Towards Safe Robot Foundation Models Using Inductive Biases

Maximilian Tölle[⋆,1,2] **Theo Gruner**[⋆,1,3] **Daniel Palenicek**[⋆,1,3]
Tim Schneider[1] **Jonas Günster**[1] **Joe Watson**[4] **Davide Tateo**[1] **Puze Liu**[1,2] **Jan Peters**[1,2,3,5,6]
[⋆]Equal contribution [1]Technical University of Darmstadt [2]German Research Center for AI (DFKI)
[3]hessian.AI [4]University of Oxford [5]Robotics Institute Germany (RIG) [6]Centre for Cognitive Science

Abstract: Safety is a critical requirement for the real-world deployment of robotic systems. Unfortunately, while current robot foundation models show promising generalization capabilities across a wide variety of tasks, they fail to address safety, an important aspect for ensuring long-term operation. Current robot foundation models assume that safe behavior should emerge by learning from a sufficiently large dataset of demonstrations. However, this approach has two clear major drawbacks. Firstly, there are no formal safety guarantees for a behavior cloning policy trained using supervised learning. Secondly, without explicit knowledge of any safety constraints, the policy may require an unreasonable number of additional demonstrations to even approximate the desired constrained behavior. To solve these key issues, we show how we can instead combine robot foundation models with geometric inductive biases using ATACOM, a safety layer placed after the foundation policy that ensures safe state transitions by enforcing action constraints. With this approach, we can ensure formal safety guarantees for generalist policies without providing extensive demonstrations of safe behavior, and without requiring any specific fine-tuning for safety. Our experiments show that our approach can be beneficial both for classical manipulation tasks, where we avoid unwanted collisions with irrelevant objects, and for dynamic tasks, such as the robot air hockey environment, where we can generate fast trajectories respecting complex tasks and joint space constraints. For experimental results, see https://sites.google.com/view/safe-robot-foundation-models.

Keywords: robot foundation models, generalist policies, safety

1 Introduction

Robot foundation models (RFMs) [1, 2, 3, 4] have shown a promising direction to carry out a large set of tasks across a wide range of robotics systems using textual commands as task descriptions. The current developments of RFMs focus on enhancing the generalization across tasks, data modalities, and robot embodiments. In contrast, other key practical properties of robotic systems, such as safety, are not considered in depth. The key assumption behind this choice is that the emergent behavior will be inherently safe since RFMs are trained using behavior cloning (BC). If the data distribution only contains safe trajectories, one would expect the learned behavior to also be safe. However, while this assumption may hold in simple pick-and-place scenarios, we argue that in general settings and dynamic environments, this assumption may be limiting for many key reasons. Firstly, there is no way to ensure any formal theoretical guarantees that a RFM will satisfy any safety constraint. Scaling the amount of data will move us closer and closer to the demonstrator distribution, but this does not guarantee that the policy will not generate dangerous behavior. Secondly, often safety constraints such as collision avoidance, joint limits, or any other geometric constraints are both common in robotics applications and can be easily enforced. While it would be possible to learn a policy able to deal with these safety constraints, it may require an unreasonable amount of data, particularly to

Workshop on Safe and Robust Robot Learning for Operation in the Real World at CoRL 2025.

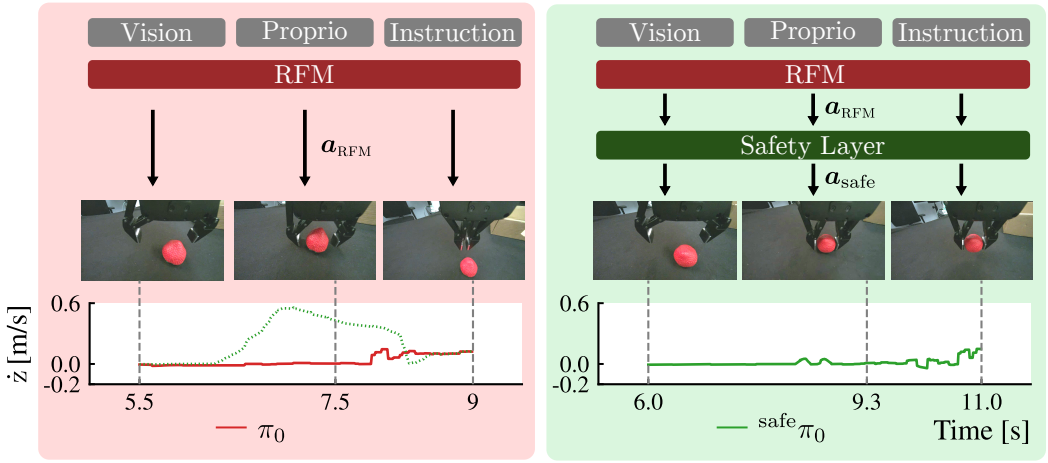

Figure 1: Our proposed safety layer can be added to the output of an arbitrary RFM, e.g., π_0. (left) Without the added ATACOM safety layer, the vanilla π_0 policy crashes the robot into the table. We highlight the importance of the safety layer by plotting the impact of the π_0 action on the end effector's vertical velocity. While vanilla π_0 corrects the z position after the impact (red), the safety layer would have engaged earlier to circumvent crashing into the table. (right) Deployment with the safety layer results in a safe rollout without pronounced z-correction.

make the policy robust to out of distribution objects, distractors, or unexpected disturbances of the workspace, such as a human or an animal entering the working area of the robot. In some cases, generating this data would not be even possible, due to the danger of allowing living beings in the work area of the robot, particularly if the robot's embodiment is not intrinsically safe, as happens in the setting of self-driving cars or heavy industrial robots.

