# LANTRA: Language-guided Quadruped Trajectory Generation and Control in Indoor Environments

Nader Zantout
Robotics Institute - Carnegie Mellon University
nzantout@andrew.cmu.edu

***Abstract*—Recent advancements in vision-language foundation models have unlocked new capabilities in grounded language understanding for embodied agents. However, most existing systems remain limited to object-referential or action-centric commands, overlooking user instructions that implicitly or explicitly refer to navigation trajectories. In this work, we present LANTRA, an end-to-end system that translates trajectory-referential indoor navigation language queries into full control trajectories for quadrupedal robots. LANTRA interprets natural language commands into high-level discrete trajectories, refines them through a bilevel optimization process to produce kinematically feasible and collision-avoidant paths, and executes them using an iterative Linear Quadratic Regulator (iLQR) controller. To our knowledge, LANTRA is the first system to directly address trajectory-aware language grounding in vision-language-action pipelines for indoor quadrupedal navigation. Our results demonstrate that LANTRA can accurately follow complex user instructions that specify both goals and navigation constraints, marking a step forward toward more general-purpose, language-driven embodied intelligence.**

***Index Terms*—Computer Vision, Optimal Control, Large Language Models**

## I. Introduction

With the advent of vision and language foundation models, we have seen rapid progression towards general intelligence in embodied agents which are able to understand language, think, plan, and reason about their environments. Large language models (LLMs) and vision-language models (VLMs), in particular, have seen significant use in general multi-task language-conditioned policies both when fine-tuned to produce discrete actions [1] and with few-shot prompting [2]–[4].

Earlier releases of general language-conditioned policies were constrained to simple referential language that referred to unambiguous objects in the scene, like *"pick up the coke can"* or *"pick the bag at the edge of the table"* [1]. These statements form a small subdomain of the types of queries a user can issue to an assistive robot. As capturing the entire domain of embodied language is still a very open problem, research since then has focused on expanding this subdomain to include more types of queries by improving the perception and action submodules of these systems. This includes reasoning about complex referential statements in rooms with many same-class distractor objects like *"go to the chair in the middle of the two other chairs next to the window-side desk"* [5]–[7], and statements that refer directly to robot actions such as *"make the robot dog stand on two feet"* and *"shake the mustard bottle"* [8]–[10].

In this work, we focus on navigation-related actions in indoor environments in particular, which both refer to objects in the scene and describing variations in trajectory. This includes statements like ***"Fetch the tv remote, but don't get too close to the couch on the way."*** This requires a system that is able to:

1) Perceive and reason about objects in the scene, and navigate between these objects.
2) Reason about the trajectory taken to navigate between the objects.
3) Generate control actions to track this trajectory.

We therefore propose LANTRA, an end-to-end system that translates trajectory-referential navigational instructions in indoor scenes into trajectory-tracking control signals for a quadruped. As far as we know, this is the first work in the vision-language-action space that explicitly tackles trajectory-referential language in an indoor navigation scenario. Our contributions include:

1) A high-level language-conditioned planning system that translates user queries into discrete trajectories.
2) A bilevel trajectory optimization and control system that transforms a rough discretized trajectory into a kinematically feasible collision-constrained reference trajectory for a quadruped, then tracks the trajectory using an iterative linear-quadratic regulator (iLQR).

## II. Approach

We note that, due to the discrete nature of language token space, LLMs excel at human common sense and linguistically-based reasoning, but are fundamentally limited in spatial and pure numerical reasoning. This, in addition to the distribution of their training corpora which lean heavily towards articles and code, means LLMs are fundamentally ill-suited to directly translating language into high-dimensional action trajectories. The most promising approach for translating trajectory-referential language to control commands is to therefore constrain the model to generating low-resolution discrete trajectories or parametrized cost functions [9], [10]. We go with the former approach, and relegate the latter approach to future work. Our pipeline, depicted in figure 1, therefore consists of:

1) A perception module that extracts the bounding boxes of relevant objects in an indoor scene and discretizes their coordinates to make mathematical reasoning easier for LLMs.

