



UPPSALA
UNIVERSITET

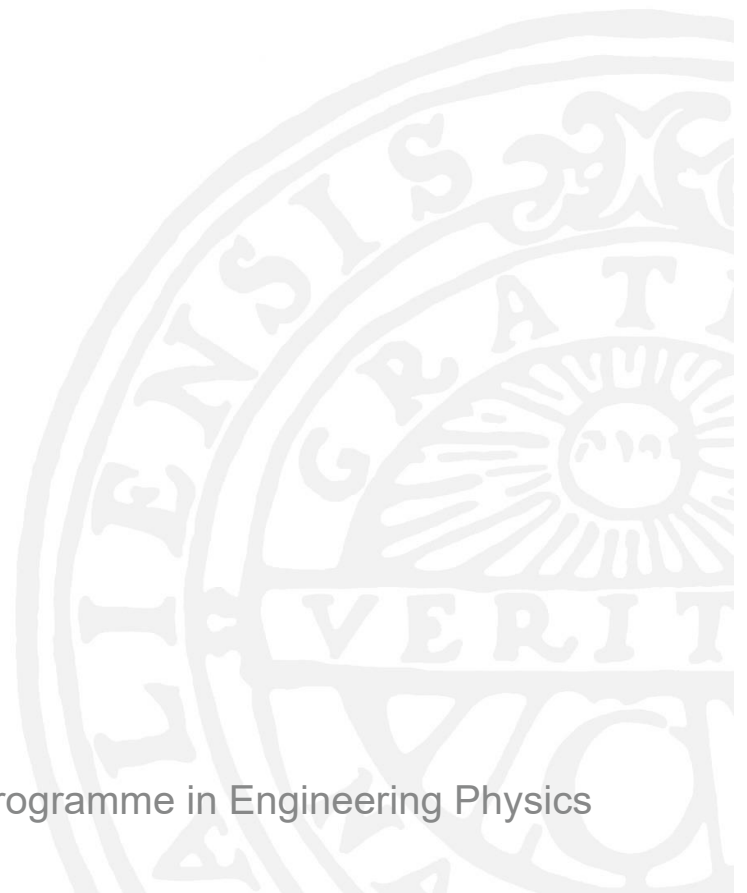
UPTEC F 25028

Degree project 30 credits

June 2025

Combining foundation models and numerical solvers for physics-informed motion control

Anna Johansson



Master's Programme in Engineering Physics



UPPSALA
UNIVERSITET

Combining foundation models and numerical solvers for
physics-informed motion control

Anna Johansson

Abstract

Multi-modal foundation models for robotics control is a rapidly evolving field. Models are trained on large amounts of robot trajectory data and, in some cases, additional web data. This approach to robot control allows for control policies that are capable of reasoning and that have the capability of performing previously unseen tasks in unseen environments. However, these models have no safety guarantee built into their motion planning. As a contribution towards increasing the safety of these models, we propose a novel scheme for capturing unsafe trajectories before they are executed, called Sparse Evaluation. This method integrates a numerical physics solver with the foundation model, to screen proposed actions and assert their safety before real-life execution. In this thesis paper, we provide a small proof-of-concept for this scheme, and the results indicate that it is possible to use this method to detect if an action will be carried out as expected or not, depending on some parameters in the system. We also provide a brief survey of existing models for robot control with foundation models.

Faculty of Science and Technology

Uppsala University, Place of publication: Uppsala

Supervisor: Marin Servin Subject reader: Per Isaksson

Examiner: Tomas Nyberg

Populärvetenskaplig sammanfattning

AI-modeller blir en allt större del av vår vardag, särskilt de så kallade *grundmodellerna*. Det är en typ av artificiell intelligens som tränas på enorma mängder data och som har förmågan att utföra många olika uppgifter. En vanlig form av grundmodell är så kallade *Large Language Models* (LLM), alltså stora språkmodeller. De ligger till grund för chattbotar, som till exempel de digitala assistenterna vi har i våra mobiltelefoner. En annan typ av grundmodell är *Vision-Language Models* (VLM), som kombinerar bild- och textförståelse. Dessa används bland annat när vi gör bildsökningar på nätet.

En viktig egenskap hos grundmodeller är deras förmåga att förstå sammanhang och resonera logiskt. Det har också gjort dem intressanta inom robotik, där AI nu används för att styra robotar på ett mer flexibelt och intelligent sätt. Sedan 2023 har det utvecklats flera grundmodeller särskilt anpassade för robotstyrning, och en särskild sorts modell som är anpassad för detta ändamål är *Vision-Language-Action Models* (VLA), vilka bygger på VLM-modeller. De kan ta emot instruktioner i form av text, analysera sin omgivning med hjälp av kameror och direkt utföra uppgifterna med en fysisk robot. Vissa av dessa modeller klarar till och med relativt avancerade uppdrag som att städa ett bord och skilja mellan vad som är skräp och vad som ska sparas, till exempel att de ska kasta ett papper men lämna kvar en tallrik.

Grundmodeller har dock en tendens att kunna hitta på saker eller svara fel på frågor, och när man låter en AI styra en fysisk robot kan liknande fel bli katastrofala. Robotar som utför osäkra handlingar kan både skada sig själva och sin omgivning, på ett sätt som till exempel en digital assistent aldrig skulle kunna göra. Det kan dessutom vara väldigt svårt att förutsäga allting om omvärlden bara genom visuella intryck, vilket är vad dessa modeller gör. Ett objekt kan ju till exempel vara tyngre än väntat, eller att friktionen är lägre eller högre än vad modellen förutsätter. I vissa komplicerade situationer kan också vissa rörelser ge oönskade konsekvenser som modellen kanske inte tar hänsyn till. En robot som ska flytta på en sten i en hög av stenar kan till exempel råka göra att hela högen kollapsar, och en robot som håller i en kedja skulle kunna orsaka en farlig snärt ifall den accelererade för snabbt.

Än så länge har säkerhetsaspekten vid robotstyrning med AI varit relativt outforskad, men i det här examensarbetet har vi skapat en prototyp av en metod som vi tror skulle kunna lösa det problemet. Vi kallar metoden *Sparse Evaluation*, eller gles evaluering, och den bygger på att man kontrollerar alla handlingar som en AI har planerat att utföra med hjälp av en verklighetstrogen fysikmotor, innan man låter AI:n utföra dem med den fysiska roboten.

En fysikmotor är ett datorprogram som används för att simulera verkliga rörelser och krafter i en digital miljö. Den räknar ut hur objekt påverkas av till exempel gravitation, kollisioner och friktion. Detta görs genom matematiska beräkningar som baseras på verklig fysik, och med en nog exakt fysikmotor kan man alltså förutspå exakt vilka konsekvenser en viss handling kommer få i verkliga livet.

Vår metod består av tre steg. I det första steget låter vi en robot utföra

hela den planerade handlingen i en digital kopia av den verkliga miljön, en så kallad digital tvilling, som skapats med hjälp av fysikmotorn. Simuleringen bygger på antaganden om de osäkra parametrarna i systemet, såsom vikten på de olika objekten.

I steg två identifieras några "kritiska ögonblick" som kan ses som representativa för hela händelseförloppet. I dessa kritiska ögonblick varierar de osäkra parametrarna i systemet, och simuleringen får gå framåt ett kort ögonblick. Genom att göra detta för ett stort antal olika värden på parametrarna, går det att få en uppfattning om ifall handlingen kan genomföras på ett säkert sätt även om till exempel ett objekt skulle väga mer än vad AI:n tror. Anledningen till att evalueringen bara görs i några få, kritiska, ögonblick är för att det krävs ett stort antal beräkningar för att simulera hela händelseförloppet, vilket till slut blir väldigt tidskrävande om det måste göras för alla möjliga värden.

Steg tre går sedan ut på att använda denna information. Ifall den planerade handlingen verkar vara säker, baserat på resultatet från den glesa evalueringen, kan man låta AI:n utföra handlingen med den verkliga roboten. Annars kan man till exempel ge AI:n en lämpligare instruktion. Ifall den ursprungliga instruktionen var "lyft bort stenen", men den glesa evalueringen verkar indikera att roboten är för svag, skulle man istället kunna ge AI:n instruktionen "putta bort stenen".

I det här examensarbetet har vi tagit fram en första prototyp av metoden. Även om den fortfarande är i ett tidigt skede visar våra tester att den kan bli ett värdefullt verktyg för att i förväg upptäcka riskfyllda situationer som en robot kan orsaka, och därmed ge möjlighet att undvika dem innan de sker i verkligheten.

