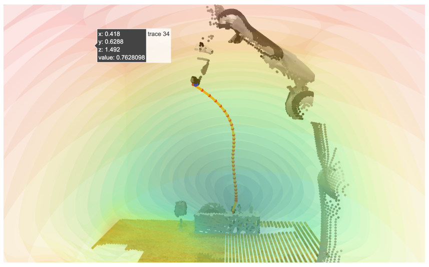
SteerPoser: Guiding Robots towards Preferred Behavior

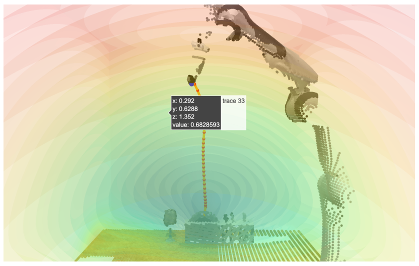
Aakash Aanegola
Columbia University
aa5506@columbia.edu

Danelle Tuchman
Columbia University
dst2161@columbia.edu

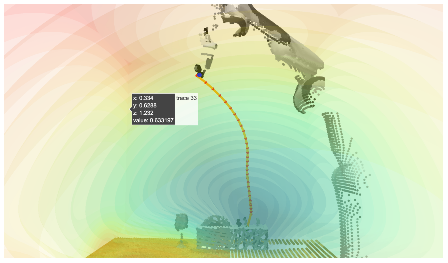
Sammy Agrawal
Columbia University
ssa2206@columbia.edu



(a) **User 1:** moderately healthy (Granola)



(b) **User 2:** very healthy (Apple)



(c) **User 3:** unhealthy (Chocolate)

Fig. 1: SteerPoser guides LLM outputs based on user preferences given the same input prompt, ”Grab me a snack.” The resulting trajectories and selected objects reflect varying user preferences—ranging from healthy to unhealthy—demonstrating that activation steering can effectively influence LLM-driven robotic planning and manipulation.

Abstract—**In this project, we propose SteerPoser, an end-to-end system to structurally influence the planning behavior of Large Language Models (LLMs) using causal interventions. Taking heavy inspiration from works like VoxPoser and ReKep, we extend robotic manipulation pipelines in which LLM guided task planning defines a sequence of actions to execute high level tasks. In the face of under specified or ambiguous task instructions, we investigate how well activation steering methods succeed in influencing a robot's actions towards preferred outcomes. We also demonstrate the viability of learning to accommodate complex user preferences over time via recorded human feedback. We train several activation steering vectors for semantically meaningful concepts such as healthiness, environmental consciousness, and object safety. SteerPoser is the result of adding activation steering to an altered version of VoxPoser. We used activation steering within the LLM to teach the model to select the user's favorite foods when prompted to grab a snack. Based on how much they valued eating healthy food in the past, we steer the robot to grab a certain food item. We evaluated this system based on successful runtime execution and semantic preference alignment. SteerPoser does better than VoxPoser at fulfilling user-aligned completions with a slight drop in success rate due to a less powerful LLM model. All code for this project is available at our public Github Repository, Supplemental material includes the datasets in steer-data/datasets and additional plots in ‘activation steering figures‘.**

I. Introduction

Large Language Models (LLMs) are increasingly becoming integrated with robotics. As an interface for human interaction, language is an essential medium for conveying desired actions and imbuing robots with “common sense”. LLMs show promise in modeling human-like behaviors and acting as a human-machine interface for robotic interactions. However, this paradigm introduces its own set of challenges. Although increasingly competent, LLMs may act unpredictably and might not always be aligned with what users want. Furthermore, users may themselves violate their own preferences or provide unclear instructions. Using LLMs for robotics tasks (in the embodied setting) also raises several safety concerns. Clever prompt engineering can be used to set implicit bounds but can't defend against adversarial inputs from users. Fortunately, this problems extend far beyond the domain of robotics; increasing the trustworthiness and reliability of Language Models is an active area of research. In this work, we introduce ”SteerPoser”, an investigation into integrating interpretability and behavior modification methods from the language domain into existing robotics pipelines. Specifically, we add activation steering to the LLM to guide both the outcome of a robotic task and the method by which the task is completed. Through SteerPoser, we demonstrate that steering increases the robot's ability to complete a task based on user preferences, unique personality traits, and recorded user patterns. SteerPoser was heavily influenced by VoxPoser and Rekep, which use LLMs for well-defined, short horizon tasks. SteerPoser is unique because it has proven successful in more ambiguous and open-ended tasks that have a wider variety of technically successful but ultimately suboptimal outcomes. To implement SteerPoser, we set up a simulation environment using RL Bench, the

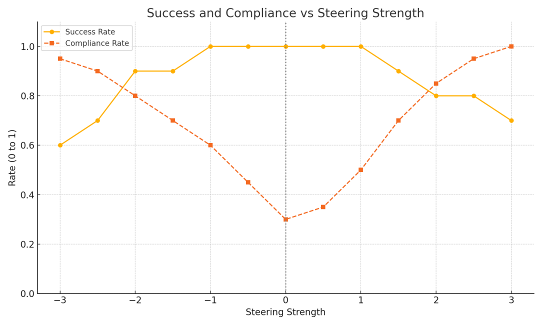


Fig. 2: Success vs. Compliance for different steering strengths. As expected, stronger steering generally leads to lower success rates due to model collapse and malformed LLM output.

environment simulator that VoxPoser uses. Somewhat limited by the scope of the simulation, we assigned four random objects as ”chocolate,” ”apple,” ”granola,” and ”pasta.” We tested activation steering in two ways: the first was a general steering towards healthier options, and the second was based on user preference. We trained the robot using different user preferences for each of the foods, and then gave the robot the ambiguous task of ”getting a snack.” Depending on which user was asking, the robot grabbed different items that represent the correct food choice. Our success rates along with the compliance ratio with user preference is shown in 2.