For this reason, we believe that empowering robots with the knowledge of safety constraints and enforcing them directly at the policy level is extremely beneficial both in terms of formal safety guarantees — fundamental for the acceptance of robotics systems in society —and in terms of amount of data and computation needed to operate safely. To reach this objective, we rely on the key concept of inductive biases: instead of collecting demonstrations to account for the full space of possible solutions that RFMs can generate, we restrict our policy to only generate safe trajectories. Furthermore, we can rely on the fact that robots interact with the physical world, where the geometry of the rigid objects plays a crucial role. Using the information coming from the object geometry, which can be easily extracted from off-the-shelf perception pipelines, we can robustly generate safety geometric constraints that can be combined with existing prior knowledge about the robot to ensure safety with formal guarantees.

To achieve this ambitious goal, we propose a modular safety layer based on the ATACOM approach [5, 6], which translates known safety constraints into a constraint manifold to ensure safe actions. Due to the design of ATACOM, we can easily combine this safety layer with any RFM and enforce safe behavior w.r.t. a given set of safety constraints. Furthermore, assuming proper constraint evaluation ATACOM provides safety guarantees [6] in terms of forward invariance and input-to-state stability w.r.t. the safe set.

To further demonstrate the generality of this approach, we present a semi-automated pipeline based on SAM2 [7] to automatically build box constraints for collision avoidance with any object. This allows our method to go beyond user-defined constraints, e.g., workspace constraints or self-collision avoidance. While our proposed solution is not a fully-fledged automatic constraint generation methodology, it shows that this module could be, in principle, implemented robustly and reliably.

We extensively evaluate our approach both in simulation and in real-world platforms, using two different RFMs, π_0 and OCTO. Specifically, we focus on two different platforms, solving two very different types of tasks: a classical manipulation setup using the Franka robot, and a dynamic task,

where we train a Kuka IIWA robot to perform a hitting motion on the air hockey game. We want to prove that our methodology is both flexible and computationally efficient while ensuring safety. Our results show that our safety layer approach, despite the lack of specific fine-tuning data, is not considerably affecting the success rate, while ensuring safe policy execution both in the dynamics and quasistatic tasks.

2 A Safety Module for Robot Foundation Models

In the following, we explain in detail how to combine an arbitrary RFM with the ATACOM safety layer. Our approach follows a different framework w.r.t. existing approaches, as most approaches neglect the safety issue, assuming that enough safe data is provided to the model. Prior attempts at safety perform post-training alignment of RFMs to make the models *safer* [8]. In contrast to these approaches, we seek to make the model inherently safe on an architectural level, under the assumption of known explicit safety constraints. In principle, it is possible to use an arbitrary safety layer, however, ATACOM is a good choice in terms of robustness and simplicity.

To achieve safety through the ATACOM safety layer, we require two key assumptions.

Assumption 1 *Access to the system's state $\boldsymbol{s}$ and a control affine system $\dot{\boldsymbol{s}} = f(\boldsymbol{s}) + G(\boldsymbol{s})\boldsymbol{a}$.*

This assumption appears quite restrictive as we require access to the model of the robot and its state. However, for most tasks considered so far for RFMs, only the kinematic model is required, and therefore, many settings of interest can be described in terms of control-affine systems. Moreover, even if kinematic knowledge is not strictly required by the foundation model, this information is normally used further down the control stack and therefore readily available.

Assumption 2 *The safety conditions can be described as continuously differentiable constraints $\mathbf{0} \geq g(\boldsymbol{x}) \in \mathcal{C}^1$. The constraint function should be known analytically.*

While this assumption is quite strong, it holds in most practical scenarios, particularly when safety is critical. In general, some constraints are trivial to impose, e.g., workspace limits. We show in Section 2.2 how we can easily generate safety constraints for simple visual manipulation tasks. Similar approaches can be used to generate arbitrary constraints, or it could be possible to exploit common-sense knowledge coming from a general-purpose large language model. In general, to generate arbitrary safety constraints, it is necessary to learn them automatically from environment interaction. This problem is an active research topic, and many solutions already exist in the literature [9, 10, 11], including how to deal with constraint uncertainty [12, 13]. However, in this paper, we consider only the setting where constraints are known, leaving more advanced automatic approaches for future work.

2.1 Acting on the Tangent Space of the Constraint Manifold

Under assumptions 1 and 2, we can couple any arbitrary RFM with the ATACOM [5] safety layer. The key idea of ATACOM is to generate a safe action space where we can sample arbitrary actions, while ensuring the satisfaction of the safety constraints. This satisfaction is achieved by constructing the so-called constraint manifold, computing the tangent space at the current robot configuration, and using this tangent space as a safe action space. By taking actions on the tangent space of the constraint manifold, we generate paths moving on the manifold, corresponding to safe trajectories of the robot. Under mild assumptions, this approach ensures theoretical guarantees in terms of forward invariance of the constraint manifold and input-to-state stability [6], ensuring that safety is achieved even under disturbances.

The ATACOM safety layer takes as input an arbitrary action, i.e., the action $\boldsymbol{a}_{\text{RFM}}$ sampled from the RFM, and produces a safe action to apply to the system by constraining the input action when necessary. We refer to Liu et al. [6] for the technical details. At a high level, the ATACOM action can

Figure 2: (left) Spheres cover the robot's hull at critical areas to formulate distance-based constraints ensuring safe executions of the vision-language-actions (VLAs) action predictions. (middle) Bounding boxes of obstacles are generated from 2D instance segmentation and depth information. (right) We calculate the distance between the covering spheres and the obstacle's bounding box by projecting the sphere's center into the bounding box's coordinate frame and estimating the distance to the bounding box's hull.

be decomposed into three components as follows

$$\boldsymbol{a}_{\text{safe}} = \boldsymbol{a}_{\text{drift}}(\boldsymbol{s}) + \boldsymbol{a}_{\text{err}}(\boldsymbol{s}) + \boldsymbol{B}(\boldsymbol{s})\boldsymbol{a}_{\text{RFM}}, \tag{1}$$

where $\boldsymbol{a}_{\text{drift}}(\boldsymbol{s})$ is a state-dependant compensation term that compensates for the change of the constraint function due to the affine component of our system (the "drift" of the system), $\boldsymbol{a}_{\text{err}}(\boldsymbol{s})$ is an additional error correction term that is active only in case of constraint violations (e.g., in case of disturbance), bringing the system back to the safe set. The last component represents the tangent space basis $\boldsymbol{B}(\boldsymbol{s})$ in the current state, having the effect of morphing the action sampled from the RFM to avoid constraint violations. While the ATACOM action space can, in principle, have a different physical meaning w.r.t. the vanilla action space, the $\boldsymbol{B}$ matrix can be chosen such that the morphing retains as much as possible the action semantics. This allows us to combine the safety layer with any RFM without performing a safety-layer specific fine tuning.