Contents

1	Introduction	6
2	Robot Control with Foundation models	7
2.1	Background	7
2.2	PaLM-E	9
2.3	RT-2	9
2.4	Octo	10
2.5	OpenVLA	10
2.6	CoT-VLA	11
2.7	π_0	11
2.8	$\pi_{0.5}$	12
2.9	Safe Control	12
2.10	Discussing the Future of Embodied AI	13
3	Sparse Evaluation	15
3.1	Method Description	15
3.2	AGX Dynamics	16
3.2.1	Multi-body Dynamics	16
3.2.2	Time integration	17
3.2.3	MCP	18
3.3	ACT	18
3.4	Modeling	19
3.5	Implementation	20
4	Results	23
5	Discussion	25
5.1	Interpreting the Results	25
5.2	Limitations and Future Work	26
5.3	Comparing to Earlier Work	27
6	Conclusion	27
	References	29
A	Results Frequent Evaluation	33

1 Introduction

Embodied AI and other autonomous control is a growing area of interest in the Robotics Community. One particular area of interest is the integration of robotics and foundation models, a class of models trained on large-scale datasets [1]. This could, in theory, imbue an embodied AI with the same kind of reasoning abilities and generalization capabilities to new situations as what popular Large Language Models like Chat-GPT and Gemini possess. One important advantage a foundation model would have over more classical approaches to robot control using Machine Learning (ML), like imitation learning, is that these models have the advantage of being broadly applicable in different situations without specialized training data [1]. This is particularly advantageous for autonomous heavy machinery, which is often produced in low quantities, and each machine is used in wide range of specialized applications [2]. One promising approach is the Vision-Language-Action model (VLA), which is an emerging model capable of processing multimodal inputs and outputting robot actions [3].

Because of the nature of foundations models, huge amounts of data are needed for training. In the robotics community, this has been a challenge since data from this field have not been readily available [2]. To fill this vacuum, the team behind the *X-Embodiment* data set have made great efforts to create a huge amount of annotated data of robots performing varying tasks [4]. With the help of this and other available data, foundation models capable of completing complex and unseen tasks have already been trained [5–8].

However, safety is a concern when using foundation models for autonomous machinery, as erroneous decisions can lead to heavy damages in many use cases, such as when controlling heavy machinery or performing tasks close to humans. It is a well-documented fact that foundation models like Large Language Models, and large Vision-Language Models, are prone to hallucination [9–11]. This can, in extension, cause models for autonomous control built on these foundation models to hallucinate as well [12]. While there are no papers showcasing this phenomenon for robotics control specifically that we are aware of, this is naturally a source of concern for the future of robotics control with foundation models [1].

Another concern is that since ML models generally are not based in physics, it is possible for them to make decisions that do not abide the laws of physics in their environments, both by hallucination and from lack of physics knowledge. In addition, it is not always possible to infer everything about the world only from visual inputs, which is what many existing models for embodied AI do. An object might for example be heavier or lighter than expected, and this might cause damages to the machine or cause other unwanted effects if the machine tries to lift it. In complex scenarios with many parameters, like if an autonomous robot is to manipulate a pile of stones or a chain with unpredictable motions, certain actions can have disastrous and dangerous consequences. The wrong move when manipulating a pile of stones could, for example, lead to a rock slide, and accelerating a chain too fast might lead to

dangerous whipping effects. To mitigate these concerns, a viable approach is to combine ML with a physics-based model. In this thesis paper, we provide a proof-of-concept for a novel scheme that evaluates plans generated by an ML model. We evaluate the safety of performing the model-generated task by simulating it in a physics engine at critical moments, and account for uncertainties such as the mass of objects. In addition to this, we present a small survey of existing and current state-of-the-art foundation models for robot control.

2 Robot Control with Foundation models

This section provides a brief survey of the applications of foundation models in robotics control. It begins with an overview of more general foundation models which lay the basis for the use of foundation models in this area, followed by a discussion of a few representative foundation models tailored for this domain. In this section, we also touch on the concept of fine-tuning and its effects, as well as provide a brief discussion on the future of foundation models.

2.1 Background

The term *foundation model* as introduced by [13] encompasses any machine learning model trained on broad data, and that can be adapted to a wide range of tasks. They usually rely on huge sets of training data, and thus most utilize self-supervision for training, which erases the need for annotated data. Many foundation models make use of some kind of transformer architecture, which makes it possible to capture global dependencies between tokens, making the model effective for inferring context [14]. The implementation of foundation models have already had a huge impact with fields like large language modeling (e.g. GPT- n and LLAMA). Simply put, a Large Language Model (LLM) predicts the next token based on the previous tokens in a sequence. This is called autoregression. A token can for example be a word or a sentence. LLMs generally consist of large and deep neural networks implementing some kind of transformer architecture.

While LLMs in themselves are impressive, they are limited by the fact that they can only process one type of data: text. A Vision-Language Model (VLM) is a type of foundation model capable of processing both images, videos, and text. These models are usually trained on images with associated text and are thus able to associate keywords in the text with the image, and further process this information in the vein of LLMs. The multimodal properties of a VLM make it possible to extend the reasoning and transfer learning capabilities of foundation models into the field of computer vision. VLMs demonstrate broader generalization capabilities compared to traditional computer vision approaches, for example in classification tasks. They can perform zero-shot classification and they exhibit the ability to reason about visual content, drawing inferences from the composition and content of images [15, 16].

Recently, interest for foundation models has also grown in the robotics community [1]. An important characteristic of foundation models in the context of robot manipulation is that foundation models are built on *transfer learning*, which means that the models can transfer knowledge learned from one task to another [13]. Consequently, a robot controlled by a foundation model would, in theory, not require task-specific training, provided the model has been trained on a sufficiently large and diverse dataset. This presents a significant advantage, as the necessity to collect data for each individual task poses a major limitation to the scalability and practicality of autonomous robot control with AI.

In reality, however, some fine-tuning, or post-training, for downstream tasks is generally needed for foundation models to be effective. Foundation models are typically trained through a multi-stage process. In the initial phase, called pre-training, the model is trained on a large and diverse dataset, and the entire set of model parameters is optimized to minimize a loss function. This stage is critical for establishing the model’s broad generalization and reasoning capabilities and is also the most computationally intensive part of the training pipeline. However, the resulting base model, while capable of handling a wide range of data, is generally not well-suited for specific downstream tasks without further adaptation. This adaptation is achieved through fine-tuning, where the model is further trained on a smaller, task-specific dataset to specialize its behavior. For instance, in the case of a chat-bot, the pre-training dataset might encompass extensive web data from various domains, whereas the fine-tuning dataset would consist of curated prompt-response pairs designed to guide the model toward generating appropriate and contextually relevant replies, instead of, for instance, just continuing the sentence in a grammatically correct way [17]. Fine-tuning can be done on all or a subset of the pre-trained model parameters.

In the case of foundation models for robotics control, pre-training usually involves training the model on huge sets of data for solving diverse embodied tasks with diverse robot models. It is worth noting that in the case of a VLA model, which is a type of model built on a pretrained VLM backbone (see section 2.3), this phase is in reality also a kind of fine-tuning, with the VLM as base model. What we will refer to when referencing fine-tuning a VLA model in this thesis paper, however, is the fine-tuning for downstream tasks that comes after the base VLA model has been trained.

The fine-tuning of a foundation model for robotics control is generally done with a more specific set of data than the pre-training. This dataset usually contains the specific robot which the model is to control, and the specific tasks it will carry out [5, 7, 18]. Since fine-tuning typically requires substantially less data than pre-training, adapting a model that is already capable of generating robot actions and reasoning for downstream tasks, rather than training it from scratch on task-specific data, also helps mitigate the data scarcity challenge.

Earlier works integrating foundation models with robotics include SayCan, Inner Monologue and Text2Motion [19–21]. These models rely on an LLM to propose suitable robot actions from an action library to solve a task given via a

language prompt. While showing promise for robotics control and task planning with foundation models, they are very limited by the model having a finite set of possible actions and that they are unable to process visual inputs, which makes them unable to react to visual cues and to operate in previously unseen environments.

2.2 PaLM-E

For an embodied AI to be able to carry out complex tasks in a physical space, it would need to be able to process impressions from the physical space it operates in, and generate suitable actions in response to these. One embodied foundation model capable of this is the VLM PaLM-E [22]. The E in PaLM-E stands for Embodied, and it is a multimodal model adapted from the LLM PaLM [23] to handle embodied tasks. These tasks are given in the form of language and image inputs, and the outputs can come in the form of, among other things, suitable robot actions. PaLM-E can not directly output low-level robot actions that the embodied agent can execute, but instead outputs the actions in the form of natural language instructions. These instructions must then be translated into low-level actions by another model like LangLp [24] or RT-1 [25] for the robot to be able to carry them out.

2.3 RT-2

A type of foundation model capable of handling multimodal data and which has garnered interest in the robot community is the *Vision-Language-Action* model (VLA). This is an extension of the VLM, but instead of generating text, VLA models generate low-level actions in response to textual prompts and images. A VLA model is, in essence, a pre-trained VLM which is fine-tuned with robot trajectory data to output robot actions, resulting in a model capable of end-to-end planning and execution of robot actions. It is worth noting that the term sometimes encompasses any model that can process multimodal inputs from vision and language and produce low-level robot actions to accomplish an embodied task [3], but we will stick to the original, more narrow definition in this thesis paper. The term was coined by researchers at Google DeepMind in [26]. In their paper presenting this new type of model, the already existing VLMs PaLI-X and PaLM-E [15, 22] were adapted into VLA models, named RT-2-PaLI-X and RT-2-PaLM-E, or just RT-2 as a more general name. Both of these preexisting models generate a sequence of tokens, which are originally translated into natural language. To adapt these models into VLAs, they were co-fine-tuned with both web data and robot trajectory data, to instead output tokens representing low-level actions in the form of end-effector poses in cartesian coordinates. The result was two models that take inputs in the form of an image of the scene, combined with instructions in the form of a natural language prompt, and are able to carry out said task with a manipulator. These models show promising generalization capabilities across tasks, allowing them to perform previously unseen tasks, and are able to reason about how

to solve a task with the help of natural language. In previously seen tasks, RT-2 performs comparably to the older imitation-learning model RT-1, but it significantly outperforms this model in tasks with previously unseen elements. Since the launch of RT-2, the same model architecture has also been trained on the larger X-Embodiment Dataset, creating RT-2-X [27]. This increased the models performance in tasks including spacial awareness, showing that training embodied models on more diverse data may improve task-solving skills.