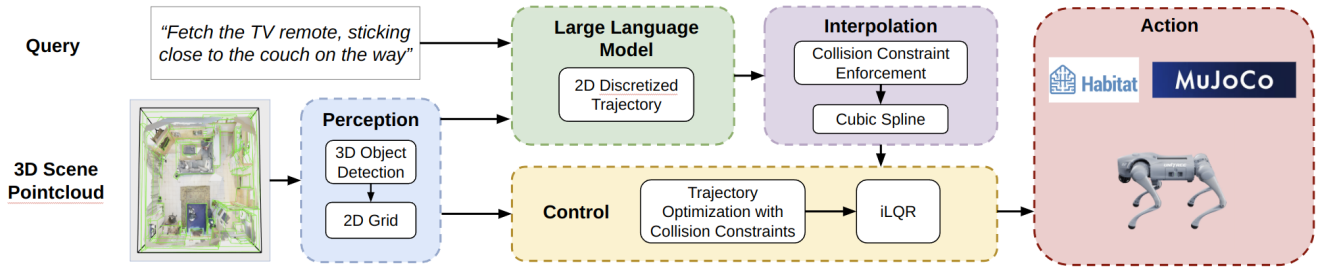



Fig. 1: The full diagram of our system.

2) An LLM-based reference trajectory planner that uses the object coordinates to translates a user language query describing a trajectory into a rough discrete trajectory.
3) An initial interpolation submodule that enforces collision constraints on the LLM generated trajectory points, then interpolates the trajectory.
4) A direct collocation module that optimizes the interpolated trajectory enforcing kinematic unicycle model constraints and collision constraints to generate a feasible trajectory for the quadruped.
5) An iLQR controller that outputs joint control efforts based on the reference body trajectory.

We delineate the details of each step in the pipeline in the following sections.

### A. Perception

We first use off the shelf indoor 3D instance segmentation models [11] or semantic mapping systems [5] to obtain a list of 3D axis-aligned bounding boxes of objects in the room. We then discretize the maximum and minimum x and y coordinates of each box into a grid with a cell size of $0.2m \times 0.2m$, obtaining a list of objects where each entry contains a unique identifying number of the object, its name, and its bounding coordinates: `[id, name, x_min, y_min, x_max, y_max]`. We note that we only deal with planar trajectories, and leave vertical maneuvers like climbing to future work. We similarly the quadruped's initial 2D position, and discretize its heading angle into 8 octants between 0 and $-\pi/4$ to input into the LLM.

### B. LLM Trajectory Generation

Given the list of relevant objects, a user query denoting a control trajectory, and the robot's current discretized planar pose, we prompt an LLM with chain of thought prompting to generate a discrete trajectory with few-shot examples. The LLM outputs a trajectory of points $\mathcal{T}_{LLM}$ in the $XY$ plane. Each point in the output trajectory is then processed to ensure it does not fall within the axis-aligned 2D collision box $B \in \mathcal{B}$ of any object in the scene. The procedure is described in algorithm 1. We maintain a conservative clearance of 0.3m from any bounding box to prevent collision.

**Algorithm 1** Collision Polygon Merging and Trajectory Projection

**Input:** Set of axis-aligned bounding boxes $\mathcal{B}$, LLM-generated trajectory points $\mathcal{T}_{LLM}$
**Output:** Collision-constrained trajectory $\mathcal{T}'_{LLM}$

**for** each bounding box $B \in \mathcal{B}$ **do**
  Inflate $B$ by adding $0.3\,\text{m}$ to each side
**end for**
Initialize graph $G = (V, E)$ with $V \leftarrow \mathcal{B}$
**for** each pair $(B_i, B_j) \in \mathcal{B}$, $i \neq j$ **do**
  **if** $B_i$ intersects $B_j$ **then**
    Add edge $(B_i, B_j)$ to $E$
  **end if**
**end for**
Find connected components $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ of $G$ using depth-first search
**for** each component $C_i \in \mathcal{C}$ **do**
  Merge all boxes in $C_i$ into a single polygon $P_i$
**end for**
Initialize modified trajectory $\mathcal{T}'_{LLM} \leftarrow \mathcal{T}_{LLM}$
**for** each point $p \in \mathcal{T}'$ **do**
  **for** each polygon $P_i$ **do**
    **if** $p$ lies within $P_i$ **then**
      Project $p$ onto the closest edge of $P_i$
    **end if**
  **end for**
**end for**
**return** $\mathcal{T}'$

After enforcing collision constraints, we interpolate $\mathcal{T}'_{LLM}$ using a cubic spline into

$$N = \frac{T}{dt} = \frac{s(\mathcal{T}'_{LLM})}{v_{max} dt}$$

timesteps, where $s(\mathcal{T}'_{LLM})$ is the total length of the LLM generated trajectory, $v_{max}$ is a maximum velocity parameter which we use to constrain the full optimized trajectory in the following section, and $dt = 0.1s$.

