1 Introduction

Modern robots are showing increasing intelligence and capability, being able to complete complex tasks across many industries, from manufacturing and logistics to critical real-world missions like search and rescue, patrol, and delivery (Liu et al., 2022). However, unlike animals which possess a remarkable ability to sustain operation after injury by creating compensatory behaviours (Bongard et al., 2006), current robotic systems often remain fragile when faced with unexpected hardware failures. Actuators can experience issues like joint locking, free-swinging, or broken components (Liu et al., 2022), while manipulators used in applications such as surgery or space exploration, which require careful fine-tuning, are prone to the same faults that could lead to catastrophic consequences (Pham et al., 2024). Such failures not only risk harm but also casue unwanted downtime and decreases the availability (Liu et al., 2022). Achieving **robust fault tolerance**— which is the ability for a system to continue operation under faulty conditions (Blanke et al., 1997; Ahmed et al., 2020)—is an important consideration in the design and control of autonomous systems, particularly for complex robots operating in unpredictable environments (Liu et al., 2022).

Reinforcement Learning (RL) is a powerful framework for developing controllers for such systems , it allows agents to learn control strategies - policies, directly from interaction data (Andrew and Richard S,2018). A policy decides the robot's actions based on it's current state and environment, with the aim to maximise long-term performance, measured by the total reward which it accumalates over its current lifespan. In the context of achieving fault tolerance and RL, there have been numerous stratergies that have been developed. **This literature review will primarily focus on methods that aim to achieve robustness proactively during the policy learning phase.** In specific, we will focus on approach where hardware failures, such as actuator malfunctions or sensors become noisy - which leads to them not giving accurate data, are injected during training. With the main goal of developing control policies that are resillient to these faults when deployed to the real world. This RL methodology contrasts with the traditional **Fault-Tolerant Control (FTC)** architectures used, which rely on seperate modules for fault detection and diagnosis which lead to accomodation stratergies used *after* a fault occurs (Blanke et al., 1997; Ahmed et al., 2020). The field of RL has many relevant paradigms to achieve fault tolerance, such as **Domain Randomization (DR)** which trains policies with varied simulation parameters (Liu et al., 2022; Rajeswaran et al., 2016), **Adversarial Training** where policies learn against opponents which try to disrupt your behaviour, **Online Adaptation** techniques where the robot learns the compensatory behaviours to damage *after* it is damaged (Cully et al., 2015; Chatzilygeroudis et al., 2018; Pham et al., 2024) and **Policy Regularization** methods that promote smooth or stable policies, which directly contribute to robustness (e.g., Shen et al., 2020).

Policies trained with randomization may still fail when they experience failure modes not learnt or seen in entirety in training, or they may become overly conservative (Rajeswaran et al., 2016). Therefore developing effective training strategies that directly incorporate realistic failure modes during training, such as limb dropout - central to this review's focus, can produce a notable improvement to the robot's resilience. The development of policies robust to common hardware failures like actuator malfunction or sensor noise enablea more reliable long-term usage of autonomous systems in complex, safety-critical, or remote environments, reducing the need for constant human oversight or complex recovery mechanisms (Cully et al., 2015; Liu et al., 2022).

This review aims to synthesize the current state of research relevant to learning robust policies via training-time failure injection. Section 2 provides necessary background concepts. Section 3 explores the different approaches identified in the literature for achieving robustness, detailing the specific reinforcement learning methodologies employed within each paradigm. Section 4 discusses how policy robustness is evaluated, and Section 5 concludes the review.

2 Background

This section introduces the foundational concepts of reinforcement learning, the specifics of applying it to continuous robot control, the challenges necessitating robustness, and the primary paradigms developed to address these challenges.