II. Related Work

A. LLM Planning

Prior works have demonstrated the ability of LLMs to generate task plans, infer spatial constraints, and reason about affordances from multimodal inputs. However, these methods struggle to deal with issues of ambiguity, safety, and user under-specification; primarily because they are not designed or intended to do so. Nevertheless, these are critical considerations before the widespread deployment of human-facing robots. ReKep [1] introduces a framework for encoding robotic manipulation tasks as relational key point constraints, leveraging vision-language models to specify constraints. By structuring tasks as a sequence of optimizable constraints in SE(3), ReKep provides a flexible and efficient means of generating task plans. Still, it remains susceptible to the limitations of its underlying LLM, which can lead to unintended behavior due to hallucinated constraints [2]. Extensive prior work has explored LLMs for robotic task planning and execution [1, 3–5]. [6] presents a multi-layer LLM approach for enhancing robotic task execution, emphasizing structured planning. More recently, multimodal LLMs have been applied to end-to-end robotic manipulation [7], incorporating vision and language priors for task inference. However, these approaches largely rely on LLMs’ intrinsic reasoning capabilities, making them prone to hallucinations that may lead to unsafe or suboptimal robotic behaviors.

VoxPoser addresses leveraging the breadth of LLMs internalized knowledge towards the fine-grained action level for robots, without requiring extensive data collection or manual designs. This is achieved through the exposure of a code-generating planner which queries and acts upon its environment through a programmatic API. The API abstracts away much of the underlying physics and low level dynamics as to avoid burdening the language model with extraneous information. VoxPoser intuits that LLMs are great at decomposing instructions into tangible sub-tasks. They can also infer language-conditioned affordances and constraints, which can easily be represented as dense 3D voxel maps. Through the API, these maps provide a task-aware reward that guides trajectory planning towards important locations in observation space. Thus, VoxPoser presents a zero-shot framework that leverages language instructions as feedback to indirectly guide potentially complex optimization problem better suited for deterministic methods. Both VoxPoser and its more sophisticated follow-up work, SteerKep, provide elegant solutions for clearly defining the role of the language model within the broader system. VoxPoser was tested on its performance on everyday manipulation tasks both in simulation and in the real world. It was trained on specific tasks like ”open a drawer,” but has proven successful in randomized tasks as well, as long as they are still specific and tell the robot which objects to interact with and how[5].

Another notable system, LM-SymOpt, integrates planning for complicated and intricate tasks with low motion capabilities. It uses an innovative language-guided symbolic task planning framework with optimization to help a robot adapt to unseen tasks. This method includes deriving symbolic representations from the natural language instructions to reduce the planning search space since actions are represented as high level symbols. LM-SymOpt utilizes LLMs to assess the action probability for each task and implements a weighted random sampling method to generate candidate plans. The likelihood of each task being able to be completed is assessed through the symbolic reasoning, and trajectory optimization for finding the optimal planning method is used to find the cost efficiency of each method. LMSymOpt has proven to be more successful than other LLM based methods [8].

Each of these previous works are specifically successful at utilizing LLMs to guide robotic motion for well defined, short horizon tasks [1] [5] [8]. The task decomposition is largely left to the innate reasoning capabilities of the underlying LLM without much investigation into what guides model output. Furthermore, none are specifically tested on ambiguous tasks in which the correct sub-task sequence is less clear. What would happen if you asked these systems to clean your entire kitchen? What is the correct task sequence there? How can we move beyond the fixed environment-defined task success metrics common to all three setups?

B. Activation Steering

To dig deeper into these answers, we conducted a literature review of how the general uncertainty around LLM reasoning outcomes is addressed in other domains. In addition to methods that retrain model weights, such as supervised finetuning

Steering Strength	Observed Failure Modes
+2 (Strong Healthiness)	• Generates ambiguous (logical) 'ors' in the object to grasp. This could be due to prompt inadequacy or model size. • Code contains repeated calls to `grasp(object)` without any additional logic. • Repeats code meaninglessly in loops or statements with no effect. • Hallucinates objects not present in the scene and commands that weren't provided.
+1 (Weak Healthiness)	No significant failure modes observed. Code is generally valid and grasps are accurate in most runs.
0 (No Preference)	Baseline behavior — successful and neutral. No compliance measured; no notable failure modes.
-2 (Strong Unhealthy Preference)	• Similar issues as +2: hallucinated objects and grasp ambiguity. • Frequent repetition of meaningless code. • Multiple identical grasp commands with no clear plan structure.

