, and R. Noble. AE—Automation and Emerging Technologies. 78(1):15–23.

[165] Mingsen Huang, Xiaohu Jiang, Long He, Daeun Choi, John Pecchia, and Yaoming Li. Development of a Robotic Harvesting Mechanism for Button Mushrooms. 64:565–575.

[166] Mingsen Huang, Long He, Daeun Choi, John Pecchia, and Yaoming Li. Picking dynamic analysis for robotic harvesting of Agaricus bisporus mushrooms. 185:106145.

[167] A. McGarry and K.S. Burton. Mechanical properties of the mushroom, Agaricus bisporus. 98(2):241–245.

[168] J. Bohdziewicz, G. Czachor, and P. Grzemski. Anisotropy of mechanical properties of mushrooms (Agaricus bisporus (J.E. Lange) Imbach). R. 17, nr 4, t. 2.

[169] Ali Roshanianfard and Noboru Noguchi. Characterization of pumpkin for a harvesting robot. 51(17):23–30.

[170] Ji Wei, Ding Yi, Xu Bo, Chen Guangyu, and Zhao Dean. Adaptive Variable Parameter Impedance Control for Apple Harvesting Robot Compliant Picking. 2020:1–15.

[171] Lingxin Bu, Guangrui Hu, Chengkun Chen, Adilet Sugirbay, and Jun Chen. Experimental and simulation analysis of optimum picking patterns for robotic apple harvesting. 261:108937.

[172] Kai Junge, Catarina Pires, and Josie Hughes. Lab2Field transfer of a robotic raspberry harvester enabled by a soft sensorized physical twin. 2(1):40.

[173] S. S. Mehta, W. MacKunis, and T. F. Burks. Nonlinear Robust Visual Servo Control for Robotic Citrus Harvesting. 47(3):8110–8115.

[174] Mahyar Abdeetedal and Mehrdad R. Kermani. Grasp synthesis for purposeful fracturing of object. 105:47–58.

[175] Mahyar Abdeetedal and Mehrdad R. Kermani. Grasp synthesis for purposeful fracturing of object. 105:47–58.

[176] Cosimo Della Santina, Christian Duriez, and Daniela Rus. Model Based Control of Soft Robots: A Survey of the State of the Art and Open Challenges. 43(3):30–65. Comment: 69 pages, 13 figures.

[177] Zhongkui Wang and Shinichi Hirai. Soft Gripper Dynamics Using a Line-Segment Model With an Optimization-Based Parameter Identification Method. 2(2):624–631.

[178] Sami Haddadin, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jähne, Lukas Hausperger, and Simon Haddadin. The Franka Emika Robot: A Reference Platform for Robotics Research and Education. 29(2):46–64.

[179] Introducing the Intel® RealSense™ Depth Camera D455.

[180] Depth Camera D405.

[181] Sebastián Valladares, Mayerly Toscano, Rodrigo Tufino, Paulina Morillo, and Diego Vallejo. Performance Evaluation of the Nvidia Jetson Nano Through a Real-Time Machine Learning Application. pages 343–349.

[182] Frankaemika/franka_ros.

[183] MoveIt Motion Planning Framework.

[184] On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward.

[185] Ewen Dantec, Michel Taïx, and Nicolas Mansard. First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback. 7(2):4448–4455.

[186] Thomas George Thuruthel, Federico Renda, and Fumiya Iida. First-Order Dynamic Modeling and Control of Soft Robots. 7:95.

[187] Nikolaos Peladarinos, Dimitrios Piromalis, Vasileios Cheimaras, Efthymios Tserepas, Radu Adrian Munteanu, and Panagiotis Papageorgas. Enhancing Smart Agriculture by Implementing Digital Twins: A Comprehensive Review. 23(16):7128.

[188] Daode Zhang, Wei Zhang, Hualin Yang, and Haibing Yang. Application of Soft Grippers in the Field of Agricultural Harvesting: A Review. 13(1):55.

[189] Costanza Armanini, Kai Junge, Philip Johnson, Charles Whitfield, Federico Renda, Marcello Calisti, and Josie Hughes. Soft robotics for farm to fork: Applications in agriculture & farming. 19(2).

[190] A.T. Miller and P.K. Allen. Graspit! A versatile simulator for robotic grasping. 11(4):110–122.

[191] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. Comment: This is an extended version of "Learning Hand-Eye Coordination for Robotic Grasping with Large-Scale Data Collection," ISER 2016. Draft modified to correct typo in Algorithm 1 and add a link to the publicly available dataset.

[192] prefix=ten useprefix=false family=Pas, given=Andreas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp Pose Detection in Point Clouds. Comment: arXiv admin note: text overlap with arXiv:1603.01564.

[193] Jeffrey Mahler, Florian T. Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964.

[194] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. Comment: To appear at Robotics: Science and Systems 2017.

[195] Jeffrey Mahler and Ken Goldberg. Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 515–524. PMLR.

[196] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. Dex-Net 3.0: Computing Robust Robot Vacuum Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning. Comment: Accepted to ICRA 2018.

[197] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. 4(26):eaau4984.

[198] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes. Comment: ICRA 2021. Video of the real world experiments and code are available at https://research.nvidia.com/publication/2021-03_Contact-GraspNet%3A–Efficient.

[199] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. ACRONYM: A Large-Scale Grasp Dataset Based on Simulation.

[200] Thomas Weng, David Held, Franziska Meier, and Mustafa Mukadam. Neural Grasp Distance Fields for Robot Manipulation. Comment: Accepted to ICRA 2023.

[201] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. DexGraspNet: A Large-Scale Robotic Dexterous Grasp Dataset for General Objects Based on Simulation.

