PEACE: A Planner–Executor Agent with Constraint Enforcement for UAVs

Erdem Uysal[1] and Timo Kehrer[1] and Sebastiano Panichella[1,2]

***Abstract*— Foundation models are increasingly used to drive autonomous systems, yet existing approaches either keep the model in a tight control loop, raising latency and hallucination risk, or compile natural language into opaque end-to-end policies that are hard to explain, constraint and require domain-specific datasets and fine-tuning. We propose a planner-executor agent for PX4-based drones that decouples high-level mission planning from low-level control. A large language model performs single-pass task planning, while execution is handled through a structured ROS 2 tool-calling interface bridged to MAVLink. The system constructs a world model by combining modular 2D detectors (e.g., YOLO or vision–language models) with a pinhole depth projection module for 3D object localization. A constraint enforcement layer enforces altitude limits and horizontal geofencing, and bounded replanning enables recovery from execution-time action failures. We position our approach within three common design patterns for foundation-model-based robotics systems and demonstrate its feasibility in PX4 software-in-the-loop simulations in Gazebo. Results highlight improved explainability, constraint enforcement, and reduced LLM calls compared to tightly coupled LLM control. The code, dataset, videos, and other material can be found at the following link: https://github.com/erdemuysalx/PEACE**

I. Introduction

Recent surveys argue that integrating foundation models with UAVs shifts them from remotely-piloted tools towards proactive agents capable of natural-language interaction and contextual reasoning [1], [2]. Translating this promise into deployable systems, however, exposes three recurring bottlenecks: (i) keeping the foundation model in a tight perception–reasoning–action loop, as in ReAct-style [3] agents that interleave a fresh LLM call with every tool invocation, inflates end-to-end latency and exposes the vehicle to hallucinated or ill-typed commands that can cause direct physical harm, motivating recent physical-safety benchmarks for LLM-driven UAVs [4], [1]; (ii) compiling natural language into end-to-end visuomotor policies yields controllers that are hard to explain, hard to constrain, and difficult to audit against regulatory or safety requirements [4], [2]; and (iii) those same end-to-end policies typically depend on domain-specific datasets and fine-tuning, which limits portability across platforms and tasks, and amplifies the cost of development and any sim-to-real failure [1], [2].

Three families dominate the current foundation-model robotics literature, each addressing some of these bottlenecks while inheriting others. Vision–language–navigation (VLN) systems learn or compose policies that follow free-form navigation instructions in previously unseen scenes [5], [6], [7], [8]. End-to-end vision–language–action (VLA) models such as RT-1 [9], RT-2 [10], PaLM-E [11], CognitiveDrone [12] and RaceVLA [13] learn a direct mapping from pixels and text to control, achieving closed-loop reactivity at the cost of opacity and a heavy dependence on domain-specific training data. A third family uses foundation models as high-level planners or code generators that emit scripts, structured plans, or action primitives for a classical controller to execute, ranging from DSL-generation systems such as TypeFly [14] and FlockGPT [15] to VLM-based mission planners [16], [17]; the resulting plans are human-readable and free of domain-specific training data, but their safety must be re-established for every emitted artifact.

Our work belongs to this third family. Within it, the dominant agent paradigm is ReAct-style reasoning that interleaves a fresh LLM call with every tool invocation, paying a model-inference cost at each iteration and keeping the model in the fast control path. We instead adopt the planner–executor paradigm: a single LLM pass produces a complete, typed mission plan that the executor dispatches deterministically through ROS 2, with bounded replanning only on tool failure. This deliberate bias toward determinism and auditability amortizes reasoning over the whole mission, exposes the action space as a fixed, inspectable schema, and — combined with an independent safety validation layer — targets the latency, opacity, auditability, and data-dependence bottlenecks above without any task-specific training. Our contributions are:

- *Planner–Executor agent* in which an LLM generates a complete mission plan in a single pass as a sequence of structured tool calls, executed without further model intervention.
- *World-grounded context building* which integrates pluggable 2D detectors and a depth projection module to support object-referential natural language commands grounded in a live 3D world model.
- *Safety constraint enforcement layer and recovery mechanism* that enforces safety constraints such as altitude limit or geofence, combined with bounded replanning for execution-time action failures.