2.4 Octo

Octo [5] is an open-source generalist policy for robotics control trained on data from the Open-X Embodiment data set [27]. The policy is not built on a pre-trained VLM, but instead uses its own transformer backbone, and as such is not technically a VLA model even though it has many similarities to one. Like a VLA, it is a multimodal model capable of processing inputs in the form of language instructions and visual information, and outputting low-level robot actions. Since it is not trained on any web data, however, it might actually be more comparable to the imitation-learning-based model RT-1 [25] than its successor RT-2. Octo’s action outputs come in the form of *action chunks*. Action chunking is a method proposed by Zhao et al. [28], where actions over a time horizon H are inferred at every timestep, and this is supposed to reduce the effective horizon of the task and reduce compounding errors. It also reduces the number of times new actions have to be generated, when used open-loop like in Octo.

Octo is able to control several robot models out of the box by outputting desired end-effector poses, in several different environments. These capabilities are, however, limited to situations present in the pre-training data. The main contribution of the article is that it shows that pre-training a model on large amounts of diverse data makes it possible to adapt the model to scenarios not seen in pre-training with only a small amount of fine-tuning. This combats the data-scarcity challenge present in the area of autonomous robotics control. The article shows that Octo can be fine-tuned to new embodiments and new action- and observation spaces with only a few hours of additional training.

2.5 OpenVLA

OpenVLA [18] is an open-source VLA model which was released soon after Octo. Like RT-2, OpenVLA is in essence a pre-existing VLM model fine-tuned on robot trajectory data, in this case from the Open-X Embodiment data set [27]. OpenVLA uses Prismatic [29] as its VLM backbone, with weights from Llama 2 7B [30] dominating. Like RT-2 [26], OpenVLA directly translates tokens from the VLM to robot actions, in the form of desired end-effector poses. Following the recipe of RT-2-PaLM-E, this is done by discretizing the possible poses into the least-used token bins from the VLM backbone. While an easy and effective solution, this makes the model unable to output desirable poses in between these discrete action tokens. OpenVLA is an autoregressive policy

that does not use action-chunking. The main contributions the team cites with their work beyond the fact that the model seems to perform well compared to other then-existing models with similar objectives according to their testing, is that the model is the first open-source of its kind. It does however, still typically perform with less than 90 percent accuracy on testing.

Recently, OpenVLA was used with a new fine-tuning recipe, creating OpenVLA-OFT [31]. This new fine-tuning recipe utilizes parallel decoding to simultaneously produce several actions that together forms an action chunk. It also allows for inferring actions from a continuous action space, leading to higher precision in the action predictions. Altogether, this fine-tuning recipe results in a model capable of solving tasks with an average 97.1 percent success rate on the benchmarks it was tested on, while also showing faster inference speed.

2.6 CoT-VLA

CoT-VLA, or Chain-of-Thought-VLA [32], is a model that aims to improve on previous VLA models’ task planning capabilities by incorporating an additional intermediate reasoning phase prior to action generation. Specifically, CoT-VLA first generates an image representing a visual sub-goal, which serves as an intermediate planning step. This image is then used, together with the original visual input and the language prompt, to generate an action chunk in the form of sequential end-effector poses. This extra reasoning step in the inference does seem to improve the models reasoning and task-following capabilities, showing higher or comparable performance to both Octo and OpenVLA on the benchmarks evaluated in the paper. Notably, the policy exhibits higher language-prompt following capabilities than the compared models. The need to generate an image at every inference cycle does, however, slow down the inference speed somewhat.

2.7 π_0

π_0 [7] is a more recent generalist VLA model by the team at Physical Intelligence, which consists partly of the same minds as those behind OpenVLA. π_0 , in contrast to OpenVLA and RT-2, does not translate tokens from the VLM backbone directly into discrete robot actions, but instead utilizes flow matching [33, 34] to generate continuous robot actions instead of discrete ones, like the earlier models mentioned. This enables it to better perform dexterous and high-frequency tasks. It also makes use of a so called action expert to handle the robot-specific tokens, instead of feeding them into the VLM backbone together with the language and image tokens. The action expert consists of a separate set of weights specifically designed for robot actions and it handles the inputs describing the current state, and outputs robot actions in joint-space. In pre-training, the action expert is trained on data from several different robots with different degrees of freedom, making the model capable of controlling several different embodiments out of the box. In addition, π_0 also

utilizes action-chunking. In π_0 , like in Octo, these chunks are executed open-loop without any feedback between the inference steps. For most of the tasks showcased in the article, the control frequency was 50 Hz and the inference frequency was every 25 timesteps. Consequently, the model carried out action chunks spanning 0.5 seconds at a time. According to the paper, π_0 can perform more challenging tasks with much higher success rate than previous comparable models, like OpenVLA and Octo. These tasks include tasks requiring some planning and high level reasoning, like bussing tables. The more complicated tasks do require fine-tuning, however. Since the original publishing of the article, π_0 have been open-sourced, with the code and the weights available at Physical Intelligence’s repository at GitHub.

2.8 $\pi_{0.5}$

Recently, a model called $\pi_{0.5}$ [8] which is built on π_0 , was presented by the same team. This model utilizes the reasoning capabilities of the VLM backbone to first generate high-level actions in natural language, which acts as sub-goals of the language input. These new sub-goals are then processed in the same way that π_0 processes original prompts, making $\pi_{0.5}$ quite similar to CoT-VLA. Like π_0 , $\pi_{0.5}$ utilizes a flow-matching action expert to generate low-level robot actions from a continuous space. In training, $\pi_{0.5}$ uses the FAST action tokenizer [35] to represent actions as discrete tokens. This method is also the product of Physical Intelligence, and they claim that it can speed up the training time five times. This representation, however, is not used at inference time since it increases latency. The main contribution of the $\pi_{0.5}$ paper is that it showcases the generalization and high-level reasoning capabilities of a VLA model, when trained on diverse datasets. The team uses a co-training recipe where the VLA model is trained on data from a range of robot models, tasks, and environments, as well as high-level task planning and multimodal web data in pre-training. During fine-tuning, $\pi_{0.5}$ is trained on more task-specific data, which in this case is cleaning homes. The model shows capabilities of performing tasks in unseen environments at the same level of success as models specifically trained for these tasks in the same environments.

2.9 Safe Control

Ensuring safety when controlling robots with foundation models is so far a relatively unexplored area, but for more traditional approaches to robotics control such as Model Predictive Control (MPC) and Reinforcement Learning (RL), there have been extensive research [36]. Two existing approaches involves creating safety constraints which can guarantee that the robot operates within a safe domain, and letting the policy learn safe behavior with cost functions.

Some recent work have attempted to utilize the reasoning capabilities of an LLM to create these aforementioned safety functions. Brunke et al. [37] let an LLM semantically decide if performing an action was safe or not depending on the objects involved and their relative positions. For example, an unsafe

action was to hold a candle beneath a balloon. The results were used to create a semantic safety filter, through which proposed actions were passed. In a similar vein, Ling et al. [38] let a VLM design costs associated to a robot making contact with different objects. This enabled the embodied agent to solve tasks in cluttered environments without causing *dangerous* collisions since the robot only avoided contact with objects with high costs, like glass, while allowing for contacts with objects made of, for example, rubber.

These two works, however, aimed at utilizing the reasoning capabilities of foundation models to increase the safety of more traditional control schemes, as opposed to increasing the safety of foundation models for robotics control.

Zhang et al. recently released SafeVLA [39], a reinforcement learning approach to solve the safety problem of letting a foundation model control a robot. They fine-tuned the underlying robotics control policy with their by-hand designed objective function, which rewards things like collision avoidance, and show results that indicate that their method both improves the safety and the success rate of the policy. Their approach does, however, not guarantee that the robot’s actions are safe, only increases the likeliness of it. In the same article, the authors also introduce a new simulation benchmark for safety evaluations of models for robotics control, called Safety-CHORES.

Another recent work by Tölle et al. [40] proposed a method for ensuring safe robot actions by restricting the action space of the foundation model with a “safety layer”. This layer is an additional layer through which the output action of the foundation model is passed, and which constrains the action to an action space that is deemed safe. This action space can be constructed so that things like joint limits and collision avoidance are guaranteed to be obeyed, thus only containing “safe” actions. The article shows that constraining the action space in such a way does not significantly negatively impact the policy’s success rate, which shows promise that these kind of restrictions can be a viable path forward. Their method relies on knowing the analytical constraint functions, the system’s state and a control affine system, and that the safety functions are known, twice differentiable, functions.

2.10 Discussing the Future of Embodied AI

Many foundation models for embodied AI show impressive capabilities, and recent models such as $\pi_{0.5}$ exhibit substantial performance improvements over models developed just a year earlier, like Octo and OpenVLA. This rapid progress suggests a strong trajectory toward increased reliability and generalization in the near future. Looking at the success rates of state-of-the-art models like π_0 and $\pi_{0.5}$, however, reveals that there is still some way to go before the models can perform more difficult tasks with near 100% accuracy, something that would be necessary for most real-life use cases.