[202] Lirui Wang, Yu Xiang, and Dieter Fox. Manipulation Trajectory Optimization with Online Grasp Synthesis and Selection.

[203] Panagiotis Mavridis, Nikolaos Mavrikis, Athanasios Mastrogeorgiou, and Panagiotis Chatzakos. Low-cost, accurate robotic harvesting system for existing mushroom farms. In *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 144–149.

[204] Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo.

[205] Corrado Pezzato, Chadi Salmi, Max Spahn, Elia Trevisan, Javier Alonso-Mora, and Carlos Hernandez Corbato. Sampling-based Model Predictive Control Leveraging Parallelizable Physics Simulations. Comment: Submitted to RA-L. Code and videos available at https://sites.google.com/view/mppi-isaac/.

[206] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation. Comment: Accepted for oral presentation at the Conference on Robot Learning (CoRL), 2021. Code available at: https://github.com/NVlabs/storm.

[207] Tao Pang, H. J. Terry Suh, Lujie Yang, and Russ Tedrake. Global Planning for Contact-Rich Manipulation via Local Smoothing of Quasi-dynamic Contact Models. Comment: The first two authors contributed equally to this work.

[208] Julius Jankowski, Lara Brudermüller, Nick Hawes, and Sylvain Calinon. VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior. Comment: *Authors contributed equally.

[209] Rachel Holladay, Tomás Lozano-Pérez, and Alberto Rodriguez. Planning for Multi-stage Forceful Manipulation. Comment: Accepted to IEEE ICRA 2021. For videos see: https://mcube.mit.edu/forceful-manipulation/.

[210] T Maruccci, J Umenberger, P A Parrilo, and R Tedrake. Shortest Paths in Graphs of Convex Sets.

[211] Elia Trevisan and Javier Alonso-Mora. Biased-MPPI: Informing Sampling-Based Model Predictive Control by Fusing Ancillary Controllers. 9(6):5871–5878. Comment: Accepted for Robotics and Automation Letters, April 2024.

[212] Mehdi Testouri, Gamal Elghazaly, and Raphael Frank. Towards a Safe Real-Time Motion Planning Framework for Autonomous Driving Systems: An MPPI Approach.

[213] Mehran Raisi, Amirhossein Noohian, and Saber Fallah. A fault-tolerant and robust controller using model predictive path integral control for free-flying space robots. 9.

[214] Anastasia Mavrommati, Carlos Osorio, Roberto G. Valenti, Akshay Rajhans, and Pieter J. Mosterman. An Application of Model Predictive Control to Reactive Motion Planning of Robot Manipulators. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 915–920.

[215] Leonard Jung, Alexander Estornell, and Michael Everett. Contingency Constrained Planning with MPPI within MPPI. Comment: 12 Pages, 6 Figures.

[216] Denton Cockburn and Ziad Kobti. Memory-Bounded D* Lite.

[217] Fahad Islam, Chris Paxton, Clemens Eppner, Bryan Peele, Maxim Likhachev, and Dieter Fox. Alternative Paths Planner (APP) for Provably Fixed-time Manipulation Planning in Semi-structured Environments.

[218] Jyh-Ming Lien and Yanyan Lu. Planning Motion in Environments with Similar Obstacles.

[219] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model Predictive Path Integral Control: From Theory to Parallel Computation. 40(2):344–357.

[220] Yulun Zhuang and Ziqi Han. Bayesian Optimization for MPPI Control of Robot Arm Planar Pushing.

[221] Liwei Hou, Hengsheng Wang, Haoran Zou, and Yalin Zhou. Robotic Manipulation Planning for Automatic Peeling of Glass Substrate Based on Online Learning Model Predictive Path Integral. 22(3):1292.

[222] Pedram Rabiee and Jesse B. Hoagg. Guaranteed-Safe MPPI Through Composite Control Barrier Functions for Efficient Sampling in Multi-Constrained Robotic Systems. Comment: Preprint submitted to American Control Conference (ACC) 2025.

[223] Hardik Parwana, Mitchell Black, Georgios Fainekos, Bardh Hoxha, Hideki Okamoto, and Danil Prokhorov. Model Predictive Path Integral Methods with Reach-Avoid Tasks and Control Barrier Functions.

[224] Alexandros Nikou, Shahab Heshmati-alamdari, and Dimos V. Dimarogonas. Scalable time-constrained planning of multi-robot systems. 44(8):1451–1467.

[225] Mohammad Rastegarpanah, Mohammed Eesa Asif, Javaid Butt, Holger Voos, and Alireza Rastegarpanah. Mobile robotics and 3D printing: Addressing challenges in path planning and scalability. 19(1):e2433588.

[226] Thomas Parr, Giovanni Pezzulo, and Karl J. Friston. *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. The MIT Press.

[227] Yuezhe Zhang, Corrado Pezzato, Elia Trevisan, Chadi Salmi, Carlos Hernández Corbato, and Javier Alonso-Mora. Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning. Comment: Submitted to RA-L. Experiments' videos are available at https://sites.google.com/view/m3p2i-aip.

[228] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation.

[229] Kaarel Hänni, Jake Mendel, Dmitry Vaintrob, and Lawrence Chan. Mathematical Models of Computation in Superposition. Comment: 28 pages, 5 figures. Published at the ICML 2024 Mechanistic Interpretability (MI) Workshop.

[230] Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases.

[231] Leonard Bereska and Efstratios Gavves. Mechanistic Interpretability for AI Safety – A Review.

[232] David Ha and Jürgen Schmidhuber. World Models.