# Action Hierarchies for Control

**Summary**
Integrate action hierarchies for control of the Sima Agent.

**Authors**: Ryan Faulkner
**Contributors**: @add_yourself
**Created**: Apr 4, 2024
**Self Link**: go/sima-ah

| Have you read/reviewed this design proposal? If so, please put your name below: |
|---|
| Usernames: Yulan Liu |
| WANT_LGTM: Yulan Liu Karol Gregor |

## Preamble

This proposal follows in the vein of others with links to Gemini Fine Tuning projects and the Self-Improvement Pipeline. In particular, this work would probably fall within the categories of recently proposed work on Gemini fine tuning. These ideas are not necessarily new and have been discussed over time by many people on the project; the aim here is to cast and circulate the approach described below more widely and to gather thoughts on it.

## Abstract

Our current agent lacks a coherent strategy for going from high level task description or goal, to natural language, to low level action in the keyboard and mouse. Some recent work in Sima has been done to tokenize our low level actions as a proof of concept using pre-trained language models. Also, action hierarchies in natural language have been used effectively in the RT-H agent and to generate improvements via human interventions. As Sima begins to integrate Gemini models we can consider how goals, instructions, and low-level actions can combine to produce a goal driven instructable agent making use of these ideas. This proposal's aim is to effectively use action hierarchies in the Sima agent.

## Motivation & Idea

We've explored action tokenisation in the SHA policy as a proof of concept developing a way to tokenize keyboard and mouse actions in language and model the policy by way of fine tuning pre-trained models over these action tokens. This work has been recently extended by fine-tuning Gemini in Tunelab yielding more promising results in modelling action tokens.

Much of this work was based on the VLA ("Vision-Language-Action") model proposed in the RT-2 paper which has since been built upon with action hierarchies in the RT-H model. Here we see that when a hierarchy of actions is available the trained agent gains on performance, can make better use of its context, and can generalise better. Further, a hierarchy can expose actions in natural language or **"language motions"**, or simple context independent actions expressed in natural language, which provide a means to include human corrections in the loop. This final point should lead to improved data quality and may fit nicely into the proposed self-improvement pipeline.

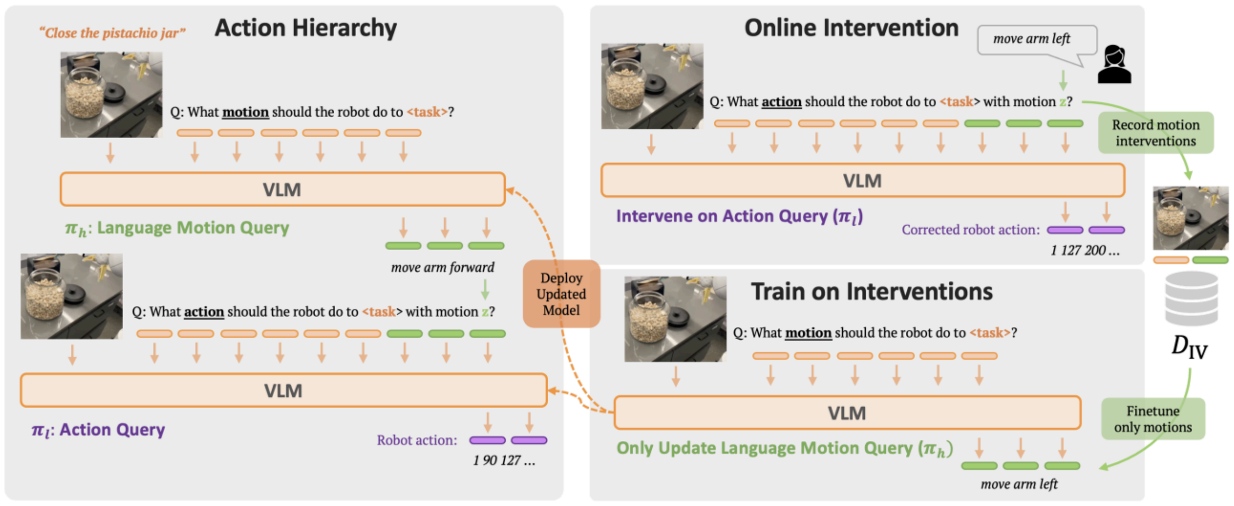
"Close the pistachio jar"
Action Hierarchy
Q: What motion should the robot do to <task>?
VLM
$\pi_h$: Language Motion Query
move arm forward
Q: What action should the robot do to <task> with motion z?
VLM
$\pi_l$: Action Query
Robot action:
1 90 127 ...
Deploy Updated Model
Online Intervention
move arm left
Q: What action should the robot do to <task> with motion z?
Record motion interventions
VLM
Intervene on Action Query ($\pi_l$)
Corrected robot action:
1 127 200 ...
Train on Interventions
Q: What motion should the robot do to <task>?
$D_{IV}$
VLM
Finetune only motions
Only Update Language Motion Query ($\pi_h$)
move arm left