Another important issue is that satisfying safety constraints may require a higher control frequency than what is possible with current RFM due to their slower latency compared to simpler robot policies. This requirement is because the safety constraint depends on the robot's state, which may evolve at a faster timescale than the action frequency. However, the computation needed for the ATACOM safety layer can be performed at a much higher frequency, as we only require computing the drift term, the error correction term, and the basis matrix $\boldsymbol{B}$ using the state and the analytical constraints. In practice, we sample the action from the RFM at a fixed rate, e.g., 15Hz, repeating the single action or predicting a series of actions. The action is then applied to the ATACOM layer, where the basis, error correction, and drift compensation terms change at a higher frequency, e.g., 60Hz.

Using the ATACOM framework, we can impose a wide variety of constraints. Several core safety constraints can be defined using only the robot's kinematic model to ensure a general notion of safety. *Joint limit constraints* prevent the robot from exceeding its mechanical bounds while still allowing control near those limits, enabling high flexibility. *Workspace constraints* allow the positioning of cameras and other essential components within the scene. *Self-collision constraints* are implemented by approximating safety-critical parts of the robot with geometric volumes (e.g., spheres) that enclose the mesh. ATACOM efficiently handles a large number of constraints through parallelization of the constraint computations. If the geometry of the obstacles in the environment is available, it is possible to impose complex *collision avoidance constraints*. This capability can be achieved by defining the constraint as a signed distance field (SDF) [14, 15]. However, if it is not necessary to operate with extreme precision around a given obstacle, it is always possible to define a bounding box as constraints around the object. In the next section, we will explain how to easily generate bounding box constraints from visual input.

2.2 Visual Constraint Generation

Manual definition of constraints for the RFM is time-consuming and often requires expert knowledge and environmental information. State-based environments offer all the necessary information out of

the box to specify safety constraints. However, RFMs usually only observe visual information about the environment they operate in order to generalize to arbitrary real-world environments. As such, occlusions can lead to partial observability, and the 2D camera stream needs to be mapped into the 3D task space of the robot to infer and define constraints effectively. While the definition of static workspace constraints and joint constraints is trivial, the definition of constraints for non-static objects in the scene is tedious and non-trivial. Technologies like OptiTrack could be used, but reliance on such specialized hardware would pose a significant limitation and only work in lab environments. As such, we provide an intuitive, cost-effective, and lightweight approach to automatic constraint generation in the visual space.

We leverage instance segmentation in 2D using SAM2 [7] and lift obtained multi-view segmentation masks into 3D using the pinhole camera equation and the camera intrinsics. Based on the 3D instance segmentation, we calculate minimum bounding boxes for each object in the scene. We draw the bounding boxes ourselves to obtain reliable bounding boxes for every evaluation run. This process can be automated in future work by incorporating more grounded segmentation masks, for example, following the approaches in [16, 17].

We calculate the distance between points on the mesh of the robot and the bounding box planes. We parameterize the oriented bounding box (bb) through its center in the world frame ${}^{\text{base}}\boldsymbol{x}_{\text{center}}$, its rotation ${}^{\text{base}}\boldsymbol{R}_{\text{bb}}$, and its extent in the bounding boxes frame ${}^{\text{bb}}\boldsymbol{h}$. To obtain distance estimates between the robot and the segmented objects, we spawn spheres at key robot positions that cover the manipulator's hull (Figure 2). Each sphere is parameterized by its center position and radius $(\boldsymbol{x}, r)$. To ensure safety, ATACOM guarantees that the distance between each sphere's hull and the obstacle's bounding box remains positive. This is done by projecting the sphere's center into the oriented bounding box frame ${}^{\text{bb}}\boldsymbol{x} = {}^{\text{bb}}\boldsymbol{R}_{\text{base}}{}^{\text{base}}\boldsymbol{x} + {}^{\text{bb}}\boldsymbol{x}_{\text{center}}$. We then calculate the distance between the sphere and the closest point on the bounding boxes surface

$$d_{\text{bb}} = ||\alpha^{\text{bb}}\boldsymbol{x} - \boldsymbol{p}||; \quad \boldsymbol{p}_i = \text{clip}(\alpha^{\text{bb}}\boldsymbol{x}_i, -{}^{\text{bb}}\boldsymbol{h}_i/2, {}^{\text{bb}}\boldsymbol{h}_i/2); \quad \alpha = 1 - r/\|{}^{\text{bb}}\boldsymbol{x}\|. \tag{2}$$

Here, $\boldsymbol{p}$ denotes the closest point of the bounding box to the sphere and can be obtained by clipping the sphere's reach onto the bounding box's limits, and α projects the sphere's center to its boundary. With that, we have a simple and effective method to estimate a bounding box constraint $g_{\text{bb}}(\boldsymbol{x}) = -d_{\text{bb}}$ with an analytical gradient estimation.

3 Experiments

We provide several experiments in two different environments to evaluate the behavior of RFMs from a safety perspective. First, we set up a quasi-static pick-and-place environment, most commonly represented in current large-scale training datasets [1, 18]. Here, a Franka Research 3 (FR3) needs to be controlled to accomplish the task of picking and placing various items while not colliding with the workspace or obstacles within the workspace (Figure 4). Next, we use the air hockey environment [19] to showcase the highly dynamic task of puck hitting with a Kuka LBR IIWA 14, which comes with its own safety challenges.