2.1 Reinforcement Learning for Continuous Control

Reinforcement Learning (RL) is a framework where an agent **learns** to make decisions by interacting with an environment (Andrew and Richard S, 2018). it then receives a *scalar* reward based the quality of the *action* it decides to take, and thereafter *transitions* to a new state. The main goal of the agent is to learn a *policy*, which is a mapping from *states* to *actions* (or a distribution over actions), that maximises it's expected sum of rewards over time. An extension to RL is *Deep Reinforcement Learning (DRL)*, in this framework the policy and associated value functions are represented using deep neural networks. This allows RL to handle problems with high-dimensional states and action spaces. It has shown success in many domains from games to robotics (Mnih et al., 2015; Levine et al., 2018; Silver et al., 2017) and is a strong extension for **FTC**.

Classic RL is usually used with discrete action sets (board games, recommender systems, etc.), whereas robotics applications mainly involve *continuous control* (Kober et al., 2013). The state space (joint angles, velocities, sensor readings) and the action space (motor torques, joint positions) are real-valued variables. These real-valued variables and high-dimensionality of data lead to an infinite number of possible actions the agent may take, prompting the need for meticulous algorithm design (Kober et al., 2013; Recht, 2019). A common approach to continuous control utilises an *actor-critic* structure, in which an 'actor' network will learn the policy and the 'critic' network will learn to evaluate state-action pairs. Additional algorithms commonly used in this domain, also relevant to the works discussed here, include: Proximal Policy Optimization (PPO) (Schulman et al., 2017; Liu et al., 2022; Pham et al., 2024), Soft Actor-Critic (SAC) (Haarnoja et al., 2018), and Twin-Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018; Jia et al., 2023).

2.2 The Need for Robustness and Fault Tolerance

Robots trained in simulation which are then deployed to the real world, sooner or later encounter conditions that differ from their training environment. They may face: component degradation, sensor noise and hardware failures (Liu et al., 2022; Pham et al., 2024). This fact directly contributes to the need for control policies that are **robust** - able to maintain acceptable performance despite uncertainty or variations in the system's dynamics, observations, actions, or the environment itself (Pinto et al., 2017; Glossop et al., 2022). We can generalise this to unseen conditions and resilience against disturbances (Zhou and Doyle, 1998). *Fault-Tolerant Control (FTC)*, defined as the ability of a system to continue operation, with varying performance, under faulty conditions (as cited by Ahmed et al., 2020; Blanke et al., 1997). The difference lies where traditional FTC methods use explicit fault detection, diagnosis and accommodation steps (Ahmed et al., 2020; Zhang and Jiang, 2008), we aim to achieve fault tolerance implicitly with a learning-based approach which has a resilient policy. The types of failures this review will be dealing with are actuator malfunctions (joint locking) (Liu et al., 2022), intermittent failures (Pham et al., 2024), limb damage or loss (Cully et al., 2015; Chatzilygeroudis et al., 2018; Bongard et al., 2006), gradual degradation of components (Ahmed et al., 2020), as well as sensor noise and external disturbances (Glossop et al., 2022; Pinto et al., 2017).

2.3 Simulation-Based Learning and the Sim-to-Real Gap

Training directly on physical robots brings rise to multiple challenges such as, it can be very expensive, time consuming and potentially unsafe (Kober et al., 2013). This is why DRL for robotics relies on physics-based simulation environments (MuJoco, Isaac Gym) (Todorov et al., 2012; Makoviychuk et al., 2021). These simulators allow for rapid and safe development. However, another challenge arises, which is the *sim-to-real gap*: the unavoidable discrepancies between the simulated physics and the complexities of the real world (Zhao et al., 2020). These discrepancies come from dynamics which have not been accounted for (e.g., motor backlash, flexibility), sensor noise characteristics, or actuation delays (Peng et al., 2018) or inaccurate system identification (e.g., friction, mass). Therefore, policies that are trained only in simulation perform poorly or fail entirely on physical hardware (Pinto et al., 2017; Rajeswaran et al., 2016). Bridging this gap is a primary focus for developing robust learning techniques, such as Domain Randomization (DR), that trains policies capable of tolerating these modelling errors and environmental variations upon transfer.