Figure: Examples of the two-level hierarchy used in RT-H. On the left, language motion then actions are queried. On the right is an example of how human corrections can be gathered and used in fine tuning.

**The proposal here is to use action hierarchies in Sima.** This could be considered a way to think of our agent modules, the Nexus and the Short Horizon Actor, along the levels of this hierarchy with a single model backbone. Here are some of the things that we might gain from this approach:

- Gemini provides a natural interface to output tokenized actions as has already been demonstrated in the Tunelab work.
- Language gives us a way to compress actions. Given context (ie. environment state) with language we can encode language motions in fewer bits than low level actions. There is also higher correlation among language motions via high level concepts that abstract action sequences (navigation, interaction, etc.).
- Language motions can reference context ("Go to the red cube").
- The action hierarchy establishes a causal relationship downward. This leads to factored distribution of actions that should be easier to model, although this will depend on how we constrain higher level actions ("language motions"). : $\boldsymbol{\pi(\text{action}, z|\text{obs}, \text{task}) = \pi_h(z|o, g)\pi_l(\text{action}|\text{obs}, \text{task}, z)}$ [where z is a language motion (ie. tokens), $\boldsymbol{\pi_h}$ is the high level policy, $\boldsymbol{\pi_l}$ the low level policy]
- Language motions provide a natural way to describe behaviour that is accessible to humans.
  - Provides a means for us to introduce human corrections.
- Language motions are arguably more relevant to us where we are subject to a great deal of context diversity and a large action space than in the controlled robotic settings since modelling a flat policy alone will be harder. This is demonstrable considering the same language motions will have valid context dependent meaning across different Sima domains (e.g. "move forward").
- Past language motions could function as memory vectors (possibly coupled with their context latents)
- We can begin to free ourselves from thinking about actions as dependent on time steps. Ultimately we should not need any time based boundaries (except causality) when considering taking action, it's all tokens.
- Integration into the Self-Improving Pipeline.

Finally, this should be done within the framework of Gemini finetuning which will mean that having a working finetuning setup is a prerequisite.

## Approach

This section includes a general picture of what the implementation could look like. It is purposely kept general enough so that finer details can be fleshed out once the approach becomes clearer and our Gemini fine tuning ecosystem has been built.

The top level workstreams breakdown into:

1. Develop a fine tuning pipeline for the action hierarchy.

2. Generating language motions in the dataset. We already have a way to generate action tokens.
3. Setup for human intervention and integration into the Self-Improving Pipeline

## Hierarchical Agent & Training

Following the figure above, the "VLM" is Gemini, this form aligns well with our current conception of Nexus + SHA. The bottom level of the hierarchy is similar to what's already been done in Tunelab, that is to generate Sima (tokenized) actions from a query of the context conditioned on the language motion. The upper level of the hierarchy consists of a query of the context conditioned on the task (or goal).

Of course, it wouldn't be difficult to implement this approach with the existing agent infrastructure, however given that we already intend to explore the SHA with Gemini, and that language pre-training in this case will be important, and will likely be most fruitful to explore with Gemini models.

To train these models we will need to learn a low level action policy (query with language motions, $\boldsymbol{\pi_l}$) and a high level one (query with goals/task description, $\boldsymbol{\pi_h}$). We might think of $\boldsymbol{\pi_h}$ as something similar to Nexus in this case, in which case this proposal largely reduces to a strategy to combine Nexus + SHA where we model the low level policy in the loop conditional on the language motion: $\boldsymbol{\pi_l(\text{action}|\text{obs, task, } z)}$. The policies are co-trained over language motion queries and action queries. We may wish to use a smaller model for $\boldsymbol{\pi_l}$ as has been considered in our typical setup, however depending on the length of the action horizon we may be able to get away with larger models for inference. This could be investigated empirically.