### C. Trajectory Optimization with Collision Constraints

To simplify the problem, we decompose the full trajectory tracking with collision avoidance into two control steps:

1) Using direct collocation on a simplified unicycle kinematics model with collision constraints to obtain a kinematically feasible planar trajectory.
2) Tracking the planar trajectory on the quadruped using iLQR.

For the first direct collocation step, we use the following unicycle kinematics model:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{a} \\ \dot{\omega} \end{bmatrix} = f(\mathbf{x}) + \mathbf{u} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \\ a \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ j \\ \alpha \end{bmatrix}$$

where $\theta$ and $\omega$ are the quadruped's heading angle with respect to $+x$ and angular velocity, and $v$ and $a$ are the longitudinal velocity and acceleration in the quadruped's body frame. $j$ and $\alpha$ are the longitudinal jerk and angular acceleration respectively, and we consider them control inputs. We note that adding a jerk term serves to make changes in acceleration much smoother, preventing sudden decelerations which may cause the quadruped to go off balance and fall. We minimize the quadratic cost:

$$\begin{aligned} J = \sum_{t=0}^{T-1} [&(\mathbf{x}_i - \mathbf{x}_{ref,i})^T Q (\mathbf{x}_i - \mathbf{x}_{ref,i}) \\ &+ (\mathbf{u}_i - \mathbf{u}_{ref,i})^T R (\mathbf{u}_i - \mathbf{u}_{ref,i})] \\ &+ (\mathbf{x}_T - \mathbf{x}_{ref,T})^T Q_f (\mathbf{x}_T - \mathbf{x}_{ref,T}) \end{aligned}$$

subject to

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}_{0,ref} \\ \mathbf{x}_T &= \mathbf{x}_{T,ref} \\ hs(\mathbf{x}_{k+1}, f(\mathbf{x}_k), \mathbf{u}_k) &= 0, \ \forall\ 0 \le k \le T-1 \\ \mathbf{x}[1:2] &\notin B', \ \forall\ B' \in \mathcal{B}' \end{aligned}$$

Where $hs(\cdot)$ is the hermite-simpson implicit integrator and $\mathcal{B}'$ is the set of planar axis-aligned bounding boxes of objects in the scene, inflated by adding a 0.3m collision buffer to each side.

We additionally impose the state and control bounds in table I. We set

$$\begin{aligned} Q = Q_f &= \mathbf{I}_{6\times 6} \\ R &= 0.1\mathbf{I}_{2\times 2} \end{aligned}$$

and solve the optimization problem using an interior-point optimizer (IPOPT).

TABLE I: State and control bounds for unicycle model trajectory optimization

| State or Control | Lower Bound | Upper Bound |
|---|---|---|
| $v\ (m/s)$ | $-0.25$ | $0.25$ |
| $a\ (m/s^2)$ | $-0.25$ | $0.25$ |
| $\omega\ (rad/s)$ | $-0.25$ | $0.25$ |
| $j\ m/s^3$ | $-0.25$ | $0.25$ |
| $\alpha\ (rad/s^2)$ | $-0.25$ | $0.25$ |

Finally, we interpolate the obtained optimized trajectory $\mathcal{T}_{opt}$ from a timestep of $0.1s$ to a timestep of $0.002s$ to obtain a final reference trajectory $\mathcal{T}_{ref}$ matching the MuJoCo simulation timestep [12]. We find that this approach gives almost the same result as directly using a smaller timestep in the initial interpolation step before optimization (figure 4), but using a larger timestep in the initial interpolation step speeds up optimization tenfold on an Intel Core i7-9750H.

### D. Quadruped Control

We follow [13] and use MuJoCo MPC [14] to track the reference trajectory using an iLQR controller on a Unitree Go2 quadruped in the MuJoCo simulator. This controller uses full body dynamics of the quadruped, with a 31-dimensional state space consisting of the main body's spatial position and orientation quaternion, along with the angles and velocities of the 12 joints.