2.4 Key Robustness Paradigms: An Overview

Several distinct paradigms, which form the basis of the approaches discussed in Section 3 (replace with your actual section label), are commonly used to imbue RL policies with robustness:

- **Domain Randomization (DR):** A technique primarily aimed at improving sim-to-real transfer and generalization by intentionally randomizing parameters of the *simulation* environment during training (Tobin et al., 2017). The randomisation parameters may be the intentional injection of: varying dynamic parameters such as mass, friction and damping (Peng et al., 2018; Rajeswaran et al., 2016), sensor noise, delays and crucially for this review, simulated hardware faults and reduced actuator effectiveness (Liu et al., 2022). By injecting these faults and variations during simulation, the policy is encouraged to learn solutions that are robust to these variations, therefore generalizing better to the real world (for sim-to-real transfer) and specific failure conditions (limb dropout).

- **Adversarial Training:** This involves two entities: 1) our primary RL agent -protagonist and 2) an adaptive adversary. In Adversarial Training the goal of the adaptive adversary is to hinder the protagonist's performance. It does this by learning forces to apply to the protagonist that optimally destabilize protagonist (Pinto et al., 2017) or directly perturb the protagonist's actions (Tessler et al., 2019). This paradigm then forces the protagonist to learn policies that are robust to worst-case scenarios (applied by the adversary), which in turn effectively models uncertainty and potential disruptions as an opponent.

- **Online Adaptation / Recovery:** This subset of methods allow the robot to adjust its behaviour *after* damage or unexpected changes occur during its deployment. Repertoire-based methods which use Quaility-Diversity (QD) algorithms such as MAP-Elites (Mouret and Clune, 2015) to generate a wide subset of skills offline, which are then selected or adapted online using techniques like Bayesian Optimization or MCTS (Cully et al., 2015; Chatzilygeroudis et al., 2018; Allard et al., 2023) fall under this subset of methods. Other approaches use online RL updates to adapt directly (Pham et al., 2024) or use self-modeling to infer the robot's changed state (Bongard et al., 2006).

- **Policy Regularization:** This approach aims to achieve robustness by directly modifying the objective function associated with the RL algorithm, to promote certain policy characteristics. Of these characteristics we have smoothness (penalises large action changes for small state changes), with the aim of potentially making the policy less sensitive to state changes or noise (Shen et al., 2020)

Understanding these basic concepts and strategies used helps us understand - in detail, the specific methodologies used to create robust and fault-tolerant control policies.

3 Approaches to Learning Robust Control Policies

Different fundamental strategies are involved in achieving robustness in RL policies. How we achieve the robustness in these policies, has a significant impact on our training requirements and the types of failures it can handle. We can categorise these approaches based on whether they build robustness *proactively* - during a training-based simulation phase or are *reactive* - adapt after a failure occurs during deployment.

3.1 Proactive Training for Robustness

Proactive approaches aim to learn a single control policy, usually with Deep Reinforcement Learning (DRL), that is robust to a range of variations or failures that is knows of before the robot is deployed.

3.1.1 Domain Randomization and Ensemble Training

A dominant proactive strategy is **Domain Randomization (DR)**, which is crucial for fault tolerance. The randomization is extended to include simulated hardware failures. Liu et al. (2022) demonstrates this well by randomizing joint-locking failures (which joint, when, and severity) during PPO training within a teacher-student framework, achieving zero-shot transfer of a fault-tolerant locomotion policy for a quadruped. Similarly, Jia et al. (2023) trained a TD3 agent for launch vehicle control while incorporating random thrust drops in the actuators during training episodes.

Ensemble Training a related concept to *Domain Randomisation*. In *Ensemble Training* instead of just randomising the parameters, the RL agent (e.g. using TRPO) is trained over an explicitly defined distribution or ensemble of simulator models. Rajeswaran et al. (2016) further betters robustness in this setting by applying adversarial sampling (optimizing for Conditional Value at Risk), by focusing on policy updates on the simulated models where the current policy performs worst. Training policies to be robust against simulated limb dropout is a direct application of these DR principles.