TABLE I: Failure modes observed across different steering strengths over 10 runs. Stronger steering tends to degrade planning/code quality.

and Reinforcement Learning with Human Feedback, the LLM safety community has also developed methods to alter LLM behavior without retraining in any way. A new trend in AI safety and interpretability research is the use of activation steering [9] to override user prompts and causally elicit desired behaviors. Rather than relying on user-provided contexts or prompt injections, steering intervenes directly on the internal model weights, offering the potential of greater resiliency. Steering traces its routes to the field of Mechanistic Interpretability, in which fastidious analysis of transformer weights and activations is performed in the hopes of finding concept representations and "circuits" for computation [10]. Mechanistic Interpretability has enjoyed some notable successes, namely reverse engineering non-trivial algorithms employed by transformers [11] and finding seemingly robust concept representations "in the wild" [12]. However, there are still serious debates as to whether or not there is a scalable path forward for producing robust interpretability. Many existing methods are predicated on the hypothesis that transformer hidden layer activations can be decomposed into linear combinations of human-understandable "features"; this is broadly referred to as the Linear Representation Hypothesis [13]. Figure 3 provides a simple visual example of how activation steering influences hidden states. When steering towards love, the model will choose to love or hate the object in front of it [14]. To isolate a specific feature, datasets of contrastive prompts in which the feature direction is present or absent are used. Although the assumption of linearity yields great tractability, the hypothesis is likely not true in full generality, as non-linearity is crucial to the universality properties of neural networks. Indeed, more recent works have either demonstrated non-linear geometry in representation space [15] or found gaps in the reliability of steering vectors formed through linear methods [16]. Regardless, enough work has been done to validate that deep learning architectures are not strict black boxes and that some sort of concept representation and manipulation is determinable from existing architectures. Exceedingly simple methods to selectively modify model activations often produce qualitatively reasonable results. Regardless of the full robustness of concept representations, it becomes possible to steer the model's outputs by then amplifying or suppressing concepts during inference. To the best of our knowledge, there remains a gap in the current literature of applying this methodology to robotic manipulation and planning.

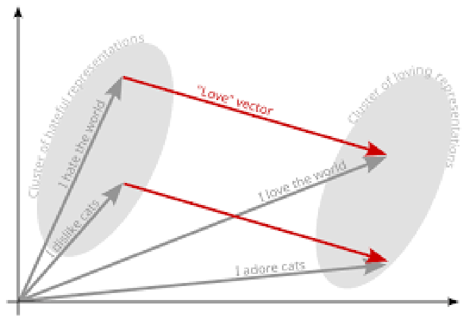


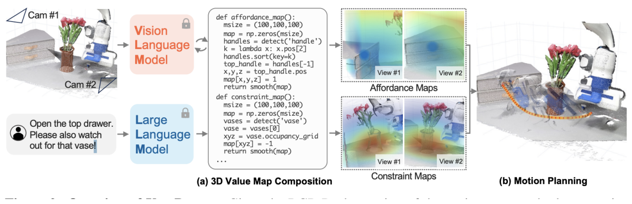
Fig. 3: An example of how steering towards love will influence a decision.

III. Methods

A. Challenges with Simulation Environment

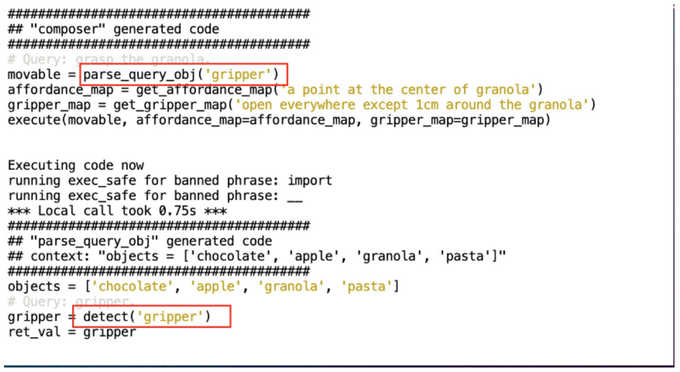
Many simulation setup attempts were tried and tested before the creation of SteerPoser. A key issue was our limited access to GPU compute, meaning that we were frequently migrating our codebase to varied setups to retry getting a successful rendering. The initial intention was to use OmniGibson and enhance the ReKep pipeline ("SteerKep"). There were several motivations for this. Relational keypoints embedded as coded up loss functions are a dynamic method of constraint formulation. The space of possible preferences that can be written as code is extremely large and extends beyond the basic object level primitives for which they are currently used in existing work. For example, instead of just preventing dropping, imagine an explicit trajectory penalty for "unfair" actions. In comparison, the affordance value maps used by VoxPoser are quite limited. By producing real-number value maps directly over physical space, more abstract trajectory preferences cannot really be specified. Secondly, although mechanistic interpretability traces its modern roots to work done in Computer Vision, the current paradigm is to align the behavior of LLMs. Thus, the most sophisticated steering methods are only well developed for text-based models. But for robotics specifically, it is of special interest to assess how well these hidden layer interventions generalize to multi modal systems (Vision-Language or even Vision-Language-Action). Because the VoxPoser planner is unimodal (in reference to the clean layers of separation described above), questions of multi-modal concept generalization (what is a *visual* representation of the ”fairness” concept as applied to trajectory planning) remain unanswered despite being a strong initial motivation. Two main issues prevented these avenues from being explored. First was because of our setup abilities. Although OmniGibson and Nvidia IsaacSim were successfully downloaded, running in a headless environment required virtual framebuffer libraries not allowed on our remote HPC system. Significant Operating System compatibility and missing QT library functionality raised a host of issues. While these was eventually resolved through containerization, additional issues prevented the rendering of ReKep specifically. Because of these issues, we applied for free credits on a AI cloud platform called RunPod [17]. Although RunPod could likely have fixed these issues given more time, a more fundamental limitation was the lack of emphasis on multimodality in existing research. Our early experiments in steering vision language models raised doubts about how effectively existing tools could handle transformer cross attention between image and text embeddings. Limiting ourselves to steering self-attention layers to bypass cross attention produced insufficient change in behavior as well as degradation in base model performance. The resulting time limitation of all these problems prompted switching to the less robust, but far simpler VoxPoser code base. RLBench is a far easier simulation environment to get working. The multimodal steering issues are also likely fixable with the deeper understanding of steering we now have, but proved intractable within the semester time frame. We still believe that the integration of steering into Voxposer has produced interesting outcomes; nevertheless, we wanted to note that alternative avenues were tried and abandoned.