We set the solver's friction coefficient `impratio` to 100 to generate smoother control trajectories [13]. We run iLQR at 50Hz (skipping every 10 solver timesteps), with a horizon of 0.35s. As we constrain the maximum speed to be $0.25m/s$, we enforce the gait generator to follow a trotting gait. We keep the default cost function weights.

## III. Experimental Setup

We simulate a simplified version of the simulated indoor scene in figure 2 by keeping only a single couch and a TV stand, adding their bounding boxes as solid collisions in MuJoCo (figure 3).

Fig. 2: The indoor scene, showing the sofa and the TV stand.

We test the system on four statements given to the LLM:

1) Go forward and go around the couch to the TV, but don't stick too close to the couch.
2) Go forward and go around the couch to the TV, hugging the couch on the way.
3) Go around the right side of the sofa to the TV.
4) Go around the right side of the sofa to the TV, but stay at least 1m away from the couch.

These statements test the LLM's ability to reason about egocentric view-dependent queries and proximity from objects in the scene. We use Mistral Large 2 [15] as the LLM. The robot starts at $(x, y, \theta) = (0, 0, \pi/2)$ for statements 1 and 2,

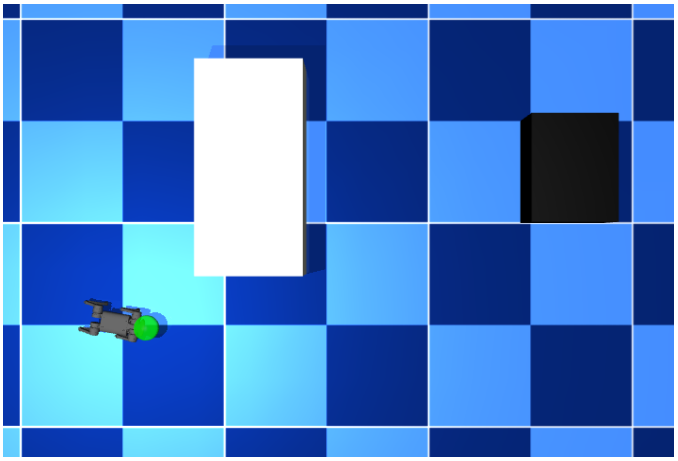


Fig. 3: The simulated version of the indoor scene in MuJoCo. The white/beige box represents the sofa, while the black box represents the TV stand.

and at $(x, y, \theta) = (0, 0, 0)$ for statements 3 and 4. We record the total costs for all four executed trajectories, along with the floating body's position in the $XY$ plane. We discuss trajectory generation, optimization, and tracking results in the following section.

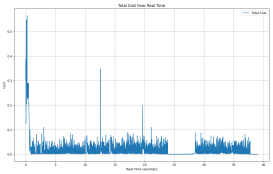
## IV. Results and Discussion

### A. Trajectory Generation and Optimization

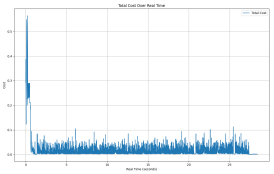
We find that the LLM is able to generate reasonable discrete trajectories for all four queries, as shown in figure 4. We note that the LLM generated trajectory for statement 3 in figure 4c is relatively unintuitive. We attribute that to the ambiguity in resolving the relative directional statement; and the right side of the couch is ambiguous depending on the viewpoint used. LLMs still struggle with reasoning about these ambiguities [5], so we deem this trajectory acceptable.

We also note that the trajectory optimizer smooths the sharp turns due to the constraints on angular velocity and acceleration, and sometimes skips large chunks of the trajectory (figures 4c and 4d). This still produces valid answers to the language queries in this case, but we note that, in figure 4d, this ends up slightly violating the statement "stay at least 1m away from the couch" in the optimized trajectory. There are two ways to resolve this: one of which is by increasing the maximum velocity parameter in the trajectory optimizer relative to the number of timesteps calculated in section II-B, and increasing the cost terms for $x$ and $y$ to ensure closer tracking. We note that this is also caused by the robot facing $+x$ in figure 4d and the LLM generating a trajectory that immediately goes towards $-y$. This can be remedied by prompting the LLM with few-shot examples to generate a buffer larger from the sofa, or by allowing the LLM to decide the collision buffer for each object and enforce a buffer of $1m$ on the sofa, which we relegate to future work.