3.1.2 Adversarial Training

Tessler et al. (2019) adapt this idea to action uncertainty, developing Action Robust RL (using variants of DDPG) where the adversary learns to optimally perturb or replace the protagonist's intended actions. These methods directly model uncertainty or potential failures as an intelligent adversary to cause the protagonist to learn robust policies.

3.1.3 Policy Regularization

Shen et al. (2020) proposed Smooth Regularized Reinforcement Learning ([def:srl]SR^{2}L), which adds a regularisation term to standard DRL algorithms (like TRPO or DDPG). This term penalizes the policy for having large output changes for small input state changes. By making sure our policy is smooth, we can create policies less sensitive to sensor noise or small state changes, which lead to overall stability and robustness.

3.2 Reactive Adaptation and Recovery Methods

Contrasting with proactive training, reactive methods focus on enabling the robot to adapt its behaviour online, *after* damage has occurred or environmental conditions have unexpectedly changed.

3.2.1 Repertoire-Based Adaptation

A prominent line of work utilizes behavioural repertoires generated offline using Quality-Diversity (QD) algorithms like MAP-Elites. In the Intelligent Trial-and-Error (ITE) algorithm (Cully et al., 2015), a diverse map of behaviours from the *intact* robot is generated in simulation. After damage, Bayesian Optimization, guided by a Gaussian Process (GP) model updated with physical trials, rapidly searches this map to find a compensatory behaviour. Reset-Free Trial-and-Error (RTE) (Chatzilygeroudis et al., 2018) builds on this, using MCTS for online planning with the learned GP model and enabling reset-free operation. Hierarchical Trial-and-Error (HTE) (Allard et al., 2023) further extends this by using hierarchical repertoires to manage complexity and increase skill diversity, employing Bayesian Optimization to select appropriate low-level execution details online. Notably, while these methods leverage machine learning (GPs, BayesOpt, MCTS, MAP-Elites), the core online adaptation typically does not involve standard DRL policy updates.

3.2.2 Direct Reinforcement Learning Adaptation

Other approaches use RL more directly for online adaptation. Pham et al. (2024) frame joint failure in manipulators as a Partially Observable Markov Decision Process (POMDP) and train a PPO agent to learn a policy that can implicitly infer the joint's status from observations and compensate dynamically during task execution. Ahmed et al. (2020) address gradual degradation by mixing online PPO updates with offline training on a frequently updated data-driven model of the system, allowing the policy to track slow changes.

3.2.3 Continuous Self-Modeling

A distinct adaptive philosophy is continuous self-modeling (Bongard et al., 2006). Here, the robot does not rely on pre-programmed models or repertoires but instead uses ongoing interaction and internal optimization processes to continuously infer its own current physical structure. If damage occurs, this process allows the robot to generate a new self-model reflecting the change, which is then used to synthesize a novel compensatory behaviour. This approach focuses on self-discovery rather than learning from external reward signals in the typical RL sense.

3.3 Comparison of Proactive and Reactive Paradigms

The choice between proactive training and reactive adaptation involves significant trade-offs, mirroring the comparison between searching in generator versus level space in procedural content generation (**?**).

Proactive methods (DR, Adversarial Training, Regularization) invest computational effort offline to produce a single policy intended to be robust upon deployment.

- *Advantages:* Can react instantly to failures within their trained distribution, potentially simpler deployment logic.
- *Disadvantages:* Require extensive simulation covering potential failures, may struggle with truly novel failures not anticipated in training, risk of learning overly conservative policies, performance depends heavily on the quality of the simulation and randomization strategy.

Reactive methods (Repertoire-based, Online RL Adaptation, Self-Modeling) focus on algorithms that enable adaptation during deployment.

- *Advantages:* Potential to handle novel or unforeseen failures, can fine-tune behaviour to the specific damage experienced.
- *Disadvantages:* Adaptation takes time online (not instantaneous), performance depends critically on the speed and success of the adaptation process, may require significant online computation, reset-free learning/adaptation can be difficult.