B. Extensions to RLBench and VoxPoser Environment



```
def affordance_map():
  msize = (100,100,100)
  map = np.zeros(msize)
  handles = detect('handle')
  k = lambda x: x.pos[2]
  handles.sort(key=k)
  top_handle = handles[-1]
  x,y,z = top_handle.pos
  map[x,y,z] = 1
  return smooth(map)
def constraint_map():
  msize = (100,100,100)
  map = np.zeros(msize)
  vases = detect('vase')
  vase = vases[0]
  xyz = vase.occupancy_grid
  map[xyz] = -1
  return smooth(map)
...
```

Fig. 4: In this example steering will influence the output to discussing weddings even if the initial prompt has nothing to do with weddings since both steering vectors express opinions about discussing weddings.



```
########################################
## "composer" generated code
########################################
# Query: grasp the granola.
movable = parse_query_obj('gripper')
affordance_map = get_affordance_map('a point at the center of granola')
gripper_map = get_gripper_map('open everywhere except 1cm around the granola')
execute(movable, affordance_map=affordance_map, gripper_map=gripper_map)


Executing code now
running exec_safe for banned phrase: import
running exec_safe for banned phrase: __
*** Local call took 0.75s ***
########################################
## "parse_query_obj" generated code
## context: "objects = ['chocolate', 'apple', 'granola', 'pasta']"
########################################
objects = ['chocolate', 'apple', 'granola', 'pasta']
# Query: gripper.
gripper = detect('gripper')
ret_val = gripper
```

Fig. 5: In this example steering will influence the output to discussing weddings even if the initial prompt has nothing to do with weddings since both steering vectors express opinions about discussing weddings.

We built on VoxPoser [5], which defines a set of Language Model Programs (LMPs) using few-shot code generation with exposed API calls and fixed base prompts to generate example code snippets for high-level tasks. In 4, which is Figure 2 of the original paper, a visualization of the VoxPoser pipeline is provided. This design makes VoxPoser naturally compatible with activation steering. To adapt it for our purposes, we replaced OpenAI API calls with a locally hosted open-source LLM and significantly refactored the codebase to support modular swapping of individual Language Model Programs (LMPs). We also defined an additional LMP to specifically make use of steering, but ended up not using it because of instability. An example of the code generation sequence in our work is provided in 5. This allowed us to substitute standard LMPs with versions that use the steered model. While our initial plan was to create new tasks within the RLBench framework, this proved infeasible in a headless simulation setup. Instead, we repurposed existing tasks by relabeling objects as food items with varying degrees of healthiness. This enabled the robot to respond to prompts like ”Grab me a snack” by selecting from pre-existing objects redefined to reflect different user preferences. To accommodate the smaller size and limitations of our 7B model compared to GPT-4, we also increased the amount of context provided and introduced LLaMA-specific regex-based postprocessing to ensure successful code execution. VoxPoser [5] describes the LMP setup in detail and we made no major modifications to the overall pipeline other than what is descibed here.

C. Creation of Steering Vectors

Recall that activation steering is based on the Linear Representation Hypothesis; when a concept is necessary for the inference performed by the model (i.e. if it is inherent in the provided prompt), the corresponding latent feature should be present in the hidden activations of the model. We refer to Figure 6 to demonstrate the basic idea of steering vectors; in our work, we primarily focus on steering the idea of healthiness instead of ”loving to talk about weddings” or

”hating to talk about weddings [18]. For our steering vectors,

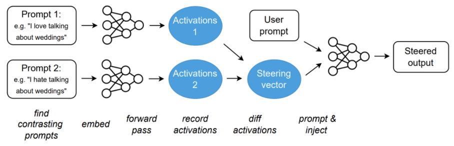


Fig. 6: In this example steering will influence the output to discussing weddings even if the initial prompt has nothing to do with weddings since both steering vectors express opinions about discussing weddings.

we extended work done at IBM by Lee et al [19]. They open sourced their research and provided an open source code base. Our fork and extension of their work is linked as a submodule in our own repository. The broad steps to creating steering vectors is as follows:

1) Construct a contrastive dataset $\mathcal{D}^+, D^-$ in which the desired behavior is elicited versus not elicited. An individual contrastive pair consists of a query, a compliant response, and a non-compliant response. For example, a query could read: ”After a long day at work, she opened the fridge and reached for...”. Compliant behavior is ”a bowl of fruit and Greek yogurt” while the noncompliant behavior is ”leftover cheeseburger and fries”. From N examples of queries and responses, N^2 pairwise query-responses can be created (assuming the query and response are relatively separable). All training datasets can be found in our supplementary material as well as in the public codebase on Github.
2) For stored LLM model $\mathcal{M}$, we extract the hidden layers during the forward pass for all L model layers. $H(D) = \{(h_1, h_2, ...h_L)_i | \forall i \in D\}$. Define H^+ to be the hidden states for compliant continuations and H^- to be non-compliant.
3) After performing mean-centering across all hidden state layers (positive and negative datasets), extract the first principal component of the combined dataset. The idea here is that the activation direction of greatest variance between two clusters of positive and negative samples will be the most robust axis of distinction. The result is a steering vector at every layer of the model, i.e.

$$SV = \{PCA(\{h_l^+ - \mu_l\} \cup \{h_l^- - \mu_l\}) | \forall l \in [L]\}$$

For testing, we constructed a wide variety of steering vectors. We tried isolating the distinction between blunt and sharp objects, environmental consciousness, healthiness, and general robotic safety. We experimented with various model families and sizes, ultimately converging on the Nous-Hermes Llama 27B instruction-tuned series. Smaller models proved harder to steer and were more prone to mode collapse, likely due to their shallower architectures and reduced representational capacity. Hermes, in contrast, offered long, coherent responses, a lower hallucination rate, and lacked the restrictive safety layers present in OpenAI-aligned models—benefits for probing edge cases in safety-critical behaviors. The IBM library, while not cutting-edge, was cleanly designed, well-suited for research prototyping, and easy to extend to new model backends, making it an ideal foundation for our activation steering framework.

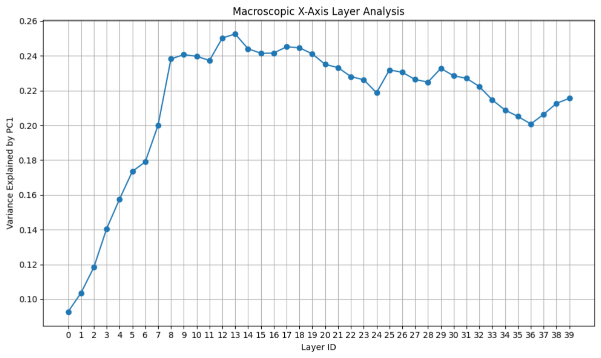


Fig. 7: Variance explained by first principal component in embedding space. Clear absence of concept formation in early layers.

D. Learning Steering Strengths from User Feedback

We use the simplest method of applying steering vectors, which is simply to add the learned vector to the corresponding hidden layer at inference [9].

$$h' \leftarrow h + \alpha \cdot v$$

The sign of the weighting term α determines whether the concept is amplified or suppressed. Too much amplification and the steering overrides the ability of the model to actually produce legible English (intervened upon activations become out of distribution). Too little results in insufficient change in behavior. Empirical trial and error on our part determined the range of steering strength for which model performance does not significantly degrade is around $[-2, 2]$. Further intervention results in garbage outputs. While initial experiments were done with manually chosen α, we then asked whether this steering strength parameter could be learned from data. Specifically, whether the degree of steering could be dialed up or down based on user preferences.

Can future robotic systems calibrate their assistance in an online learning fashion? Can this calibration map to user-approved behaviors?. We essentially treat steering vector strengths as tunable knobs that can either be explicitly set or implicitly learned to match a robotic system to the unique preferences of its user. Our initial idea was to learn a direct steering vector encoding user preferences. Compliant continuations could be instances in which the robotic system performed to the user’s liking while non-compliant continuations would be distasteful assistance. However, further literature review of the failure modes of steering were sufficiently dissuading to this naive approach. Firstly, the concept extrapolation method described in previous sections is highly sensitive to the contrastive dataset used. It is crucial for contrastive pairs

TABLE II: Steering statistics across layers for different user preferences.

(a) User 1 – very healthy

Layer	Pos.	Neg.	Score	Strength
29	4.08	0.20	0.91	1.82
30	4.26	0.24	0.90	1.79
31	4.60	0.41	0.84	1.67
32	4.86	0.56	0.79	1.59
33	5.34	0.90	0.71	1.43
34	5.16	0.57	0.80	1.60
35	5.97	1.11	0.69	1.37
36	7.56	2.48	0.51	1.01
37	9.21	3.45	0.46	0.91
38	13.74	7.22	0.31	0.62
39	34.84	6.62	0.68	1.36
Avg	9.06	2.16	0.69	1.38

(b) User 2 – moderately healthy

Layer	Pos.	Neg.	Score	Strength
29	2.58	1.07	0.41	0.83
30	2.69	1.17	0.39	0.78
31	2.90	1.35	0.36	0.73
32	3.08	1.51	0.34	0.68
33	3.48	1.85	0.31	0.61
34	3.22	1.54	0.35	0.71
35	3.99	2.15	0.30	0.60
36	5.34	3.46	0.21	0.43
37	6.50	4.30	0.20	0.41
38	10.45	7.81	0.14	0.29
39	15.87	-1.16	1.00	2.00
Avg	5.46	2.28	0.37	0.73