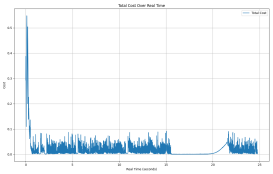
### B. Quadruped Trajectory Tracking



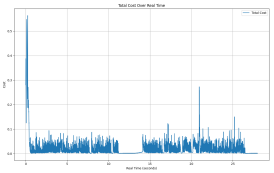
(a) Go forward and go around the couch to the TV, but don't stick too close to the couch.



(b) Go forward and go around the couch to the TV, hugging the couch on the way.



(c) Go around the right side of the sofa to the TV.



(d) Go around the right side of the sofa to the TV, but stay at least 1m away from the couch.

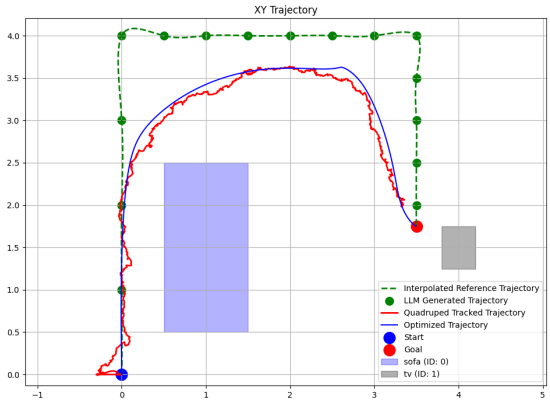
Fig. 5: Cost functions for the four tracked trajectories.

We note that the iLQR controller and gait planning system are able to successfully track all four trajectories with minimal deviation (figures 4 and 5). We note the drops to near 0 in the cost functions in figures 5b, 5c, and 5d: this is due to the quadruped decelerating and completely stopping at these points (spikes in the cost function are caused by bobbing up and down and leg movement). Optimizing the collision-free trajectory based on our kinematic model tends to cause artifacts of the trajectory decelerating and completely stopping at certain points for a few seconds. At low speeds, we note that this is safe, but at higher speeds, we note that the robot tends to fall forward if we do not further limit deceleration. We note that the best way to potentially fix this is to vary the deceleration constraint throughout the trajectory along with the jerk decrease constraint.

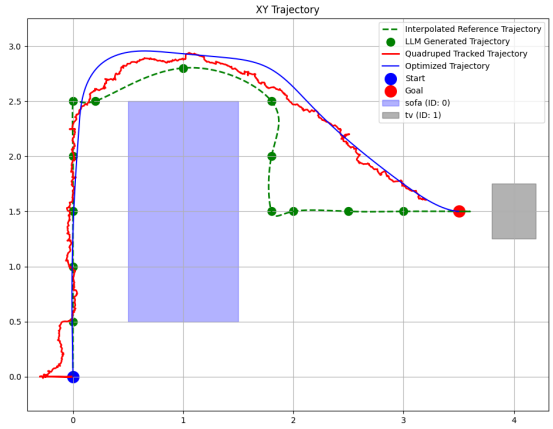
## V. Conclusion and Future Work

We present LANTRA, an end-to-end system that is able to interpret language statements that refer to trajectories and actions into discrete trajectories, then optimize kinematically feasible collision-avoidant trajectories for a quadruped which are finally tracked by an iLQR controller. We believe that this approach is a significant leap forward towards embodied AI systems that understand language at various levels, down to control actions.

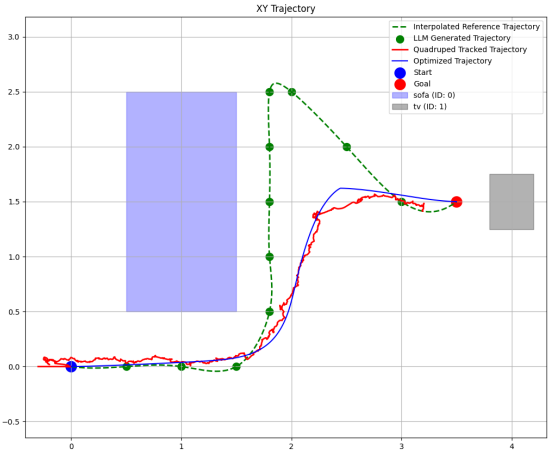
We note that our experiments were done on a single scene, and testing on larger scenes with more objects requires explicitly splitting LLM trajectory generation into multiple sub-trajectories to prevent spatial reasoning mistakes, which we relegate to future work. Additionally, our approach is only suitable for planar motions, and we do not yet handle statements referring to speed and gait, which is an important