The selection of an appropriate strategy depends on the specific application, the predictability of failures, the available computational resources (offline vs online), and the required reaction speed. For instance, training against simulated limb dropout represents a proactive strategy aiming for innate resilience.

4 Evaluation of Policy Robustness

Evaluating the effectiveness and robustness of learned control policies, particularly under potential hardware failures, is crucial for assessing progress and understanding limitations. However, comparing results across different studies presents significant challenges due to variations in tasks, metrics, failure modes, and evaluation protocols (cf. Glossop et al., 2022). This section discusses the common evaluation approaches and metrics found in the reviewed literature.

4.1 Direct Performance Metrics

A fundamental aspect of evaluation is measuring the policy's ability to successfully perform its intended task under specific conditions (nominal or faulty). This is often quantified by the cumulative reward achieved during an episode, which encapsulates the task objectives (e.g., Pinto et al., 2017; **?**). For locomotion tasks, more specific metrics include average forward velocity, measures of stability (such as fall rate or torso orientation), and survival time (how long the agent operates before failing), particularly *after* a failure is introduced (Liu et al., 2022). For goal-oriented tasks like manipulation or navigation, common metrics are the task success rate (e.g., percentage of successful drawer openings or maze completions) and the average time or number of actions required to complete the task successfully (Pham et al., 2024; Chatzilygeroudis et al., 2018; Allard et al., 2023). Learning curves plotting average reward against training steps or time are also frequently used to assess the sample efficiency and stability of the learning process itself (**??**).

4.2 Measuring Robustness and Generalization

Beyond baseline performance, specific evaluations target robustness. A prevalent method involves testing policies trained under certain conditions against a range of *unseen* variations at test time. This often includes varying physical parameters of the simulated robot or environment, such as mass or friction coefficients, and observing performance degradation (**?**Pinto et al., 2017; Tessler et al., 2019). Another approach, systematically employed by Glossop et al. (2022), involves injecting various types of disturbances (e.g., noise or forces applied to states, actions, or dynamics) with increasing magnitude at test time and measuring the drop in performance.

For methods focusing on adaptation or recovery after failure, key metrics include the *time* or *number of trials* required for the robot to regain a threshold level of performance after damage occurs (Cully et al., 2015; Chatzilygeroudis et al., 2018). Finally, for policies developed primarily in simulation, successful *sim-to-real transfer*, often assessed by zero-shot performance on physical hardware under nominal and faulty conditions, is a critical validation step (Liu et al., 2022).

4.3 Handling Policy Failures and Constraints in Evaluation

Evaluating robustness inherently involves scenarios where policies fail (e.g., the robot falls, collides, becomes unstable, or cannot complete the task). How these failures are treated within the evaluation process is important. Some studies use metrics like survival time or task success rate, which directly quantify the policy's ability to avoid or overcome catastrophic failure (Liu et al., 2022; Pham et al., 2024; Allard et al., 2023). Others terminate episodes upon critical failure, implicitly penalizing the policy through a lower cumulative reward or shorter operational time. For adaptation-focused methods, the number of attempts required before achieving success under damage is often reported (Chatzilygeroudis et al., 2018; Cully et al., 2015). Physical hardware testing introduces inherent safety constraints, often limiting the types or severity of failures that can be practically and safely induced (cf. Liu et al., 2022).

4.4 Comparability and Evaluation Challenges

Directly comparing robustness results across different studies remains a significant challenge. As highlighted by benchmarking efforts (Glossop et al., 2022), researchers employ diverse robotic platforms (quadrupeds, manipulators, wheeled robots, abstract models), simulation environments (MuJoCo, Isaac Gym, PyBullet, custom), tasks (locomotion, manipulation, balancing), failure modes (joint locking, degradation, external forces, action noise, limb loss), disturbance injection methodologies, and performance metrics. For instance, evaluating robustness to 'limb dropout' lacks a standard protocol; different works might simulate it as zero torque, joint locking, or physical removal, each potentially leading to different outcomes (cf. Liu et al., 2022; Cully et al., 2015; Bongard et al., 2006). Furthermore, the sim-to-real gap implies that even strong performance under randomized or adversarial conditions in simulation does not guarantee equivalent robustness on physical hardware without explicit validation (Liu et al., 2022; **?**). The development and adoption of standardized benchmark suites, tasks, failure models (particularly for common hardware issues like actuator malfunctions), and evaluation metrics would greatly benefit the field by enabling more rigorous and meaningful comparisons between different approaches to learning robust and fault-tolerant control policies.