(c) User 3 – unhealthy

Layer	Pos.	Neg.	Score	Strength
29	0.90	3.64	-0.60	-1.21
30	0.96	3.78	-0.60	-1.19
31	1.16	4.11	-0.56	-1.12
32	1.30	4.34	-0.54	-1.08
33	1.62	4.77	-0.49	-0.99
34	1.27	4.53	-0.56	-1.13
35	1.79	5.25	-0.49	-0.98
36	3.11	6.75	-0.37	-0.74
37	4.07	8.22	-0.34	-0.67
38	7.55	12.36	-0.24	-0.48
39	-9.00	18.77	-1.00	-2.00
Avg	1.34	6.96	-0.53	-1.05

to isolate a well-defined concept in activation space. Steering most often fails to generalize because of spurious correlations between $\mathcal{D}^+$ and $\mathcal{D}^-$. Letting the basis of steering be open-ended user interactions seemed to be a recipe for brittleness. This is especially the case with the variety of tasks potentially facing household robotic systems. Users could ask about food in one instance, cleaning in another, and shopping in a third. There is likely no coherent vector to be extracted in such scenarios, unless online classification divided feedback into discrete categories.

Instead, we imagine a use case where a library of well defined and reasonably robust steering vectors have already been designed in controlled settings. With such a litany of steering vectors available, we can instead calibrate user preferences by asking: *to what extent is this pre-defined concept present in what the user has liked in the past*. Essentially, we can use steering vectors as a similarity metric between user preferences and the original dataset used to define a concept. Our approach is rather straightforward; to our knowledge, this has not been attempted before.

1) Similar to before, define a dataset of user interactions. Instead of expressing a coherent concept, queries are tasks requested by the user and continuations are actions performed by the robot. Based on user feedback (either on what the robot actually did or by querying the user on how they *would* respond to certain actions, separate the continuations into positive and negative ratings $\mathcal{R}^+, \mathcal{R}^-$. It is crucial to differentiate the ratings dataset from the steering vector dataset. $\mathcal{R}^+$ could have a positively correlated, negatively correlated, or completely independent relationship to $\mathcal{D}^+$ depending on what the user asks the robot to do and whether they are systemically biased towards the concept or not. For example, users might not have any clear health preference.
2) Using the same methods as above, collect the hidden states H_R^+, H_R^-. Compute the average similarity between the hidden states and the predefined steering vector $V = (v_1, ..., v_L)$ independently at each layer.

$$\hat{p}_l = \frac{1}{|\mathcal{R}^+|} \sum_{h_l \in [H_R^+]_l} sim(h_l, v_l)$$

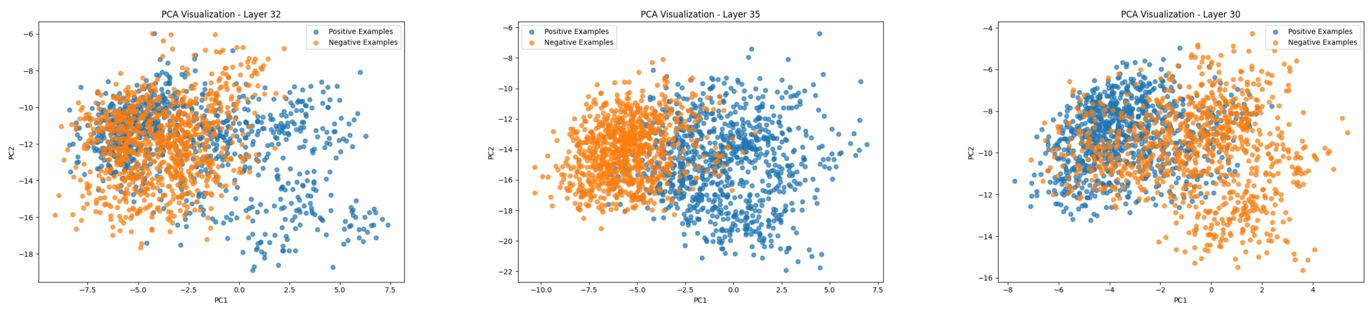
$$\hat{n}_l ;= \frac{1}{|\mathcal{R}^-|} \sum_{h_l \in [H_R^+]_l} sim(h_l, v_l)$$

We use projection as our similarity metric, i.e. $sim(h, v) = h \cdot \frac{v}{||v||}$. Thus, p and n independently capture how much the steering vector concept is present in robotic actions the user likes and dislikes. A future extension could go beyond a binary weighting system, where ratings out of 10 or "5 stars" could weight different ratings accordingly. This would be trivial to implement but was omitted for simplicity.
3) To normalize our ratings into a standard scale, we scale the relative weightings for positively rated and negatively rated states across model layers: $r := 1L \sum_{l \in L} \hat{p}_l - \hat{n}_l |\hat{p}_l| + |\hat{n}_l| + \epsilon$. Other normalization methods were considered, such as using sigmoid scaling but this qualitatively aligned most with what seemed reasonable. Sigmoid scaling tended to under-exaggerate sentiment differences at the extremes. Our scaling method brings the relative comparison into the range $[-1, 1]$ which can be arbitrarily transformed to match desired steering ranges. Since we determined that $[-2, 2]$ is the range of valid α, we simply double the normalized rating to get the steering strength.