In addition, one limitation of these models that remains is that while many can perform simple previously unseen tasks in the environment that they are trained for, deployment in new environments and for more complicated tasks requires significant fine-tuning. This means that there is still some work to

be done until VLAs and other foundation models for robotic control can be widely deployed to consumers in the areas where their generality might be most useful. These areas include operating mobile machinery, such as excavators or forest machinery, since they need to be able to perform a wide range of tasks in varying environments, and performing extensive fine-tuning on all of these might not be feasible. It is worth noting that $\pi_{0.5}$ is able to carry out tasks in previously unseen kitchens. This ability to generalize to new environments that are similar to ones previously seen might in the future be carried over to operating autonomous machines for other use cases than cleaning.

When fine-tuning these models, being selective of how much of and which subset of the parameters to tune can have a significant impact on the GPU training time. The team behind OpenVLA, for example, alleges that their model can be fine-tuned to perform a specific downstream task in five to fifteen hours using eight A100 GPUs when updating all parameters during finetuning, but that the same performance can be achieved in the same time using only single A100 GPU if instead parameter efficient fine-tuning with LoRA [41] is performed.

In the future, fine-tuning processes may become sufficiently optimized and automated as to enable large-scale deployment, particularly in industrial applications where substantial resources are available. The development of methods such as the FAST tokenizer [35] also suggest that the training and fine-tuning of foundation models for robotic control could become increasingly efficient in the near future.

One problem with the FAST tokenizer in particular, however, is that it can lead to increased latency between inference cycles. In the original paper, an increased inference time from 100 ms to 730 ms for actions spanning one second when using π_0 -FAST instead of the π_0 base model was reported. This kind of inference time can lead to stuttered and jerky movements, which are generally undesirable since they can be dangerous to both the robot and the environment it operates in. The fine-tuning recipe used to create OpenVLA-OFT, on the other hand, actually decreased the inference time compared to the base OpenVLA model, from a 239.6 ms latency to 72.9 ms [31]. This indicates that it is possible to speed up the inference time of these models, which in the future might lead to models capable of real-time inference at high frequencies. This development might be necessary before these models can be reliably employed in areas where high precision is necessary, like in industrial contexts.

Altogether, the future for foundation models for robotics control shows promise with their generalization capabilities and their fast advancement, but some further development is still necessary for real-life deployment. The question of how to guarantee safety when using these models also remains unanswered, but this is something that we hope to contribute towards solving in this thesis.

3 Sparse Evaluation

In this section, we introduce a proof-of-concept for a new method called *Sparse Evaluation*. We present some background of our chosen physics simulator, AGX Dynamics, as well as the model on which we have chosen to implement our proof-of-concept: ACT [28].

3.1 Method Description

Building toward the same goal as those discussed in section 2.9, we propose another approach towards ensuring the safe operation of robots using foundation models. With the Sparse Evaluation method, we propose to integrate foundation models for robotics control with physics solvers at inference time, to predict if a proposed action is safe or not. The rationale behind our method is that because it can be hard to accurately predict everything in the world from only visual data, like most of these foundation models do, it is critical for the safe operation of autonomous robots that these factors are accounted for before an action is executed. Examples of these uncertain factors could be the exact size of objects, their mass, their friction, and whether they have freedom or movement or are stuck in place. In addition to these kind of uncertainties, analyzing actions with a physics solver before rolling them out can also enable things like detecting collisions and movements too close to the joint limits. This approach also facilitates the identification of unforeseen consequences resulting from certain actions, such as the whipping effects that may arise when interacting with interconnected elements, or the destabilization of a stack caused by manipulating a single component within it.

In the Sparse Evaluation method, these uncertainties are caught by simulating the proposed trajectory in a digital twin of the real world with the help of a physics solver, and then evaluating the consequences of objects having varying properties within an uncertainty span during some, sparse, critical moments. Only looking at these critical moments allows one to evaluate a large number of unknowns almost instantly if executed in parallel. The results of these evaluation can then be used as basis to decide if a trajectory proposed by an AI is safe, if further examination of the environment is needed to ensure safety, or if another approach should be taken instead. In Figure 1, we provide a flow chart over how we imagine our proposed method could be used. The foundation model (VLA here for brevity), first infers the whole trajectory in simulation (Frequent Evaluation), and Sparse Evaluation is then carried out in some critical moments. If the trajectory is deemed safe, one can let the model act it out with the robot in real life, otherwise a new prompt can be sent to the model. A more thorough description of the scheme is provided in section 3.5.

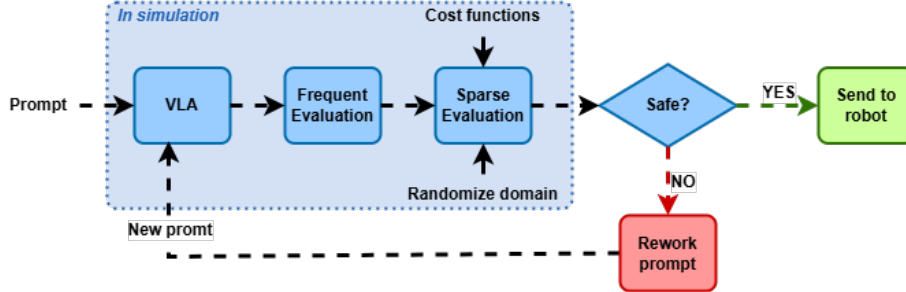


Figure 1: A flow chart over how we imagine the Sparse Evaluation method can be used together with a foundation model for robot control (VLA here for brevity).

In this thesis paper, we implement this method on a limited example, as a proof-of-concept. We consider this a first step toward integrating foundation models for robot control and physics solvers at inference time, to facilitate the safe operation of autonomous machines with these models.

3.2 AGX Dynamics

AGX Dynamics is a physics engine that enables stable high-fidelity multibody simulations, provided by the company Algoryx. It can be used to model non-smooth mechanics like impacts, contacts, and dry friction with high accuracy, which is critical when one wants to use a simulation to predict what would happen in the same scenario in real life. We therefore have chosen AGX Dynamics as the physics engine with which we test our method. In this section, we provide a brief description of some of the mathematics at work in the physics engine.

3.2.1 Multi-body Dynamics

In AGX Dynamics, the links of a robot are modeled as rigid bodies, whose movements are constrained relative to each other with joints. In total, the system consists of N_b rigid bodies, N_j constraints and actuators, and N_c contacts. AGX Dynamics solves multibody dynamics on descriptor form, and the dynamics of the system can be expressed with the following set Differential-algebraic system of equations (DAE):

$$\begin{cases} \mathbf{M}\dot{\mathbf{v}} - \mathbf{G}(\mathbf{x})^T \boldsymbol{\lambda} - \bar{\mathbf{G}}(\mathbf{x})^T \bar{\boldsymbol{\lambda}} = \mathbf{f}(\mathbf{x}, \mathbf{v}, t) & (1) \\ \varepsilon \boldsymbol{\lambda} + \mathbf{g}(\mathbf{x}) = \mathbf{0} & (2) \\ \gamma \bar{\boldsymbol{\lambda}} + \bar{\mathbf{G}}(\mathbf{x}) \mathbf{v} = \mathbf{u}(t), & (3) \end{cases}$$

where Equation 1 models the dynamics derived from classical mechanics, and Equations 2 and 3 enforces the holonomic and non-holonomic constraints, respectively. The term $\mathbf{f}(\mathbf{x}, \mathbf{v}, t)$ in Equation 1 describes the external forces acting

on the bodies, where \mathbf{x} contains the positional and rotational information of the bodies, $\mathbf{v} = \dot{\mathbf{x}}$ contains their velocities, and t is the time. \mathbf{M} is the mass matrix made up of N_b sub matrices \mathbf{M}_i , each containing the mass m_i and inertia tensor \mathbf{I}_i of the corresponding body, such that

$$\mathbf{M}_i = \begin{bmatrix} m_i I_{3 \times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_i \end{bmatrix}. \quad (4)$$

The terms $\mathbf{G}(\mathbf{x})^T \boldsymbol{\lambda}$ and $\bar{\mathbf{G}}(\mathbf{x})^T \bar{\boldsymbol{\lambda}}$ each consist of a lagrangian multiplier and a Jacobian matrix derived from the constraint functions, and are the forces enforcing the holonomic and non-holonomic constraints, respectively. An ideal holonomic constraint $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ enforces positions to a surface of the full domain, and examples of these constraints are rotational hinges and prismatic joints, which only allows for movement in one dimension. In Equation 2, however, the added term $\varepsilon \boldsymbol{\lambda}$ allows for some deviation from this surface. The parameter ε decides the constraint compliance, which can be interpreted as an inverse spring. It is zero in an ideal holonomic constraint, but can be tuned to either optimize constraint satisfaction, improve conditioning, or to describe a real physical compliance in the constraint. The Jacobian corresponding to the holonomic constraint is defined by $\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{G}(\mathbf{x})$. The non-holonomic constraint is time- and velocity dependent, $\bar{\mathbf{g}}(\mathbf{v}, t) = 0$, and can be used to model frictional contacts and actuators in the joints. Equation 3 thus enables one to set a target speed $\mathbf{u}(t)$ in the joints, and the term $\gamma \bar{\boldsymbol{\lambda}}$ models damping. Similarly to the compliance term in Equation 2, the damping γ can be set to describe a physical property in the constraint or tuned for numerical stability.