3.1 Quasi-Static Pick-and-Place Environment

We evaluate the effectiveness of our proposed safety layer with visual constraints (Section 2.2) on pick-and-place tasks on a Franka FR3 platform. In these tasks, the RFM is prompted to *"grab the [object] and put it into the box"* where *object* is taken from the set {`apple, strawberry, banana, pear, tennis ball, baseball`}. As such, the goal is to safely grab the desired object from the table surface and place it into the box. We evaluate three versions of this task with varying task complexity: `pick-fruit-easy`, `pick-ball-medium`, `pick-object-hard` (see Appendix B.3 for more task details. This common pick-and-place task represents various potential safety hazards, e.g., the robot can crash into the table while picking up the object, it can collide with obstacles in the scene, or attempt to leave the workspace.

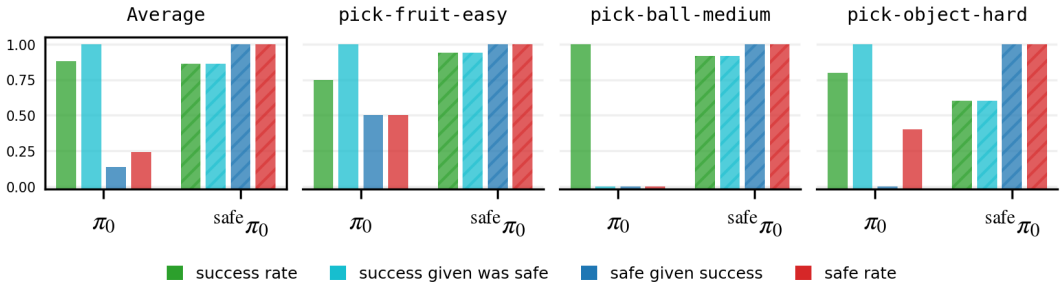

Figure 3: Results in the manipulation tasks. Dashed histograms indicate the RFM combined with the ATACOM safety layers, while the solid ones represent the vanilla π_0 model. We report success rate, success rate for safe trajectories, percentage of safe trajectories among the successful ones, and normalized execution time. Results show that the safety layer does not impact heavily the success rate, while ensuring safety.

We fine-tune π_0 [4] on expert demonstrations of our above pick-and-place tasks to adapt it to the new environment. We collect 300 expert demonstrations via teleoperation following the data collection protocol from [18]. During data collection and evaluation, we randomize the position and orientation of the fruits and the box. Further, we add and remove various *distractor objects* within the scene (e.g., different boxes, cans, and bottles). We outline further details on the setup in Appendix B.

The results for all of these experiments are presented in Figure 3. First, we evaluate each task's success rate, disregarding any safety considerations. Here, the success rate is evaluated visually, given the simplicity of the considered tasks. In general, our safety layer does not affect the performance heavily in terms of success rate. On average, the success rate is similar to that of the vanilla policy. Curiously, the safety layer increases the success rate in the pick-fruit-easy task. This counterintuitive result is because the safety constraint prevents collision with the table, allowing for a better grasp alignment. In two other tasks, the success rate is slightly lower, and it is particularly evident in the pick-object-hard task. This is reasonable, as imposing complex safety constraints makes the task execution harder.

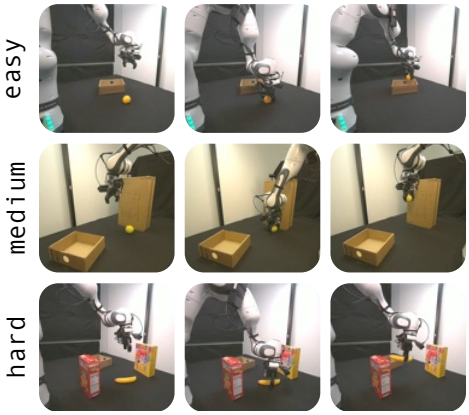

Figure 4: Video frame extracts from a rollout on three different tasks. Difficulty is dictated by the number of obstacles in the scene.

To prove the benefits in terms of safety, we evaluate the task in terms of the success rate of safe trajectories (i.e., the ratio of safe trajectories that complete the task) and the safety rate of successful trajectories (i.e., the ratio of successful trajectories that are also safe). Results clearly show that our approach always ensures safe trajectories, while vanilla π_0 presents many constraint violations: in the pick-fruit-easy task, the model frequently collides with the table, whereas in the other two tasks, although the objectives are achieved, the robot often knocks over obstacles during execution.

We also evaluate the performance in terms of execution time. The results are reported in Table 1. Specifically, we measure the execution time of successful trajectories. As expected, on average, the trajectories with the safety layer are slightly longer, particularly in the pick-ball-medium task. Again, we observe some performance gain in the pick-fruit-easy task, due to the lack of collisions with the table, which may cause the robot's end effector to drift, making the task harder for the unsafe policy.

	π_0	$^{\text{safe}}\pi_0$
pick-fruit-easy	26s	24s
pick-ball-medium	23s	30s
pick-object-hard	22s	22s
Average	23s	25s

Table 1: Average time to task completion. Safe execution sometimes prolongs task duration.