II. Related Works

We review previous work in foundation models-based robot control patterns across three categories: (i) vision–language–navigation models that enable language-conditioned navigation in unknown environments; (ii)

[1]Institute of Computer Science, University of Bern, 3012 Bern, Switzerland {ramazan.uysal, timo.kehrer, sebastiano.panichella}@unibe.ch

[2]AI4I - The Italian Institute of Artificial Intelligence, 10129 Torino, Italy sebastiano.panichella@ai4i.it

vision–language–action models that enable robots to complete tasks given natural language instructions by the user; (iii) foundation models as high-level planners and code generators.

A. Vision–Language–Navigation Models

A first family focuses specifically on language-conditioned navigation, where the robot must follow free-form instructions to reach goals or find referents in previously unseen scenes. These systems typically couple perception and action inside a single learned policy, which yields strong scene-conditioned reactivity but requires task-specific training data, ties the agent to a fixed embodiment, and offers limited room for inserting external safety or interpretability constraints. OpenUAV provides a platform and benchmark with a hierarchical multimodal LLM navigator for continuous 6-DOF aerial navigation [5]; NavAgent fuses multi-scale urban street-view information to enable embodied navigation in complex environments; [18] SINGER trains a lightweight onboard visuomotor policy for language-guided UAV navigation using Gaussian-splatting simulators to reduce the sim-to-real gap [7]. NaVILA couples a VLA model that emits mid-level language waypoints with low-level visual locomotion policies for legged robots [6], while CLOI-NAV targets open-world VLN with LLM-based instruction comprehension over informative scene snapshots [8]. Complementary to these, ReMEmbR maintains a retrieval-augmented spatio-temporal memory that an LLM navigator can query over long horizons [19].

B. Vision–Language–Action Models

A second family of works train an end-to-end pipeline that learns a direct mapping from pixels and language to action. The appeal is closed-loop reactivity at control-rate frequencies, but the resulting policies are opaque, depend on large robot-specific datasets and fine-tuning, and are difficult to audit against explicit safety or regulatory constraints because the action space is implicit in the network's weights rather than exposed as a discrete schema. RT-1 [9] and RT-2 [10] scale robotic transformers by co-fine-tuning vision–language models on robot and internet data; PaLM-E injects continuous sensor modalities into a language model for grounded planning [11]. For UAVs specifically, CognitiveDrone introduces a 7B-parameter VLA for real-time cognitive tasks [12], and RaceVLA maps FPV video and language to 4D velocity control for racing UAVs [13]. Related lines include visual foundation models for embodied intelligence such as VC-1 [20], multi-agent coordination via retrieved skills [21], and autonomous-driving VLAs such as EMMA [22] and Alpamayo-R1 [23].

C. Large Models as High-Level Planners and Code Generators for UAVs

A third family of works uses LLMs as high-level planners that emit scripts, structured plans, or frontier assignments, which a classical controller then executes. Compared with the VLN and VLA approaches, this family recovers interpretability and removes the dependence on robot-specific training data, but most existing instances either emit free-form code or a custom DSL that must be re-validated for every mission, or follow a ReAct-style loop that re-invokes the LLM at every step, and reintroduces inference latency on the control path. TypeFly generates flight plans in a domain-specific language (MiniSpec) to reduce LLM inference latency [14]. FlockGPT translates natural-language instructions into signed distance function targets that guide UAV swarms [15]. Co-NavGPT uses a vision–language model as a global planner with frontier-based visual prompting for multi-robot target search [16]. LLM-based mission planners have also been proposed for extreme-environment exploration [17], and surveys and demonstrators in swarm robotics cover language-conditioned code generation and pattern formation via multi-agent reinforcement learning [24], [25]. In our work, we forgo the closed-loop reactivity and fine-grained scene conditioning of a learned VLN/VLA policy in exchange for explainability, auditability, and low control-path latency: an off-the-shelf LLM is exposed as a single-pass planner that emits a typed `MissionPlan` rather than free-form code or a custom DSL, the executor binds each tool directly to a MAVLink via ROS 2, and an independent safety validation service validates every motion command before actuation. The action space is therefore fixed, inspectable, and decoupled from the model's weights, which we view as the right point on the expressiveness–verifiability frontier for safety-critical UAV deployment.