3.2.2 Time integration

AGX Dynamics uses an implicit time-integration scheme called the SPOOK stepper [42], which is stable even for larger time steps. With this scheme the DAE in Equations 1 - 3 are adapted to the matrix equation

$$\begin{bmatrix} \mathbf{M} & -\mathbf{G}_n & -\bar{\mathbf{G}}_n \\ \mathbf{G}_n & \boldsymbol{\Sigma} & \mathbf{0} \\ \bar{\mathbf{G}}_n & \mathbf{0} & \bar{\boldsymbol{\Sigma}} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{n+1} \\ \boldsymbol{\lambda}_n \\ \bar{\boldsymbol{\lambda}}_n \end{bmatrix} = \begin{bmatrix} \mathbf{p}_n \\ \mathbf{q}_n \\ \mathbf{u}(t) \end{bmatrix}, \quad (5)$$

which is solved at every timestep n . The positions are updated with $\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_{n+1} \Delta t$ after solving the system. \mathbf{p}_n and \mathbf{q}_n are defined as

$$\mathbf{p}_n = \mathbf{M} \mathbf{v}_n + \Delta t \mathbf{M}^{-1} \mathbf{f}_n \quad (6)$$

and

$$\mathbf{q}_n = -\frac{4}{\Delta t} \Upsilon \mathbf{g} + \Upsilon \mathbf{G} \mathbf{v}_n. \quad (7)$$

$\boldsymbol{\Sigma}$, and $\bar{\boldsymbol{\Sigma}}$ and Υ are for regularization and stabilization, and can be mapped to compliance and damping, where τ is the relaxation time for the spring-like

behavior in the compliance. They are defined as:

$$\boldsymbol{\Sigma} = \frac{4}{\Delta t^2} \text{diag} \left(\frac{\varepsilon_i}{1 + 4\tau_i/\Delta t} \right), \quad (8)$$

$$\bar{\boldsymbol{\Sigma}} = \frac{1}{\Delta t} \text{diag}(\gamma_i), \quad (9)$$

and

$$\boldsymbol{\Upsilon} = \frac{1}{\Delta t} \text{diag} \left(\frac{1}{4\tau_i/\Delta t} \right), \quad (10)$$

respectively. For further explanation on the SPOOK stepper, please refer to the original publication [42].

3.2.3 MCP

Since AGX Dynamics allows for non-smooth dynamics, like dry friction and impacts, some complementary conditions governing these laws also have to be introduced. The final Mixed Complementary Problem (MCP) that is thus solved in every step of the simulation can be formulated as

$$\begin{aligned} \mathbf{H}\mathbf{z} - \mathbf{r} &= \mathbf{w}_l - \mathbf{w}_u \\ 0 &\leq \mathbf{z} - \mathbf{l} \perp \mathbf{w}_l \geq 0, \\ 0 &\leq \mathbf{u} - \mathbf{z} \perp \mathbf{w}_u \geq 0 \end{aligned} \quad (11)$$

where \mathbf{H} , \mathbf{z} and \mathbf{r} are the matrices in Equation 5, while \mathbf{w}_l and \mathbf{w}_u are slack variables, and \mathbf{u} , \mathbf{z} are the conditions for the non-smooth dynamics.

3.3 ACT

Action Chunking with Transformers (ACT), [28] is chosen as the ML model on which we implement our Sparse Evaluation algorithm. While ACT is not a foundation model, it is chosen as the base for this proof-of-concept because it is open-source and is easy to implement, without the need of fine-tuning, since it comes pre-packaged with a simulation environment in MuJoCo, which is an open-source physics engine run by Google DeepMind. ACT is an imitation-learning algorithm based on transformer architecture, and is built to handle fine-manipulation robot tasks. It uses action-chunking to reduce the effective horizon of the task, which the authors claim reduces compounding errors. To improve the smoothness of the policy, ACT also implements *temporal ensembling*. This means that a new action chunk is inferred at every timestep, and the overlapping action chunks are averaged to produce a smoother path. The output of ACT are low-level actions in joint space, which can directly be used for robot control.

In the ACT article, eight tasks are carried out using two robotic arms of the model ViperX 300s, which has six degrees of freedom each. While most of the tasks are carried out using real robots, two of the tasks are carried out in

simulation using MuJoCo. In one of these simulated tasks, the two robots are situated across from each other on a tabletop, with a box placed between them. The goal of this task is for the left-hand robot to pick up the box and hand it over to the other robot. This is the task we have chosen to test our proof-of-concept on. The ACT repository provides a simulation environment for the task in MuJoCo, as well as scripted episodes on which we trained ACT.

3.4 Modeling

As previously stated, we used AGX Dynamics as the physics engine with which the multi-body-system was simulated, and with which our analysis was carried out. To be able to use a path generated by ACT, we thus built a simulation environment identical to the one in MuJoCo on which ACT was trained, but using AGX Dynamics instead. Figure 2 shows the initial state of the simulation, as modeled in AGX Dynamics. Note that some differences are naturally present between the original setup in MuJoCo and ours, because of the different physics engines. An active choice that was made that differentiates our setups is that for the force limits, we used the limits from the urdf-file [43] that was used to initialize the robot, instead of the very large limits used by the team behind ACT. This was done to better model realistic robot behavior.

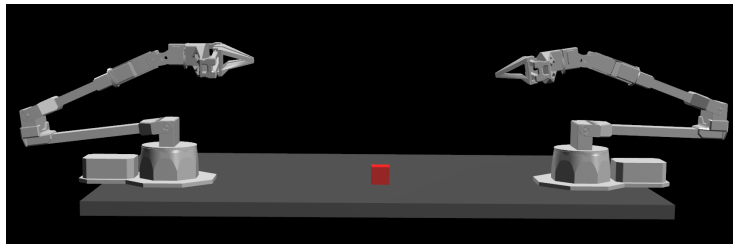


Figure 2: Setup for the experiments, as simulated in AGX Dynamics. Two ViperX 300s are situated across from each other on a tabletop, with a red cube placed between them.

In the simulation, the robots are represented by a collection of rigid bodies and each body has assigned physical qualities like mass and directional inertia. The bodies are connected in a chain by holonomic constraints, which limits their movement relative to each other to only one direction. The constraints form the joints of the arm, and in the first six joints, they are rotational. The fingers are constrained with linear, or prismatic, constraints. The robots has six degrees of freedom each, which allows them full maneuverability in the 3D space without any redundancy. The links in the robotic arms are moved by electric motors in each joint, which are modeled with non-holonomic constraints. Since ACT outputs actions in the form of desired joint positions for the next timestep, it is trivial to translate them into robot actions, without the need to calculate inverse kinematics. In MuJoCo, the motors are controlled using a PID-controller native to the physics engine, but attempting to recreate

this in AGX Dynamics resulted in choppy trajectories, and so another control method was chosen. In our simulation, the motors are controlled by setting a target speed for the motors, which is trivially calculated as the difference between the joint’s desired angle according to ACT and its current angle, divided by the timestep. This target speed is used in AGX Dynamic’s dynamics solver, and the applied torque is calculated for every motor. This allows us to model a more sophisticated controller that generates smooth actions while staying within the motor’s force limits.

AGX Dynamics supports both direct, iterative and mixed solvers for the MCP problem (Equation 11), but since our system is relatively well-conditioned, and accuracy is of big importance, we use the direct solver for everything in our simulation. AGX Dynamics also supports several friction models, but for this work we used the projected cone friction model with exact cone projection [44] and a direct solver to model dry friction in the system. In this model, the Coulomb friction law is represented by a friction cone normal to the contact point. The height of the cone is dictated by the normal force \mathbf{F}_n , and its width is dictated by the Coulomb friction law, so that its radius is $r = |\mu\mathbf{F}_n|$, where μ is the friction coefficient for the contact. The projected cone friction model ensures that the contact force always lies inside this cone as long as the contact remains in stick mode. When the contact force falls on the surface of the cone, the contact will switch to sliding mode, and gain a relative velocity in the opposite direction from the friction force [45]. This friction model ensures realistic friction forces and sliding contacts.

AGX Dynamics models contact mechanics with a small compliance built into the contact materials, to prevent numerical instability and degeneracy. Mechanically, this compliance is modeled like a linear-elastic material, with a spring constant depending on the contact’s Young modulus and rest length h . We use the area based approach [46] to determine h , since this is more accurate for objects with a small size and mass.

3.5 Implementation

In this previously described setup, a path with target joint configurations previously generated by ACT in MuJoCo was rolled out from start to finish, and the transformation matrices and velocities of the bodies, as well as some evaluation criteria, were recorded in every timestep. In this initial rollout, the mass of the box was $m = 0.05$ kg, since this is the mass ACT is expecting. This rollout is what we refer to as the *frequent evaluation*, since we let the simulation run from start to finish in a high-frequency simulation. Notably, the path generated by ACT was interpolated using cubic splines before this rollout, which resulted in a smoother robot path than what we got when using ACT’s original control frequency at 50 Hz. We could therefore instead use 500 Hz as both the control and simulation frequency. This simulation frequency allowed for modeling the system dynamics and contacts with high accuracy.