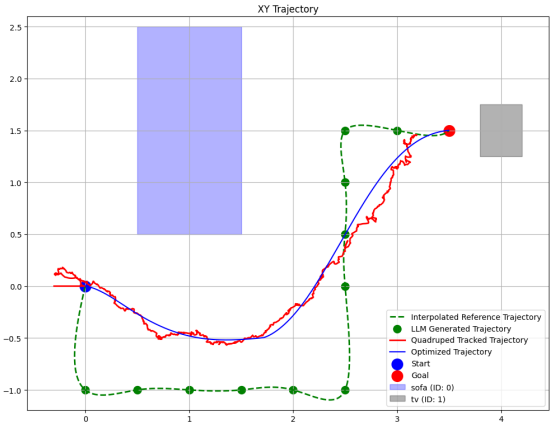

(a) Go forward and go around the couch to the TV, but don't stick too close to the couch.




(b) Go forward and go around the couch to the TV, hugging the couch on the way.




(c) Go around the right side of the sofa to the TV.




(d) Go around the right side of the sofa to the TV, but stay at least 1m away from the couch.

Fig. 4: LLM-generated (green dots), interpolated (green line), optimized (blue line), and tracked (red line) trajectories for the four spatial language statements.

future direction. Other future directions include prompting the LLM to generate code instead of trajectories which may further improve performance, and prompting the LLM to output modified costs or constraints (such as an increased collision buffer distance from objects the robot is prompted to stay away of).

## References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *arXiv preprint arXiv:2307.15818*, 2023.

[2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv preprint arXiv:2204.01691*, 2022.

[3] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 523–11 530.

[4] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence,

and A. Zeng, "Code as policies: Language model programs for embodied control," in *arXiv preprint arXiv:2209.07753*, 2022.

[5] N. Zantout, H. Zhang, P. Kachana, J. Qiu, J. Zhang, and W. Wang, "Sort3d: Spatial object-centric reasoning toolbox for zero-shot 3d grounding using large language models," 2025. [Online]. Available: https://arxiv.org/abs/2504.18684

[6] H. Zhang, N. Zantout, P. Kachana, J. Zhang, and W. Wang, "Iref-vla: A benchmark for interactive referential grounding with imperfect language in 3d scenes," 2025. [Online]. Available: https://arxiv.org/abs/2503.17406

[7] B. Jia, Y. Chen, H. Yu, Y. Wang, X. Niu, T. Liu, Q. Li, and S. Huang, "Sceneverse: Scaling 3d vision-language learning for grounded scene understanding," 2024. [Online]. Available: https://arxiv.org/abs/2401.09340

[8] T. Kwon, N. D. Palo, and E. Johns, "Language models as zero-shot trajectory generators," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, p. 6728–6735, Jul. 2024. [Online]. Available: http://dx.doi.org/10.1109/LRA.2024.3410155

[9] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. Gonzalez Arenas, H.-T. Lewis Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia, "Language to rewards for robotic skill synthesis," *Arxiv preprint arXiv:2306.08647*, 2023.

[10] S. Ismail, A. Arbues, R. Cotterell, R. Zurbrügg, and C. A. Alonso, "Narrate: Versatile language architecture for optimal control in robotics," 2024. [Online]. Available: https://arxiv.org/abs/2403.10762

[11] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe, "Mask3D: Mask Transformer for 3D Semantic Instance Segmentation," 2023.

[12] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[13] J. Z. Zhang, T. A. Howell, Z. Yi, C. Pan, G. Shi, G. Qu, T. Erez, Y. Tassa, and Z. Manchester, "Whole-body model-predictive control of legged robots with mujoco," 2025. [Online]. Available: https://arxiv.org/abs/2503.04613

[14] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," 2022. [Online]. Available: https://arxiv.org/abs/2212.00541

[15] M. A. Team, "Mistral large 2: The new generation of flag- ship model," https://mistral.ai/news/mistral-large-2407/, 2024, [Accessed 01-03-2025].