Since we only tested with one concept (healthiness), this ended up not being necessary, but we imagine easily implemented extensions in the case of multi-concept steering. We can determine a rating to be irrelevant to a specific category (Food, Sustainability, etc) if the similarity falls below a certain threshold. Thus we can assign rating relevance for our catalog of predefined steering vectors and simultaneously get a vector of strengths $(\alpha_1, \alpha_2, ..., \alpha_k)$ for steering vectors $(v_1, v_2, ..., v_k)$.

IV. Experiments

To validate our hypotheses, we conducted several experiments - from the activation steering front as well as the impact on VoxPoser. This section is divided into two parts that focus on these (independent) components.



(a) User 1 – very healthy (b) User 2 – moderately healthy (c) User 3 – unhealthy

Fig. 8: Scatterplots of PCA projections for positive and negative training examples. Positive values indicate agreement or the presence of the ”health” concept in the embeddings.

A. Activation Steering Experiments

We were excited to steer the robot to be health conscious, so when dealing given an ambiguous task like ”grab me a snack” it would grab a healthier snack. Activation steering in small models is highly sensitive to hyperparameters, requiring extensive experimentation to determine both the appropriate strength of steering vectors and the optimal layers for intervention. As shown in Figure 7, mid-to-late layers contribute more significantly to effective steering—consistent with the intuition that higher-order concepts emerge only in later stages of model reasoning. While steering in the latest layers can slightly reduce performance, applying steering too early often leads to model collapse and qualitatively degraded outputs. As a result, we focused our efforts on steering later layers where the tradeoff between control and stability was most favorable. Note that there aren't good metrics to capture this effect quantitatively and we had to reply on qualitative analyses of LLM outputs.

B. Steering Based on User Preferences

As shown in Figure 1, activation steering enables encoding user-specific preferences into the LLM's behavior, allowing the same prompt to yield different outputs tailored to each user. We further analyze this effect in Figure 2, which illustrates the tradeoff between task success and preference compliance across varying steering strengths. In our experiment, we pretended a singular household has three different occupants (Sammy, Danelle, and Aakash) who each have different preferences for how healthy they are eating. Recall our kitchen has four possible food options: ”chocolate,” ”apple,” ”granola,” and ”pasta. Depending on who the robot is serving, the ideal snack choice is different. Healthy Sammy prefers an apple; sugar junkie Aakash wants chocolate; the middle path Danelle prefers granola. We instantiate these preferences by assigning different steering vector strengths to each person. Then we run the SteerPoser pipeline; the only difference in robot behavior is that before acting, the system asks who it is interacting with. Depending on the answer, it selectively applies the health steering vector with different magnitudes of strength. We denote semantic task success with the above ground truth user preferences. Notably, stronger steering often leads to higher compliance with user preferences but degrades the quality of the generated code or plan, resulting in lower task success. Conversely, with weaker steering, success rates improve, but user preferences are less accurately reflected. Based on these observations, we recommend using moderate steering strengths in the range $\alpha \in [-2, 2]$, which balance preference alignment and execution reliability. Additionally, we observe that overly negative steering tends to cause more performance degradation than equally strong positive steering.

We hypothesize that negative steering causes more degradation than positive steering due to the asymmetry of the activation space around meaningful latent directions. Positive steering amplifies a concept that already exists in the model's learned representation (e.g., “healthy” behavior), reinforcing internal patterns. In contrast, negative steering suppresses these activations, which may not cleanly map to coherent or semantically meaningful alternatives. Rather than steering toward a clearly defined “opposite” concept (e.g., “unhealthy”), negative activation shifts may land the model in regions of the embedding space that are less structured, undertrained, or semantically unstable, leading to degraded outputs or nonsensical completions. This suggests that the latent space is not linear or symmetric around these concept axes, and that steering too far in the negative direction may push the model into underexplored or noisy subspaces.

C. Learning Strength from History of Interactions

In previous experiments, we demonstrated the effectiveness of steering towards concepts such as health. As described in Methods, we then defined three datasets of pantomined interactions between our robotic system and our three household occupants: user 1, user 2, and user 3. The prompts in the ratings datasets differ from the prompts in the dataset used to construct the steering vector. Queries are user requests, compliant and non-compliant responses are actions the robot took versus did not take. We few shot prompted ChatGPT to generate the dataset for us. Our results recover reasonable baseline steering. The observed failure modes in the steering is discussed in Table I.

To provide more visibility into our analysis, we further extended the IBM Steering library. By default, it defines the steering vector using only the first principal component of the discrepancy between positive and negative hidden pairs. We additionally stored the second principal component; for a given concept and model layer, we thus have two projection directions $v_l^{(1)}, v_l^{(2)}$. Taking the hidden states across all user ratings, we then compute our projection similarity onto each principal direction. The scatter plots of $\{sim(h_l, v_l^{(i)}) \mid h_l \in [H_R^+]_l\}_{i=1}^2$ are presented for randomly selected layers are shown in 8a, 8b, 8c. We color code the positive and negative ratings. A couple of insights are immediately discernible.