III. Approach

We develop an agent to generate mission plans for an open world UAV flight via natural language. World model representations are generated using a perception backend of choice: a pretrained object detection model or a vision–language–model. Robot states are aggregated from real-time sensor readings. We fuse the world model and the robot state histories into a system prompt to be used with the large–language–model to generate the mission plan.

A. System Overview

The agent comprises four conceptual components: a *planner–executor* that turns natural-language queries into mission plans and dispatches them, a *perception* module that produces detections and grounds them in 3D, a *tool set* that binds a fixed action space, perception, and life-cycle operations to the flight controller, and a set of shared services that aggregate the world model and robot state and validate every motion command against safety constraints. Figure 1 sketches the data flow.

B. Planner–Executor Architecture

A single LLM call converts the user query, robot state, and world model into a structured mission plan: a reasoning string followed by an ordered list of tool calls. The executor dispatches the tool calls sequentially and re-invokes the LLM only on tool failure, in which case a bounded replan is issued.

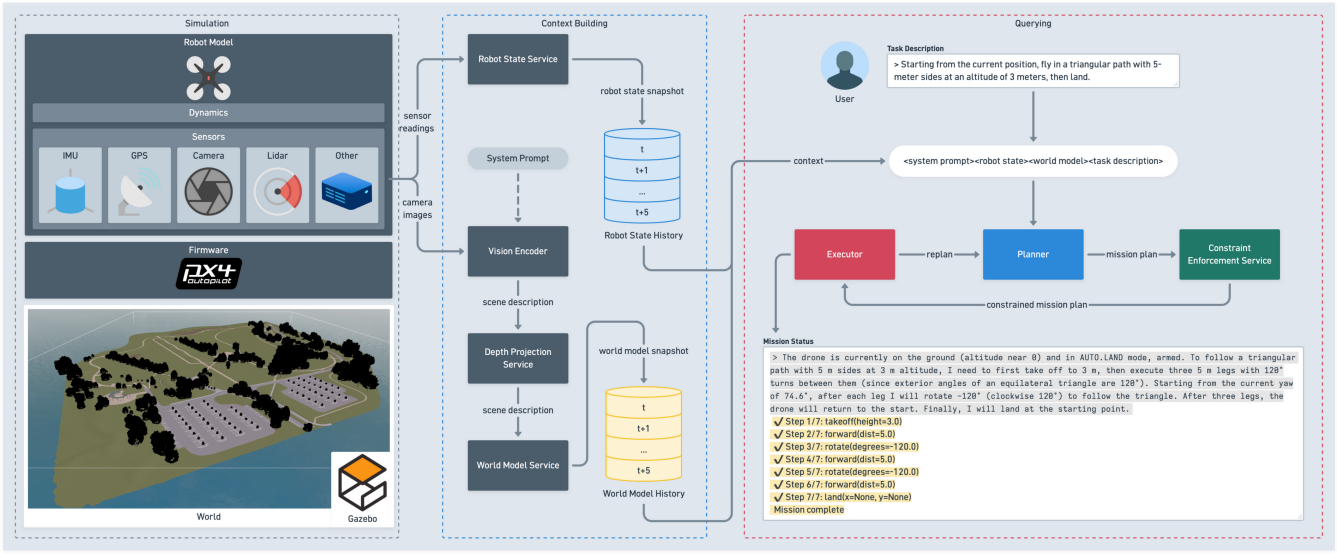

Fig. 1. System architecture overview. The system comprises three modules. (Left) The simulation environment runs a drone model with IMU, GPS, depth camera, and LiDAR sensors in Gazebo, with PX4 Autopilot handling low-level flight control. (Center) The context-building pipeline aggregates sensor readings into a Robot State Service and passes camera images through a Vision Encoder and Depth Projection Service to maintain a timestamped Robot State History and World Model History. (Right) At query time, the system prompt, robot state, world model, and user task description are composed into a single context for the Planner, which generates a mission plan validated by the Constraint Enforcement Service before being dispatched to the Executor. Execution failures trigger bounded replanning via a feedback loop to the Planner.