As evaluation criteria in the frequent evaluation, the box’s acceleration and the torques applied by the electric motors were recorded. The latter were nor-

malized with the maximum torque the respective motors could apply. Additionally, the weighted contact Factor of Safety ($FOS_{contact}$) between the box and the grippers was recorded. This measurement describes whether a contact is in slip- or stick-mode. When the contact FOS approaches one, the contact enters slip mode, and conversely, a contact FOS far greater than one indicates stick mode. The contact FOS is computed by evaluating the slip or stick factor at each of the N contact points and applying a weighting based on the sum of the normal force magnitudes at these points. This weighting scheme ensures that contact points with weak normal forces do not exert an undue influence on the overall measurement. The contact FOS weighted by the normal forces is defined in [47] as

$$FOS_{contact} = \left[\frac{1}{\sum_j^N \lambda_n^j} \sum_i^N \lambda_n^i \left(\frac{c_A^i + \mu^i \lambda_n^i}{\lambda_t^i} \right)^{-1} \right]^{-1}, \quad (12)$$

where N is the number of contact points, λ_n and λ_t are the magnitudes of the normal and tangential forces, respectively, μ is the friction coefficient, and c_A is the cohesion. Since the cohesion is zero in our case, this expression is simplified. We also introduce the measurement Motor Factor of Safety, which we define as

$$FOS_{motor} = \min\{FOS_{motor}^1, \dots, FOS_{motor}^{N_j}\}, \quad (13)$$

where

$$FOS_{motor}^i = \frac{\tau_i^{max}}{|\tau_i|}, \quad (14)$$

and $i \in [1, N_j]$ is the index of all the actuators in the system, τ_i is the torque currently applied, and τ_i^{max} is its upper torque limit. Similarly to the contact FOS, a motor FOS far greater than one indicates that the motors operate far from their limits, and a motor FOS close to one indicates that they are operating close to their limit, which might have undesirable consequences.

In our limited proof-of-concept, only the consequences of the box having another mass than expected was investigated. Two moments were chosen as critical, namely, a moment just after the box has been lifted and lost contact with the table, and a moment just after the box has been handed over to the left-hand robot. These were chosen because they show two different potential points of failure: that the robot may be unable to lift the box from the ground because of the gripper's limited strength, and that the robot might be too weak to reliably carry the box without overexerting the motors. Figure 3 shows snapshots of these exact moments in simulation.

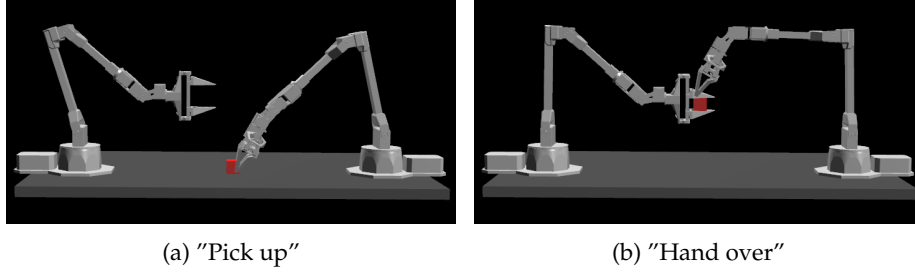


Figure 3: Snapshots showing the configuration of the system at which the sparse evaluation is done. Figure 3a shows the configuration just after the box has been lifted from the table, and Figure 3b shows a moment just after the box has been handed over to the robot on the left-hand side.

When evaluating these points, the system was initiated in exactly the same state as it was in that moment in the frequent evaluation, but with a different mass on the box. This initialization was done with the help of the transform matrices and velocities of the rigid bodies previously recorded in the frequent evaluation. The simulation was then stepped forward one timestep with the same joint target positions as before, and the results were recorded. This was done in parallel with a number of different masses. With the results from the sparse evaluation, a cost is calculated corresponding to each mass. We chose to base this cost function on the contact- and motor FOS corresponding to each mass, according to

$$C_m = Ae^{-\alpha(\text{FOS}_{\text{contact}}^m - 1)} + Be^{-\beta(\text{FOS}_{\text{motor}}^m - 1)}, \quad (15)$$

where m is the index of the corresponding mass, and $A = 0.8$, $B = 1$, $\alpha = 0.7$ and $\beta = 2$ are tunable parameters. This cost function is only an example, and can be adapted as required to different contexts. Algorithm 1 provides a description for the whole Sparse Evaluation procedure.

Algorithm 1 Sparse Evaluation

```
actions = trajectory of consecutive actions
parameters = list with parameters which are to be varied ▷ In our case,  $m$ .

N ← length of actions
states ← []
measurements ← []
init_simulation()
for i = 1 to N do ▷ Frequent evaluation
  z ← solve_simulation(actions[i]) ▷ Solve system's dynamics
  state ← update_simulation(x) ▷ Update simulation's positions, etc.
  measurements.append(z)
  states.append(state)
end for
T ← find_critical_timesteps(measurements) ▷ List with critical moments
for t in T do ▷ Sparse Evaluation in parallel over t and p
  for p in parameters do
    set_simulation_state(states[t], parameters[p])
    z ← solve_simulation(actions[t])
    C ← calculate_cost(z)
  end for
end for
```

4 Results

In this section, the results from the Sparse Evaluation in the two heuristically chosen critical moments are presented. A plot showing the measured parameters over time in the frequent evaluation can be found in Appendix A.

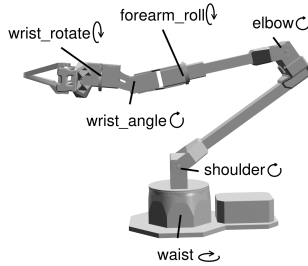
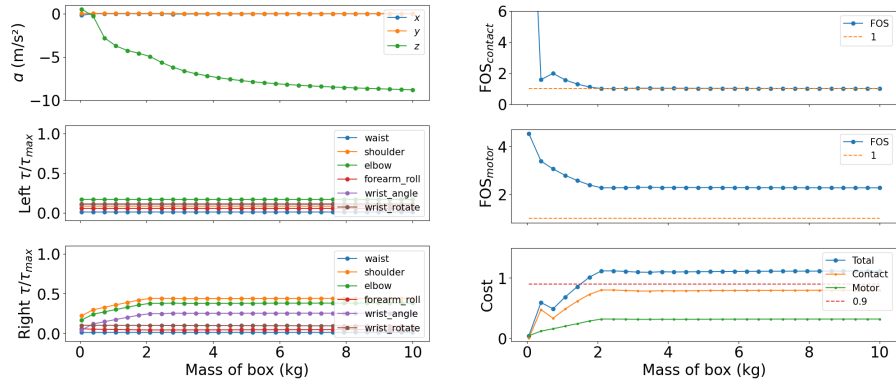


Figure 4: The ViperX 300s with labeled joints. The arrows indicate their respective axis of rotation.

The results corresponding to the “Pick up” and “Hand over” critical moments are presented in Figure 5 and Figure 6, respectively. The results are presented over varying masses of the box. This variation could be interpreted as an uncertainty span over the box’s actual mass. In the sub figures showcasing the calculated cost for each mass, a line at $C_m = 0.9$ has been drawn to symbolize a maximum acceptable cost. This is only for illustrating how such a cost function could be used, and has no practical meaning in the scope of this work. Note also that only the torques applied in the rotational joints are being

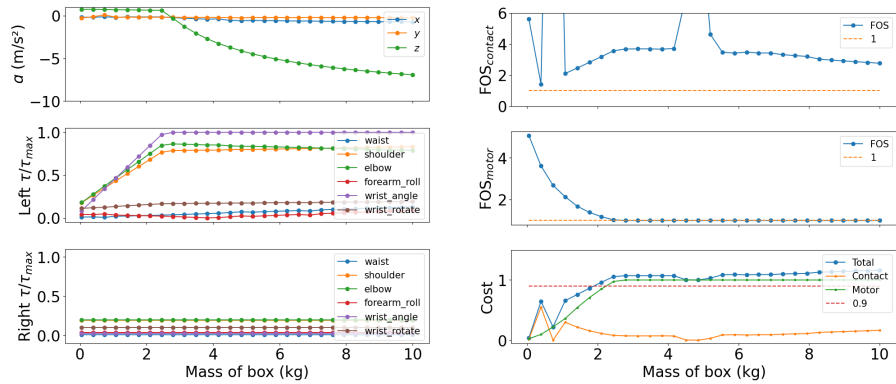
taken into account in the presented results. This is because the actuators in the prismatic joints in the gripper always reaches their force limits when grabbing the box, no matter its mass, and this measurement is therefore not very informative. Figure 4 shows the labels corresponding to each joint.



(a) Acceleration of the box & torques

(b) FOS & cost

Figure 5: Results from varying the mass of the box in the "Pick up" critical moment. Sub figure 5a contains the directional accelerations of the box, a , where z is the direction normal to the table, as well as the normalized torques in the joints of the two robots. Sub figure 5b contains the FOS values corresponding to each mass, as well as the associated cost and its two contributions.



(a) Acceleration of the box & torques

(b) FOS & cost

Figure 6: Results from varying the mass of the box in the "Hand over" critical moment. The subfigures contain the same information as in Figure 5, but for this timestep.