- As one would expect, the relative clustering of positive and negative ratings depends heavily on the underlying user preference. For user 2, the user with preference for both healthy and unhealthy foods, the orange and blue clusters have significant overlap. For user 1, positively rated examples align with positive steering vector magnitude because they prefer healthy food. Thus the blue cluster is on the right. Opposite for user 3, who prefers unhealthy food.
- As one would expect, only the first principal component of the concept is meaningful. We can see that any distinction between positive and negative ratings occurs along the x-axis; the y-axis has minimal discriminating power. This implies the existence of a concept "vector" and not a concept "plane". It would be interesting to note how relative variance captured by additional dimensions relates to the underlying level of abstractness of the feature.

V. Conclusion

We view this project as laying foundational work in extending activation steering into the domain of LLM task planning for the first time. Our experiments demonstrate that steering vectors learned from interaction data can meaningfully guide language model behavior, and our approach opens the door to more interpretable and controllable planning systems. However, a key limitation lies in validation: without clear ground truth, it's unclear whether the recovered steering weights α optimally reflect user preferences. To address this, a split between training and testing feedback could allow us to assess generalization. We could fit concept magnitudes on training data, then see how highly rated models steered with that weight satisfy unseen user requests. However, for the "test set" evaluation to be meaningful, it would be far better if the underlying dataset was collected from real user feedback data rather than LLM-generated. We could potentially interface with benchmarks such as TidyBot. In terms of actual project improvement, there are many extensions possible. Our pipeline extracts layer-specific steering magnitudes and we are already able to assess how well the principal component specific to a model layer discriminates between user ratings. This could easily enable more restrictive layer steering to reduce model degradation. Despite this capability, we only tested applying uniform α across all chosen hidden layers. Other trivial extensions includes conditional steering—adjusting vector influence based on context. More compellingly, we would love to run the experiments outlined in "Methods", namely extending to multimodal steering and relational keypoint constraints (SteerKep rather than SteerPoser). Success in that endeavor would demonstrate that what is currently a promising toy setup (SteerPoser) could truly be a viable pipeline for LLM robotics value alignment. Ultimately, our vision is to enable interactive, user-aligned planning systems powered by interpretable internal model steering to enhance safety, reliability and overall user happiness - essential preconditions for robots to cohabit spaces with humans.

References

[1] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[2] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2), January 2025. ISSN 1046-8188. doi: 10.1145/3703155. URL https://doi.org/10.1145/3703155.

[3] Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao Dong. Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. 2025. URL https://api.semanticscholar.org/CorpusID:275342381.

[4] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11888–11898, 2023.

[5] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *ArXiv*, abs/2307.05973, 2023. URL https://api.semanticscholar.org/CorpusID:259837330.

[6] Zhirong Luan, Yujun Lai, Rundong Huang, Shuanghao Bai, Yuedi Zhang, Haoran Zhang, and Qian Wang. Enhancing robot task planning and execution through multi-layer large language models. *Sensors (Basel, Switzerland)*, 24, 2024. URL https://api.semanticscholar.org/CorpusID:268341391.

[7] Jiaming Liu, Chenxuan Li, Guanqun Wang, Lily Lee, Kaichen Zhou, Sixiang Chen, Chuyan Xiong, Jiaxin Ge, Renrui Zhang, and Shanghang Zhang. Self-corrected multimodal large language model for end-to-end robot manipulation. *ArXiv*, abs/2405.17418, 2024. URL https://api.semanticscholar.org/CorpusID:270063644.

[8] Junfeng Tang, Zihan Ye, Yuping Yan, Ziqi Zheng, Ting Gao, and Yaochu Jin. Zero-shot robotic manipulation with language-guided instruction and formal task planning, 2025. URL https://arxiv.org/abs/2501.15214.

[9] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David S. Udell, Juan J. Vazquez, Ulisse Mini, and Monte Stuart MacDiarmid. Steering language models with activation engineering. 2023. URL https://api.semanticscholar.org/CorpusID:261049449.

[10] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[11] Philip Quirke and Fazl Barez. Understanding addition in transformers. *arXiv preprint arXiv:2310.13121*, 2023.

[12] Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

[13] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

[14] Annah. Implementing activation steering, February 2024. URL https://www.lesswrong.com/posts/ndyngghzFY388Dnew/implementing-activation-steering. Accessed: 2025-05-12.

[15] Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear, 2025. URL https://arxiv.org/abs/2405.14860.

[16] Daniel Tan, David Chanin, Aengus Lynch, Dimitrios Kanoulas, Brooks Paige, Adria Garriga-Alonso, and Robert Kirk. Analyzing the generalization and reliability of steering vectors, 2025. URL https://arxiv.org/abs/2407.12404.

[17] RunPod. Connect to a pod. https://docs.runpod.io/pods/connect-to-a-pod, 2025. Accessed: 2025-05-12.

[18] Matthew Gunton. Using vector steering to improve model guidance, October 2024. URL https://towardsdatascience.com/using-vector-steering-to-improve-model-guidance-9cca64635510. Accessed: 2025-05-12.

[19] Bruce W Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. *arXiv preprint arXiv:2409.05907*, 2024.