5 Conclusion

This review has surveyed the significant body of work aimed at developing robust control policies for robots using learning-based methods, with a particular focus on strategies relevant to handling hardware failures, such as actuator malfunctions and sensor noise, within continuous control domains. Considerable progress has been made, revealing diverse strategies ranging from *proactive* methods like Domain Randomization and Adversarial Training that build resilience during the learning phase, to *reactive* methods involving online adaptation or self-modeling after a failure occurs, alongside techniques that regularize policies for inherent stability.

Despite these advancements, several key challenges and gaps remain in the pursuit of truly resilient robotic systems capable of reliable long-term autonomous operation, particularly concerning common hardware failures:

- **Handling Abrupt Structural Failures:** While robustness to parameter variations, noise, or gradual degradation is increasingly addressed, effectively training policies to handle sudden, significant structural changes like complete limb or actuator failure (limb dropout) remains challenging, especially ensuring generalization across different failure types and locations.

- **Reliable Sim-to-Real Transfer for Fault Tolerance:** Although techniques like Domain Randomization targeting specific failure modes show promise (Liu et al., 2022), achieving consistent and reliable zero-shot transfer of policies trained under simulated failure conditions to diverse physical hardware remains a critical hurdle.

- **Systematic Comparison of Robustness Strategies:** There is a notable lack of direct, large-scale comparisons between proactive training strategies (e.g., failure-specific DR, adversarial methods) and reactive adaptation strategies (e.g., repertoire-based methods) for handling common hardware failures under comparable conditions, making it difficult to determine the most effective approach for different contexts.

- **Standardized Evaluation Protocols:** The field currently lacks widely adopted benchmarks, standardized failure simulation methods (especially for actuators), and unified metrics for rigorously evaluating and comparing the fault tolerance and robustness of different learning-based control approaches across varied robotic platforms and tasks, hindering objective assessment of progress (cf. Glossop et al., 2022).

Therefore, while the reviewed methodologies offer promising avenues towards more resilient robots, the overarching goal of creating controllers that reliably maintain function despite common hardware failures like limb dropout has not yet been fully achieved. Future work focusing on addressing the identified gaps—particularly through improved simulation of realistic failures during training, enhanced sim-to-real transfer techniques for fault-tolerant policies, systematic comparisons between proactive and reactive paradigms, and the development of robust, standardized evaluation protocols—will be crucial for advancing the practical deployment of resilient autonomous systems. Investigating targeted training strategies, such as the limb dropout concept central to this review's motivation, represents a pertinent step in this direction.

A Algorithm and Concept Definitions

This appendix provides brief definitions for key reinforcement learning algorithms and concepts mentioned in the review.

A.1 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a state-of-the-art, on-policy reinforcement learning algorithm widely used for continuous control tasks. It belongs to the actor-critic family and aims to improve training stability compared to standard policy gradient methods. PPO achieves this by optimizing a 'clipped' surrogate objective function, which restricts the size of policy updates in each iteration, preventing potentially destabilizing large changes to the policy and ensuring more reliable convergence. It balances sample efficiency and ease of implementation, making it a popular choice in robotics (e.g., Liu et al., 2022; Pham et al., 2024).