In the "Pick up" moment (Figure 5), one can see that the cost exceeds the limit at around $m = 1.5$ kg, with the largest contribution coming from the term containing the contact FOS. This corresponds to a contact FOS close to one, as well as a downwards acceleration at around -5 m/s². Increasing the mass further leads to the acceleration approaching the gravitational acceleration, indicating free fall for higher masses. One can also observe that the applied torques in three of joints of the robot attempting the lift increases when the mass increases, up to $m = 2$ kg. The figure also shows that the robot not in contact with the box at this moment is unaffected by a change of its mass.

Figure 6, showing results from the "Hand over" critical moment, shows that the cost limit is exceeded at around $m = 1.9$ kg, with the significant contribution coming from the motor FOS. This corresponds to the actuator controlling the wrist angle of the robot applying around 80 percent of its maximum torque. One can also observe that when the mass of the box exceeds 2.7 kg, the torque limit is reached, which in turn leads to the box gaining an acceleration downwards for higher masses, since the robot is then unable hold the desired position. Since the robot is grasping the box with the fingers over and underneath the box at this moment, the gravitational pull does not cause the box to slip out of the robots grasp. Instead, the torque limits in the joints' motors are the deciding parameters of whether the action is executed as expected or not. This grasping configuration can also explain why the contact FOS in Figure 6b does not have a clear pattern, since the gravitational pull does not cause it to slip.

5 Discussion

In this section, we discuss the proposed Sparse Evaluation method. First, we interpret and discuss our results from the proof-of-concept implementation. We also discuss some limitations of our current iteration of the method, as well as compare it to earlier works in the area.

5.1 Interpreting the Results

While the Sparse Evaluation method is implemented on a limited example in this thesis paper, the results indicate that it could be a useful method for ensuring safety when using foundation models for robotics control. If, for example, one is certain that the box does not weigh more than 1 kg in our scenario, the results indicate that the proposed trajectory is safe, since the cost does not exceed 0.9 for lower masses. If the possible span of masses includes 1.5 kg, on the other hand, there would be reason for caution, since the results from the "Pick up" critical moment show that the robots might not then be able to carry out this action as expected.

In practice, these results might be used to decide if an action is suitable to perform or not, based on an uncertainty span one has over the unknown parameter (the mass in our case). How one interprets a cost that exceeds the

“allowed” value might differ in different applications. For example, if it is absolutely critical that the proposed action does not fail, one might act on the information conservatively and interpret this as a signal that the action should not be executed. On the other hand, one could also interpret the fraction of parameters corresponding with a too high cost as a probability of success. In our example, if we have the uncertainty span $m \in [0.05, 10]$, this probability would be 15% for the “Pick up” action, and 19% for the “Hand over” action.

By examining the contributions of the terms in the cost function (Equation 15 in our case), one can trace from where the danger arises, which makes it possible to react suitably. If, for example, the results indicate that the robot might be too weak for the specific action, one could either make efforts to reduce the uncertainty of the parameter value, or one could let the foundation model generate another, more suitable, action. One such action if the goal is to move an object, for example, could be to prompt the model to push the object instead of lifting it. The cost function and how one interprets it can, naturally, be adapted to one’s specific use case.

5.2 Limitations and Future Work

In a real-life scenario there are often more uncertainties than only the mass of a specific object which have to be taken into account. These can include, but are not limited to, the friction and size of objects, as well as the consequences that moving them in certain ways might have. Because of this, we believe that an important step towards a future real-life usage of the Sparse Evaluation method is to expand the scope of what it can detect. Some things, like exploring what would happen if contacts have other friction coefficients than expected, would be trivial to implement, while other things require a bit more work. This includes a method for catching unwanted effects in complex systems, like the whip of a chain or the destabilization of a stack, like we discussed earlier. These sort of occurrences could, however, be caught by a suitable cost function, and is therefore far from unattainable. Similarly, contact detection could be achieved by designing a cost function which penalizes unwanted contacts between objects. Something that might pose a bigger problem is if the size of an object is uncertain, since our method relies on initializing a state with previously saved transformation matrices from the Frequent Evaluation, and this would present an issue if, for example, a gripper is grasping the object in question. We do, however, believe that it is not impossible to solve this problem.

In this proof-of-concept implementation, the critical moments were chosen manually, by heuristically identifying moments in which the robot might fail if the mass was different than what was expected by the policy. This would naturally be cumbersome in a real-life scenario. A more systematic possible approach could be to identify instances within the frequent evaluation where specific measured criteria suggest a change in the system’s behavior. For example, this could involve detecting a sudden increase in acceleration or a rapid variation in one of the monitored parameters between consecutive measurements. Such patterns could be detected using automated methods, like with

an integrated optimizer that searches for optimal evaluation moments. An optimizer could also be utilized to find critical parameters in a system. We believe that integrating this kind of optimizer with our method could be a step towards real-life deployment.

Another aspect of the Sparse Evaluation method which requires some work before real-life deployment is that it could be optimized in terms of time consumption. One improvement that would reduce a lot of overhead is that instead of stepping the simulation forward a full step in the sparse evaluation, and updating all positions and preparing for a new time step, it would be enough to only perform one "solve", that is, only solving the multi-body system of equations for the generated velocities and forces, and perform some contact detection if applicable.

In the future, we would also like to implement this method with a foundation model, as opposed to an imitation learning based model as was done in this proof-of-concept. This would facilitate one to provide feedback to the model based on the results from the Sparse Evaluation. We also believe that since foundation models are more prone to unpredictable and dangerous behavior than imitation learning based models, it would produce interesting results.

5.3 Comparing to Earlier Work

In comparing our Sparse Evaluation approach with prior work on ensuring safety in robot control using foundation models, we find the closest resemblance to the method proposed by Tölle et al. [40], which enforces safety by constraining actions to a predefined safe action space. A key distinction of our method is that it does not require explicit system knowledge; instead, it leverages a physics simulator to the model system dynamics. This characteristic enables the application of Sparse Evaluation on complex systems with high-dimensional parameter spaces.

Furthermore, we argue that our approach may offer stronger safety guarantees than existing methods based solely on foundation models [37, 38], since these can be susceptible to hallucination. In contrast to reinforcement learning-based approaches [39], our integration of a foundation model with a physics-based solver allows for forward prediction of system behavior, thereby enhancing the ability to ensure safe operation.

6 Conclusion

Foundation models for robotics control is a rapidly evolving area, where models like Octo, OpenVLA, π_0 and $\pi_{0.5}$ show promising reasoning and generalization capabilities [5, 7, 8, 18]. The question of how to ensure safety when operating robots with these models, however, remains to be answered. We have proposed a novel method, dubbed Sparse Evaluation, which we believe might

contribute toward solving this. This method is aimed at integrating foundation models for robot control with a physics solver, to screen proposed actions before execution and thus guarantee safety. In this thesis paper, we have provided a small proof-of-concept for this method, and we have found that the method shows promise in that it can be used to rapidly examine the consequences of a parameter having many different values, which can help with deciding if an action is safe or not depending on the uncertainty of said value. For real-life deployment, however, the method would have to be considerably expanded upon, both in scope and efficiency. It also remains to be seen how the method can be integrated with a foundation model to provide feedback on proposed actions.

References

- [1] R. Firoozi *et al.*, “Foundation models in robotics: Applications, challenges, and the future,” *The International Journal of Robotics Research*, vol. 0, no. 0, p. 02783649241281508, 0. DOI: 10.1177/02783649241281508. eprint: <https://doi.org/10.1177/02783649241281508>. [Online]. Available: <https://doi.org/10.1177/02783649241281508>.
- [2] D. Eriksson and R. Ghabcheloo, “Comparison of machine learning methods for automatic bucket filling: An imitation learning approach,” *Automation in construction*, vol. 150, pp. 104843–, 2023, ISSN: 0926-5805.
- [3] Y. Ma, Z. Song, Y. Zhuang, J. Hao, and I. King, *A survey on vision-language-action models for embodied ai*, 2025. arXiv: 2405.14093 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2405.14093>.
- [4] O. X.-E. Collaboration *et al.*, *Open X-Embodiment: Robotic learning datasets and RT-X models*, <https://arxiv.org/abs/2310.08864>, 2023.
- [5] O. M. Team *et al.*, *Octo: An open-source generalist robot policy*, 2024. arXiv: 2405.12213 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2405.12213>.
- [6] M. Kim *et al.*, “Openvla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024.
- [7] K. Black *et al.*, π_0 : *A vision-language-action flow model for general robot control*, 2024. arXiv: 2410.24164 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2410.24164>.
- [8] P. Intelligence *et al.*, $\pi_{0.5}$: *A vision-language-action model with open-world generalization*, 2025. arXiv: 2504.16054 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2504.16054>.
- [9] H. Liu *et al.*, “A survey on hallucination in large vision-language models,” *arXiv preprint arXiv:2402.00253*, 2024.
- [10] Z. Bai *et al.*, “Hallucination of multimodal large language models: A survey,” *Preprint*, 2025.
- [11] L. Huang *et al.*, “A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions,” *en, ACM Trans. Inf. Syst.*, Nov. 2024.
- [12] J. Fan, J. Wu, H. Chu, Q. Ge, and B. Gao, *Hallucination elimination and semantic enhancement framework for vision-language models in traffic scenarios*, 2024. arXiv: 2412.07518 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2412.07518>.
- [13] R. Bommasani *et al.*, *On the opportunities and risks of foundation models*, 2022. arXiv: 2108.07258 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2108.07258>.