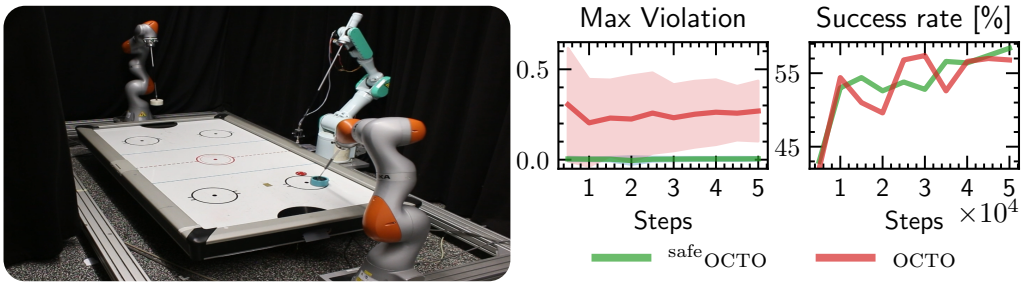

Figure 5: Safety violations of the OCTO policy w/o the safety module on the air hockey hitting task for different checkpoints during the training phase. We report the maximum constraint violation and the success rate of the robot hitting the puck into the goal over 500 episodes in simulation. When the ATACOM safety module is added, the policy remains compliant with safety constraints throughout fine-tuning, whereas the unmodified OCTO policy continues to breach safety limits. Both policies progressively improve their success rates over the number of fine-tuning steps.

3.2 Dynamic Air Hockey Environment

We empirically evaluate the proposed approach on a robot air hockey task. The objective is to hit a puck into the goal while adhering to multiple safety constraints, such as keeping the end-effector on the table surface, preventing the arm from colliding with the table, and ensuring joint position limits. We refer to [6] for a detailed description of the experimental setup. The policy's observation consists of language instructions, a goal image of the scene, and proprioceptive data in the form of joint positions, joint velocities, puck position, and puck velocity. While not needed for safety, we fine-tune a pre-trained OCTO [3] policy using behavior cloning in a simulated MUJOCO [20] environment. Importantly, we obtain the fine-tuning data by an expert policy that does not leverage ATACOM. The policy outputs desired end-effector velocities in the x-y plane of the table surface, which are converted to joint velocities using inverse kinematics. The ATACOM layer then maps these joint velocities to safe ones before passing them to a joint-space controller. We compare our safety-aware approach to an unsafe baseline, where the joint-space controller directly executes the unfiltered joint velocities. We evaluate the safety module for various fine-tuning checkpoints of OCTO on the simulated system. Several deployment videos of OCTO playing air hockey can be found on our project page.

Our experimental results, presented in Figure 5, show that the OCTO agent with the added safety module does not violate the safety constraints at deployment time. On the contrary, OCTO without the added safety layer heavily violates the constraints, even though the fine-tuning data contains safe expert demonstrations. Looking at the success rate, it is clear that using the safety module does not affect performance, as both approaches show similar behavior during the training phase. Importantly, while the fine-tuning data is not obtained with ATACOM, we still obtain high success rates, suggesting that ATACOM does not generate overly conservative control actions.

Finally, we deploy the safe policy in the real-world system. While the real-world deployment is affected by the sim-to-real gap, we still achieve reasonable performance while strictly enforcing the safety constraints. This shows that the approach is viable even in dynamic tasks and using nominal constraints, demonstrating the robustness of our approach against modeling errors.

4 Related Work

This work considers ensuring safety during the deployment of robotic foundation models, which has been identified as an important open problem in the development of these policies [21]. As a robotic foundation model is abstractly like any other robot policy, albeit more capable, many existing safe control techniques can be directly applied. In this work, we use ATACOM [5, 6], which augments the policy architecture to ensure safety in the action space. Zhang et al. [8] finetune a foundation policy

using safe reinforcement learning techniques, which augment the objective with the constraints and Lagrangian multipliers. Since safety is incorporated as an auxiliary objective, there is no guarantee that a constraint is obeyed at runtime and instead the policy pays a penalty for constraint violation. Moreover, since a sample-inefficient on-policy RL method is used, the policy is trained and evaluated in simulation rather than the real world.

For complex multimodal policies, safety can also be interpreted as the performance of the policy to natural nonstationary perturbations of the real world during deployment. Majumdar et al. [22] approach safety from this robustness direction, and use generative models to achieve adversarial data augmentation to make the policy robust to sensing environmental factors like lighting; however, it does not adapt its behavior to dynamic environments. Hancock et al. [23] use gradient information between the action and observations to augment the training data to alleviate the policy's sensitivity to the input space, as a proxy for robustness. This line of research also extends to language conditioning, and ensuring safety via robustness to jailbreaking the language model component [24].

A separate line of work uses foundation models to ensure safety, using their real-world grounding to incorporate notions of 'semantic' safety. Brunke et al. [25] use an large Language model (LLM) from semantic task descriptions to generate state and action constraints that are then enforced using control barrier function and safety filters. This approach is validated for actions generated from teleoperation and also diffusion policies. Ling et al. [26] use a VLA to generate a cost map for motion planning, where the cost map automatically incorporates acceptable tolerances for practical obstacle avoidance. This is relevant for navigating complex cluttered scenes where certain objects, e.g., brittle and fragile, require greater care over objects that are soft or rugged.

5 Discussion

We propose a safety module that can be added as the final layer of an robot foundation model (RFM) by leveraging domain-specific knowledge of the safe operation constraints. We have shown that this layer can be added to a variety of existing robot foundation model (RFM) architectures across a range of tasks, and ensure consistently safe execution with only small potential impacts to task performance. We demonstrate the effectiveness of the safety layer by evaluating a vision-language-action (VLA) policy with BC on an air hockey hitting task for which it is critical not to crash with the tabletop, and object manipulation tasks with obstacles. We believe this architecture extension is necessary and crucial in the application of RFMs for everyday tasks and real-world deployment.

Leveraging domain knowledge may seem counterintuitive for RFMs, as they show that rich behaviors can be obtained purely through large-scale datasets rather than handcrafted policies. We believe that safety is inherently contextual information for a given task and robot configuration. However, data-driven approaches leveraging this contextual cue are undesirable, as they would require collecting unsafe demonstrations in order for the policy to learn the difference between unsafe and safe behaviour. Therefore, we believe inductive biases are a more practical means of incorporating this contextual information, using minimal computational adjustments such as our proposed safety layer.