## Language Motions

This will likely be the trickiest part to get right and will almost certainly require some data preparation. More importantly, we don't yet have a clear idea of how these natural language instructions should look. For instance, we will certainly want enough variability in the language in order to accurately model the underlying actions. However, we also will want these to be simple enough to ensure that optimisation is easier. There will also be dependencies on action sequence length and, of course, by game domain. We will also have to deal with boundaries around natural language actions as these may apply over variable sequence lengths.

Following are a few ways that we might obtain language motions.

**Use Existing instructions:** We would have the option of taking our existing setter instructions or annotations.

Pros:

- Already in the data.
- In cases where they've been collected by raters they will reflect natural language (although maybe not!)

Cons:

- No way to guarantee consistency across the set.
- They may be inaccurate, especially in x-worlds
- They may run on for far too long and be too abstract

**Additional Human annotation:** Run separate collections with raters to build a dataset of language motions.

Pros:

- We can instruct the raters on frequency and language to use
- They should reflect natural language

Cons:

- Eats into rater resources and takes time
- Results aren't guaranteed to be consistent, raters make mistakes
- There may be a learning curve before we get good data
- Need to augment existing datasets or write new ones.

**Gemini annotation:** We can also annotate the language motions with an annotation model.

Pros:

- We have an annotation pipeline in place that can already be used.
- Gemini models are improving and we could bootstrap off of this cycle in improving our sets.

Cons:

- If we need to fine tune to get reasonable results this means we need to get that pipeline right.
- The annotations may be sub-par depending on the model and consistency isn't necessarily guaranteed.
- Need to augment existing datasets or write new ones.

**Algorithmically:** This is the approach RT-H takes. The idea is to determine the language motion from the keyboard and mouse actions algorithmically. We already do this for low level action tokens.

Pros:

- Consistent language motions.
- We can engineer the language motions to better ensure correctness.
- Should be straightforward to run a beam pipeline.

Cons:

- Output may be less like natural language (although probably manageable)
- Will need to ensure all actions are represent

## Human Corrections

Language motions may unlock the capability of generating human-in-the-loop interventions. I will only sketch out how this will look but this component would no doubt involve a bit of technical work to build the interface.

This would function similarly to our agent playtesting where now we would use the model at inference time to expose its high level actions. A human rater would observe these actions and intervene if they are convinced that, given the task description, the agent has chosen the incorrect behaviour. The human would then intervene with a key-press (for instance) and type in the new language motion which would overwrite the existing one which will have a behavioural effect on the agent changing the outcome of the task attempt. These episodes or spans would be recorded in addition to the core data with which we can further fine tune the model (see figure).

From here we can grow our dataset with these interventions which may form a part of the self-improving pipeline.

## Experiments

Below are some proposed experiments this some of which are aligned with the RT-H metrics here but we can certainly expand these:

- **Action Sequence Length:** What is a reasonable action horizon for the language motions? Should the horizon length be fixed or variable?
- **Language motions:** Aim to get a sense of different strategies for making the natural language actions as detailed in the section above.
- **Performance:** Does it improve scores over the non-hierarchical baseline?
- **Context:** Are language motions contextual to the scene? Does the same language motion generalise well to different settings? Domains even?
- **Generalisation:** Do action hierarchies improve agent responsiveness in out-of-distribution settings or across domains?
- **Corrections:** If we train on corrected trajectories do we see a benefit? How much?

Evaluation and metrics can be defined on the Gemini production infrastructure.

## Risks & Challenges

- This proposal deals largely with a principled approach on how to develop our policies and doesn't tie into our existing evaluation pipeline. I think that we would need to aim to understand how well these agents play sooner rather than later and ideally integrate into tagline and existing metrics to get our bearings on how this approach compares to the baseline agents. That said, this will be the same issue for all Gemini fine tuned agents.
- The RT-X work is done in robotics domains. Sima domains will likely bring their own challenges as the environment diversity will likely be much higher. (*however this may also be a strength*)
- We need to be careful of inference time in the model.
- There is potentially significant data collection work. We would need to figure out how much new data we need and devise a way to get clarity on the best data collection strategy efficiently.
- Building up intervention datasets could take time and some infrastructure work would be needed.
- ...