- [14] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, “Efficient transformers: A survey,” *eng, ACM computing surveys*, vol. 55, no. 6, pp. 1–28, 2023, ISSN: 0360-0300.
- [15] X. Chen *et al.*, *Pali-x: On scaling up a multilingual vision and language model*, 2023. arXiv: 2305.18565 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2305.18565>.
- [16] J.-B. Alayrac *et al.*, *Flamingo: A visual language model for few-shot learning*, 2022. arXiv: 2204.14198 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2204.14198>.
- [17] D. Bergmann. “What is fine-tuning?” (2024), [Online]. Available: <https://www.ibm.com/think/topics/fine-tuning> (visited on 05/20/2025).
- [18] M. J. Kim *et al.*, *Openola: An open-source vision-language-action model*, 2024. arXiv: 2406.09246 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2406.09246>.
- [19] M. Ahn *et al.*, “Do as i can and not as i say: Grounding language in robotic affordances,” in *arXiv preprint arXiv:2204.01691*, 2022.
- [20] W. Huang *et al.*, *Inner monologue: Embodied reasoning through planning with language models*, 2022. arXiv: 2207.05608 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2207.05608>.
- [21] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, “Text2motion: From natural language instructions to feasible plans,” *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, Nov. 2023, ISSN: 1573-7527. DOI: 10.1007/s10514-023-10131-7. [Online]. Available: <http://dx.doi.org/10.1007/s10514-023-10131-7>.
- [22] D. Driess *et al.*, *Palm-e: An embodied multimodal language model*, 2023. arXiv: 2303.03378 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2303.03378>.
- [23] A. Chowdhery *et al.*, *Palm: Scaling language modeling with pathways*, 2022. arXiv: 2204.02311 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2204.02311>.
- [24] C. Lynch and P. Sermanet, *Language conditioned imitation learning over unstructured data*, 2021. arXiv: 2005.07648 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2005.07648>.
- [25] A. Brohan *et al.*, *Rt-1: Robotics transformer for real-world control at scale*, 2023. arXiv: 2212.06817 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2212.06817>.
- [26] A. Brohan *et al.*, *Rt-2: Vision-language-action models transfer web knowledge to robotic control*, 2023. arXiv: 2307.15818 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2307.15818>.
- [27] E. Collaboration *et al.*, *Open x-embodiment: Robotic learning datasets and rt-x models*, 2024. arXiv: 2310.08864 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2310.08864>.

- [28] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, *Learning fine-grained bimanual manipulation with low-cost hardware*, 2023. arXiv: 2304.13705 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2304.13705>.
- [29] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh, "Prismatic vlms: Investigating the design space of visually-conditioned language models," in *International Conference on Machine Learning (ICML)*, 2024.
- [30] H. Touvron *et al.*, *Llama 2: Open foundation and fine-tuned chat models*, 2023. arXiv: 2307.09288 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2307.09288>.
- [31] M. J. Kim, C. Finn, and P. Liang, *Fine-tuning vision-language-action models: Optimizing speed and success*, 2025. arXiv: 2502.19645 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2502.19645>.
- [32] Q. Zhao *et al.*, *Cot-vla: Visual chain-of-thought reasoning for vision-language-action models*, 2025. arXiv: 2503.22020 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2503.22020>.
- [33] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, *Flow matching for generative modeling*, 2023. arXiv: 2210.02747 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2210.02747>.
- [34] Q. Liu, *Rectified flow: A marginal preserving approach to optimal transport*, 2022. arXiv: 2209.14577 [stat.ML]. [Online]. Available: <https://arxiv.org/abs/2209.14577>.
- [35] K. Pertsch *et al.*, *Fast: Efficient action tokenization for vision-language-action models*, 2025. arXiv: 2501.09747 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2501.09747>.
- [36] L. Brunke *et al.*, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Autonomous Robots*, vol. 45, pp. 411–444, May 2022. [Online]. Available: <https://doi.org/10.1146/annurev-control-042920-020211>.
- [37] L. Brunke *et al.*, *Semantically safe robot manipulation: From semantic scene understanding to motion safeguards*, 2025. arXiv: 2410.15185 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2410.15185>.
- [38] Y. Ling, K. Owalekar, O. Adesanya, E. Bıyık, and D. Seita, *Impact: Intelligent motion planning with acceptable contact trajectories via vision-language models*, 2025. arXiv: 2503.10110 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2503.10110>.
- [39] B. Zhang *et al.*, *Safevla: Towards safety alignment of vision-language-action model via safe reinforcement learning*, 2025. arXiv: 2503.03480 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2503.03480>.
- [40] M. Tölle *et al.*, *Towards safe robot foundation models using inductive biases*, 2025. arXiv: 2505.10219 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2505.10219>.

- [41] E. J. Hu *et al.*, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2106.09685>.
- [42] C. Lacoursière, “Ghosts and machines: Regularized variational methods for interactive simulations of multibodies with dry frictional contacts,” *PhD thesis, Umeå University*, 2007.
- [43] S. Wiznitzer, L. Schmitt, and M. Trossen, *Interbotix_ros_manipulators*. [Online]. Available: https://github.com/Interbotix/interbotix_ros_manipulators.
- [44] Algoryx, *Iterativeprojectedconefriction*. [Online]. Available: https://www.algoryx.se/documentation/complete/agx/tags/latest/doc/UserManual/source/creating_objects.html?highlight=projectedconefriction#iterativeprojectedconefriction (visited on 05/13/2025).
- [45] Y. Lu, J. Williams, C. Lacoursière, and J. Trinkle, *Standard interface for data analysis of solvers in multibody dynamics*, 2013. arXiv: 2501.09747 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2501.09747>.
- [46] Algoryx, *Area based approach*. [Online]. Available: https://www.algoryx.se/documentation/complete/agx/tags/latest/doc/UserManual/source/overall_structure.html#area-based-approach (visited on 05/13/2025).
- [47] T. Berglund and M. Servin, *Slope stability analysis*, 2024.

A Results Frequent Evaluation

Below the measured parameters from the sparse evaluation, with $m = 0.05$, are shown. The *Reward* measurement is defined as in the ACT article [28], and is as follows: The reward is 0 when the box is lying on the table, and does not have contact with any of the robots. The reward is 1 when the box is lying on the table but one or two of the leftmost robot’s grippers are touching the box (attempted pick-up). The reward is 2 when the box’s has lost contact with the table, and has contact with one or both of the leftmost robot’s grippers (successful pick-up). The reward is 3 when the box is not touching the table, and is in contact with both the leftmost and rightmost robot’s grippers (attempted transfer). The reward is 4 when the box is not in contact with the table, and has contact with only the rightmost robot’s grippers (successful transfer).

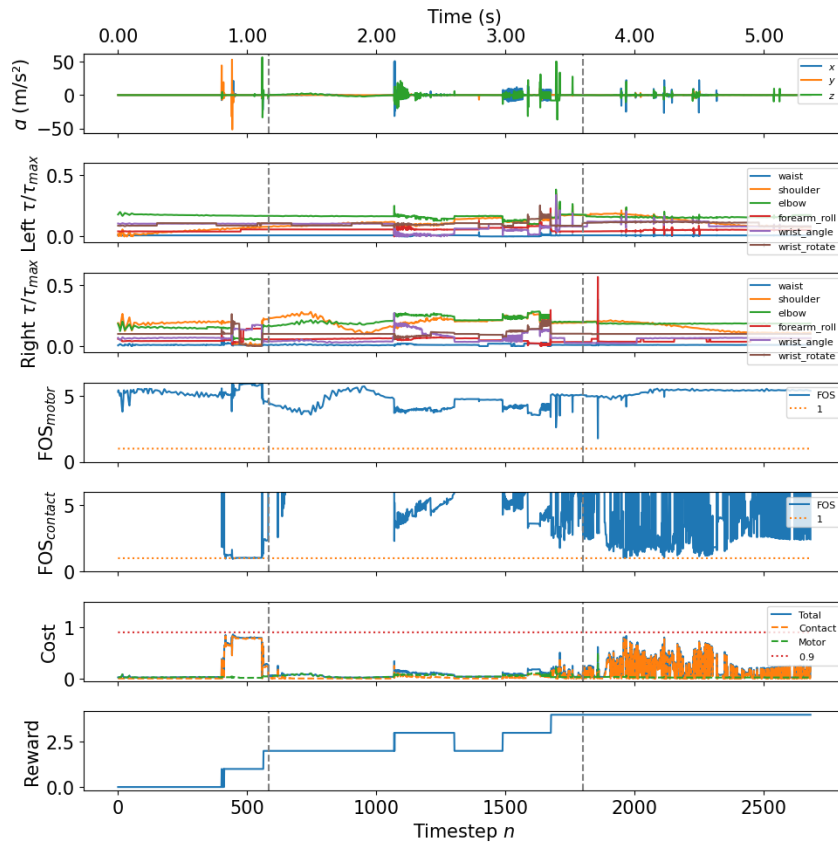


Figure 7: Measured parameters from the sparse evaluation. The gray vertical dotted lines show the moments are which the sparse evaluation was carried out.