While we emphasize that ensuring safety requires domain expertise, it can also be a demanding task to manually formulate all necessary safety constraints for a given task. One intuitive research direction is to automate the process by leveraging the inherent knowledge of vision-language models (VLMs). However, so far, VLMs have only been used to integrate semantic safety constraints such as 'keep the cup uprigh' into an already existing set of constraints [27, 25]. Beyond the formulation of safety constraints, it remains an open research question of how a more generalizable concept of safety can be formulated and applied across different embodiments, environments, and tasks. One avenue, based on our safety layer, is to extend the 'code-as-policies' paradigm [28] to a 'code-as-safety' approach, using LLMs to automate the design of future safety layers.

Acknowledgments

This work was supported by "Third Wave of AI", funded by the Excellence Program of the Hessian Ministry of Higher Education, Science, Research and Art. We also acknowledge the grant "Einrichtung eines Labors des Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) an der Technischen Universität Darmstadt" of the Hessian Ministry of Science and Research, Arts and Culture.

References

[1] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. In *International Conference on Robotics and Automation (ICRA)*, 2024.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *Conference on Robot Learning (CoRL)*, 2024.

[3] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024.

[5] P. Liu, D. Tateo, H. B. Ammar, and J. Peters. Robot reinforcement learning on the constraint manifold. In *5th Conference on Robot Learning (CoRL)*. PMLR, 2021.

[6] P. Liu, H. Bou-Ammar, J. Peters, and D. Tateo. Safe reinforcement learning on the constraint manifold: Theory and applications. *arXiv preprint arXiv:2404.09080*, 2024.

[7] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollar, and C. Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025.

[8] B. Zhang, Y. Zhang, J. Ji, Y. Lei, J. Dai, Y. Chen, and Y. Yang. SafeVLA: Towards safety alignment of vision-language-action model via safe reinforcement learning. *arXiv preprint arXiv:2503.03480*, 2025.

[9] A. Taylor, A. Singletary, Y. Yue, and A. Ames. Learning for safety-critical control with control barrier functions. In *Learning for Dynamics and Control*, pages 708–717. PMLR, 2020.

[10] D. C. Tan, F. Acero, R. McCarthy, D. Kanoulas, and Z. A. Li. Your value function is a control barrier function: Verification of learned policies using control theory. *arXiv preprint arXiv:2306.04026*, 2023.

[11] Y. Yang, Y. Jiang, Y. Liu, J. Chen, and S. E. Li. Model-free safe reinforcement learning through neural barrier certificate. *IEEE Robotics and Automation Letters*, 8(3):1295–1302, 2023.

[12] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan. Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35 (12), pages 10639–10646, 2021.

[13] J. Günster, P. Liu, J. Peters, and D. Tateo. Handling long-term safety and uncertainty in safe reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2024.

[14] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki. Regularized deep signed distance fields for reactive motion generation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6673–6680. IEEE, 2022.

[15] P. Liu, K. Zhang, D. Tateo, S. Jauhri, Z. Hu, J. Peters, and G. Chalvatzaki. Safe reinforcement learning of dynamic high-dimensional robotic tasks: navigation, manipulation, interaction. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9449–9456. IEEE, 2023.

[16] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision (ECCV)*, 2024.

[17] T. Ren, Q. Jiang, S. Liu, Z. Zeng, W. Liu, H. Gao, H. Huang, Z. Ma, X. Jiang, Y. Chen, Y. Xiong, H. Zhang, F. Li, P. Tang, K. Yu, and L. Zhang. Grounding dino 1.5: Advance the "edge" of open-set object detection, 2024.

[18] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, rAchit Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.

[19] P. Liu, J. Günster, N. Funk, S. Gröger, D. Chen, H. Bou-Ammar, J. Jankowski, A. Marić, S. Calinon, A. Orsula, M. Olivares-Mendez, H. Zhou, R. Lioutikov, G. Neumann, A. Likmeta, A. Zhalehmehrabi, T. Bonenfant, M. Restelli, D. Tateo, Z. Liu, and J. Peters. A retrospective on the robot air hockey challenge: Benchmarking robust, reliable, and safe learning techniques for real-world robotics. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, pages 9690–9726, 2024.

[20] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

[21] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 2023.

[22] A. Majumdar, M. Sharma, D. Kalashnikov, S. Singh, P. Sermanet, and V. Sindhwani. Predictive red teaming: Breaking policies without breaking robots. *arXiv preprint arXiv:2502.06575*, 2025.

[23] A. J. Hancock, A. Z. Ren, and A. Majumdar. Run-time observation interventions make vision-language-action models more visually robust. *arXiv preprint arXiv:2410.01971*, 2024.

[24] Z. Ravichandran, A. Robey, V. Kumar, G. J. Pappas, and H. Hassani. Safety guardrails for LLM-Enabled robots. *arXiv preprint arXiv:2503.07885*, 2025.

[25] L. Brunke, Y. Zhang, R. Römer, J. Naimer, N. Staykov, S. Zhou, and A. P. Schoellig. Semantically safe robot manipulation: From semantic scene understanding to motion safeguards. *IEEE Robotics and Automation Letters*, 2025.

[26] Y. Ling, K. Owalekar, O. Adesanya, E. Bıyık, and D. Seita. IMPACT: Intelligent motion planning with acceptable contact trajectories via vision-language models. *arXiv preprint arXiv:2503.10110*, 2025.

[27] L. Santos, Z. Li, L. Peters, S. Bansal, and A. Bajcsy. Updating robot safety representations online from natural language feedback, 2024.

[28] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[29] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015.

[30] Y. Lin, A. S. Wang, G. Sutanto, A. Rai, and F. Meier. Polymetis. https://facebookresearch.github.io/fairo/polymetis/, 2021.

[31] T. Schneider. Franky: High-level control library for franka robots with python and c++ support, 2025. URL https://github.com/TimSchneider42/franky.