A.2 Twin-Delayed Deep Deterministic Policy Gradient (TD3)

Twin-Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018) is an off-policy, actor-critic algorithm designed to improve upon the Deep Deterministic Policy Gradient (DDPG) algorithm for continuous control. Its key innovations address the common problem of Q-value overestimation in critic networks. TD3 employs three main techniques: (1) *Clipped Double Q-Learning:* It learns two independent critic networks (the 'twins') and uses the minimum of their Q-value estimates to form the target for updates, reducing upward bias. (2) *Delayed Policy Updates:* The actor network is updated less frequently than the critic networks, allowing the critic estimates to stabilize first. (3) *Target Policy Smoothing:* Adds noise to the target action during critic updates, smoothing the value estimate around the target policy's actions. These modifications lead to significantly improved stability and performance compared to DDPG, particularly in complex continuous control tasks (e.g., Jia et al., 2023).

A.3 Trust Region Policy Optimization (TRPO)

Trust Region Policy Optimization (TRPO) (**?**) is an influential on-policy reinforcement learning algorithm that aims to achieve stable and monotonic policy improvement. It addresses a key challenge in policy gradient methods: large, poorly chosen updates can lead to catastrophic performance collapse. TRPO tackles this by constraining the size of policy updates at each iteration, ensuring that the new policy does not deviate "too far" from the old one. This constraint is typically enforced by limiting the Kullback–Leibler (KL) divergence between the action distributions of the old and new policies within a "trust region". While computationally more complex than simpler policy gradient methods (often requiring approximations like the conjugate gradient method to solve the constrained optimization problem), TRPO demonstrated significantly more stable learning on complex continuous control tasks and served as a direct precursor to the widely adopted PPO algorithm (e.g., used as a base in Pinto et al., 2017; **?**).

References

Dikai Liu, Tianwei Zhang, Jianxiong Yin, and Simon See. Saving the limping: Fault-tolerant quadruped locomotion via reinforcement learning. *arXiv preprint arXiv:2210.00474*, 2022.

Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

Tan-Hanh Pham, Godwyll Aikins, Tri Truong, and Kim-Doang Nguyen. Adaptive compensation for robotic joint failures using partially observable reinforcement learning. *Algorithms*, 17(10):436, 2024.

Mogens Blanke, Roozbeh Izadi-Zamanabadi, Søren A Bøgh, and Charlotte P Lunau. Fault-tolerant control systems—a holistic view. *Control Engineering Practice*, 5(5):693–702, 1997.

Ibrahim Ahmed, Marcos Quiñones-Grueiro, and Gautam Biswas. Fault-tolerant control of degrading systems with on-policy reinforcement learning. *IFAC-PapersOnLine*, 53(2):13733–13738, 2020.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

Konstantinos Chatzilygeroudis, Vassilis Vassiliades, and Jean-Baptiste Mouret. Reset-free trial-and-error learning for robot damage recovery. *Robotics and Autonomous Systems*, 100:236–250, 2018.

Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pages 8707–8718. PMLR, 2020.

Barto Andrew and Sutton Richard S. Reinforcement learning: an introduction. 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37 (4-5):421–436, 2018.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274, 2013.

Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1): 253–279, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Chenhui Jia, Xiaodong Liu, Zhaolei Wang, Qinghai Gong, and Xu Huang. Robust attitude controller designation of launch vehicle under actuator failure condition via deep reinforcement learning algorithm. In *2023 35th Chinese Control and Decision Conference (CCDC)*, pages 3223–3228. IEEE, 2023.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning*, pages 2817–2826. PMLR, 2017.

Catherine R Glossop, Jacopo Panerati, Amrit Krishnan, Zhaocong Yuan, and Angela P Schoellig. Characterising the robustness of reinforcement learning for continuous control using disturbance injection. *arXiv preprint arXiv:2210.15199*, 2022.

Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.

Youmin Zhang and Jin Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual reviews in control*, 32(2):229–252, 2008.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

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. IEEE, 2020.

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. IEEE, 2018.

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. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Maxime Allard, Simón C Smith, Konstantinos Chatzilygeroudis, Bryan Lim, and Antoine Cully. Online damage recovery for physical robots with hierarchical quality-diversity. *ACM Transactions on Evolutionary Learning*, 3(2): 1–23, 2023.