[32] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, and T. Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

A Limitations

The key limitation of our approach is that we assume the safety constraints as known. This may limit the applicability of the approach to open-world tasks, where it is not feasible to specify all the safety constraints a priori. However, our basic pipeline could be in principle extended with a simple object detection pipeline. Furthermore, it could be in principle possible to exploit the capabilities of LLMs to generate a set of reasonable set of safety constraints for a given task. This generation can be carried out offline, and therefore would not impact the execution time of the RFM policy. Another limitation of this paper is the limited evaluation of the policy to solve general tasks. While this is a very important topic, our focus is mostly on guaranteeing the safe execution of the policy. We believe that future models will generalize better across tasks and perform smoother and faster manipulation movements. The last key limitation is that the setup requires multiple depth cameras to work properly and generate appropriate bounding boxes directly from perception. While this limits the applicability to general scenarios, many computer vision pipelines can generate robust bounding boxes, exploiting a moving camera. Furthermore, it may be necessary only to generate accurate bounding boxes from the side of the object perceived by the robot. For simplicity of the experimental setup, we left complex perception problems to future work.

B Experimental Details: Pick-and-Place Tasks with Franka Robot

We perform our vision-based pick-and-place experiments using a Franka Research 3 with a RH-P12-RN Robotis gripper, three Zed X mini cameras for scene capture, and a Meta Quest 3 with a single remote controller for data collection. We follow the hardware setup of DROID [18] and capture the scene with two external cameras to the left and right of the Franka robot and one gripper camera. The items to pick are selected from the classic YCB Benchmarks – Object and Model Set [29].

B.1 Expert Data Collection & Fine-Tuning

We adapt the open-source DROID pipeline [18]. Amongst others, we replace Polymetis [30] with Franky [31] to control our Franka robot. We collect over 300 trajectories for our experiments using the Meta Quest for teleoperation and showcasing pick-and-place behavior with different items in cluttered scenes. Importantly, the data is collected without ATACOM [5, 6], but showcases safe behavior. We convert the collected data into a LeRobot [32] dataset and perform full finetuning of the π_0 base model [4] on a single A100 GPU for 10.000 training steps with a batch size of 32.

While we perform our experiments in joint space control, we collect data in Cartesian velocity space using the Meta Quest remote controller. We extract the necessary joint velocity information from the robot state to train the π_0 base model [4]. Although joint velocity state and desired action values are expected to differ slightly, in practice, our finetuned joint velocity π_0 model performed well regardless.

B.2 Pick-and-Place Franka Safety Layer

We make the fine-tuned π_0 model safe by providing its joint velocity action output to our safety layer. Our safety layer is defined by several inequality constraints that define the constraint manifold. ATACOM then generates safe actions by staying on the constraint manifold. For the workspace and bounding box constraints, the robot's hull gets approximated by several spheres s_i as displayed in Figure 2. Each sphere is parameterized by its center position and its radius $({}^{\text{base}}\boldsymbol{x}, r)$. The workspace constraints enforce that each sphere remains within the prescribed lower bounds ${}^{\text{base}}\boldsymbol{x}_{\text{min}}$ and upper bounds ${}^{\text{base}}\boldsymbol{x}_{\text{max}}$ of the workspace. In particular, for each sphere s_i we require

$$\text{(Workspace)} \qquad g_{s_i}(\boldsymbol{x}) = {}^{\text{base}}\boldsymbol{x}_{\text{min}} - {}^{\text{base}}\boldsymbol{x} + r_i \leq 0 \tag{3}$$

$$g_{s_i}(\boldsymbol{x}) = {}^{\text{base}}\boldsymbol{x} - {}^{\text{base}}\boldsymbol{x}_{\text{max}} + r_i \leq 0. \tag{4}$$

For the bounding box constraints, we calculate the distance between each sphere s_i and the closest point on the bounding boxes surface

$$\text{(Bounding Box)} \qquad \begin{aligned} g_{s_i}(\boldsymbol{x}) &= -||\alpha^{\text{bb}}\boldsymbol{x} - \boldsymbol{p}|| \leq 0 \\ \boldsymbol{p}_j &= \text{clip}(\alpha^{\text{bb}}\boldsymbol{x}, -{}^{\text{bb}}\boldsymbol{h}_j/2, {}^{\text{bb}}\boldsymbol{h}_j/2) \\ \alpha &= 1 - r/\|{}^{\text{bb}}\boldsymbol{x}\|, \end{aligned} \tag{5}$$

where ${}^{\text{bb}}\boldsymbol{x}$ is the sphere's center position in the bounding box (bb) frame and ${}^{\text{bb}}\boldsymbol{h}$ corresponds to the extent of the bounding box. A third constraint category keeps the joints $\boldsymbol{q}$ within their limits $\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}$

$$\text{(Joint Limits)} \qquad \begin{aligned} g(\boldsymbol{x}) &= ((\boldsymbol{q} - \bar{\boldsymbol{q}})^2/\Delta q^2) - 1 \leq 0 \\ \bar{\boldsymbol{q}} &= (\boldsymbol{q}_{\max} + \boldsymbol{q}_{\min})/2 \\ \Delta\boldsymbol{q} &= (\boldsymbol{q}_{\max} - \boldsymbol{q}_{\min})/2. \end{aligned} \tag{6}$$

B.3 Evaluation Task Descriptions

We evaluate our Franka pick-and-place safety layer on three evaluation tasks which we further describe in the following paragraphs. Selected hyperparameters of π_0 and ATACOM can be found in Table 2.

pick-fruit-easy. This task requires picking a specified (plastic) fruit off the table and placing it in the box. We consider the task to be successful when the robot manages to place the specified fruit into the box within the maximum episode length of 1000 timesteps. The potential danger is to collide with the table surface, especially for smaller fruits like the strawberry.

pick-ball-medium. This task requires picking up a tennis ball off the table and placing it within the box. The ball is placed close ($\sim$ 10cm) from a large cardboard box, which serves as an obstacle that the robot needs to avoid during the pickup. The robot needs to avoid collisions with the obstacle as well as the table simultaneously.

pick-object-hard. This task requires the robot to pick up a specified object off the table and place it in the box. We add up to 3 obstacles into the workspace that the robot needs to avoid during operation.

Table 2: Parameter Selection for the Pick-and-Place Tasks

Hyperparameter	Value
π_0	
Model	Base
Control frequency	15hz
Action chunk size	32
ATACOM	
Control frequency	60 hz
Slack function	exp
Slack beta	10
Slack tolerance	0.001
Drift clipping	True

C Experimental Details: Pushing Task on Air Hockey Table

We perform our AirHockey experiments in the MUJOCO [20] simulator with a fine-tuned OCTO [3] model. The input to the model consists of the tuple $\boldsymbol{x} = (\boldsymbol{o}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{p}, \dot{\boldsymbol{p}}, \boldsymbol{g}_{\text{img}})$, which comprises a third person visual observation $\boldsymbol{o}$, joint positions $\boldsymbol{q} \in \mathbb{R}^7$ and the joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^7$, the puck's horizontal and rotational position $\boldsymbol{p} = [x, y, \theta]^\intercal$ as well as velocity $\dot{\boldsymbol{p}}$, and a task to achieve, given as goal image $\boldsymbol{g}_{\text{img}}$. We fine-tune OCTO [3] to predict desired Cartesian velocity action chunks in the 2D Cartesian space of the mallet end-effector on the AirHockey table. The model gets evaluated on the task of hitting a randomly initialized puck into the goal on the other side of the AirHockey table.

C.1 Expert Data Collection and Fine-Tuning

We collect our fine-tuning data with an expert policy, which does not leverage ATACOM [5, 6]. Data is collected independently of our safety layer, as we do not need fine-tuning data to become safe, but to increase the performance of the policy in the new environment. We spawn the puck randomly in

a rectangle in front of the robot with a zero initial velocity. The expert policy generates trajectories using a classical planning approach to hit the puck into the goal on the other side of the table. In total, we collect 500 trajectories for fine-tuning. We fine-tune OCTO on the collected data for 50.000 gradient steps using a batch size of 256. As we provide additional puck state information, we do not need to deal with partial observability and choose an observation history of zero. Initial evaluations have shown that we obtain the best puck-hitting results with an action chunk size of 16.

C.2 AirHockey Safety Layer

We guarantee safe deployment of the fine-tuned OCTO [3] model by feeding predicted 2D Cartesian actions into our ATACOM safety layer. As our safety layer works in joint space, we first convert the Cartesian velocities into joint velocities using inverse kinematics. ATACOM morphs these joint velocities into safe ones by acting on the tangent space of the constraint manifold. Several inequality constraints define the constraint manifold. Our table surface constraints

$$\text{(Table surface)} \qquad g_1(\boldsymbol{x}) = -z_{\text{ee}} + z_{\text{low}} \leq 0 \tag{7}$$
$$g_2(\boldsymbol{x}) = z_{\text{ee}} - z_{\text{high}} \leq 0 \tag{8}$$

ensure that the attached mallet stays within the table height bounds $z_{\text{low}}, z_{\text{high}}$. With the additional link constraints

$$\text{(Link constraints)} \qquad g_3(\boldsymbol{x}) = -x_{\text{ee}} + x_{\text{low}} \leq 0 \tag{9}$$
$$g_4(\boldsymbol{x}) = -y_{\text{ee}} + y_{\text{low}} \leq 0 \tag{10}$$
$$g_5(\boldsymbol{x}) = y_{\text{ee}} - y_{\text{high}} \leq 0 \tag{11}$$
$$g_6(\boldsymbol{x}) = -z_{\text{wrist}} + z_{\text{wrist}_{\text{low}}} \leq 0 \tag{12}$$
$$g_7(\boldsymbol{x}) = -z_{\text{elbow}} + z_{\text{elbow}_{\text{low}}} \leq 0, \tag{13}$$

we keep the mallet within the table bounds $x_{\text{low}}, y_{\text{low}}, y_{\text{high}}$ as well as the wrist and elbow above lower height bounds $z_{\text{wrist}_{\text{low}}}, z_{\text{elbow}_{\text{low}}}$. The last constraint category keeps the joints $\boldsymbol{q}$ within their limits $\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}$

$$\text{(Joint limits)} \qquad g_{8\ldots14}(\boldsymbol{x}) = ((\boldsymbol{q} - \bar{\boldsymbol{q}})^2/\Delta q^2) - 1 \leq 0 \tag{14}$$
$$\bar{\boldsymbol{q}} = (\boldsymbol{q}_{\max} + \boldsymbol{q}_{\min})/2$$
$$\Delta \boldsymbol{q} = (\boldsymbol{q}_{\max} - \boldsymbol{q}_{\min})/2.$$

C.3 Evaluation & Real-World Deployment

We evaluate every OCTO [3] fine-tuning checkpoint for 500 episodes in our MUJOCO [20] environment. Important hyperparameters of our evaluation can be found in Table 3.

After evaluation in simulation, we deployed $^{\text{safe}}$OCTO on our real-world AirHockey setup. OCTO, without our safety layer, was not secure enough to be deployed in real world. Amongst others is the risk that OCTO strongly hits the AirHockey table while performing the task. As the policy was trained in simulation with additional puck state information, we use the OptiTrack system to track the puck's position and velocity. Using proprioception and puck state information, we reconstruct the real-world state in MUJOCO [20] to obtain the simulated visual input observation. Videos of $^{\text{safe}}$OCTO playing AirHockey in the real world can be found on our project page.

Table 3: Parameter Selection for the AirHockey Pushing Task

Hyperparameter	Value
OCTO	
Model	octo-small-1.5
Control frequency	12.5 hz
Action chunk size	16
ATACOM	
Control frequency	50 hz
Slack function	exp
Slack beta	2
Slack tolerance	$1e^{-6}$
Drift clipping	True