Removing the LLM from the fast control path eliminates an entire class of latency-induced and hallucination-induced failure modes reported in reactive loop-in-the-LLM systems [4], [1].

C. Perception Backends and 3D Grounding

Perception is abstracted behind a detector interface with two interchangeable backends: a fast 2D object detector and a vision–language model generating structured detections. A pinhole-camera depth projector lifts detections to ENU-frame 3D positions using the synchronized depth image and camera intrinsics, and exposes an explicit depth-validity flag so the planner can distinguish confident localizations from uncertain ones. The abstraction keeps perception pluggable, in the spirit of recent pretrained-representation [20] and VLN [7], [8] work, without committing to any specific VLA backbone.

D. World Model and Robot-State Aggregation

A world-model service aggregates the detection stream into a bounded history of timestamped scene snapshots, and a parallel robot-state service aggregates odometry, flight mode, and arming state. Each planner invocation reads a consistent snapshot from both services, which is the property required to ground the system prompt against the world as it is at planning time [19].

E. Prompt Construction and Context Assembly

Each planner invocation is a single LLM call with two messages, decomposed by their lifetime. The *system prompt* is static and carries the invariant knowledge: the tool set, the mission rules, and the output schema. The *user prompt* is assembled at query time and carries only the volatile states: the natural-language task description, the current robot state, and the current world model, with a short history of past states for temporal reasoning. On replan, the user message is rebuilt from a fresh snapshot so that subsequent planning steps see the world as it is at the moment of replanning rather than as it was when the mission began.

F. Constraint Enforcement

A dedicated safety layer validates every motion command before it reaches the autopilot, enforcing two geometric constraints: an altitude band and a horizontal geofence around the takeoff point. Out-of-band altitude targets are clamped; geofence breaches abort the offending tool calls and surface a structured failure to the replanner. Because the check is purely geometric, it is verifiable independently of the language model, and provides the construction required by the avoid-collision and regulatory-compliance axes of recent physical-safety benchmarks for LLM-controlled UAVs [4].

IV. Experiments

We report preliminary experiments of the proposed agent. The aim is not a quantitative benchmark, but to characterize the agent's behavior over a structured set of natural-language prompts and to surface the recurring failure modes that constrain the current design.

A. Experimental Procedure

We compiled a set of 108 natural-language prompts spanning eight task categories, aligned with the capabilities exposed by the tool set: *takeoff and landing* (12), *relative motion* (12), *altitude and attitude control* (12), *waypoint navigation* (13), *geometric path following* (13), *object search*

and localization (12), *open-vocabulary navigation* (13), *composite multi-step missions* (13), and *safety violation* (8). Each prompt carries a stable identifier and is phrased as a single, free-form natural-language instruction; representative examples are "Fly to position $x = 3$, $y = 4$, $z = 3$" for waypoint navigation, rest can be obtained via [1]. Object-referential prompts are restricted to the vocabulary {*person, car, bench, traffic light, fire hydrant*}, due to the availability of assets in the simulation.

B. Preliminary Results

The mission logging makes each mission auditable offline. For the prompt "Take off to 5 meters then move forward 3 meters", for example, the planner emits a single `MissionPlan` comprising `takeoff(5)` followed by `forward(3)`; the executor dispatches the two calls in order, and the recorded reasoning-step results, and the final robot state together describe what the agent attempted, what it did, and where it ended up.

Across categories, four behavioral patterns are visible. Metric tasks (waypoint, relative motion, altitude control) admit the most reliable plans, since the parameters in the prompt map one-to-one onto a tool argument. Geometric paths are correctly decomposed into ordered waypoints when the shape and edge length are stated explicitly, and the trajectory is resolved at planning time, consistent with the open-loop design. Perception-conditioned categories add a dependence on detector recall and on the depth-validity flag exposed by the world model. In composite missions, steps depend on each other, so an undetected failure in one step can cascade into mission failure.

C. Failure Modes

The preliminary results exhibit three recurring failure modes. (i) *Planner-level errors* arise when the LLM emits a plan that violates the typed schema or a mission rule encoded in the system prompt; such plans are caught at validation and either fail terminally or trigger a bounded replan. (ii) *Perception-level uncertainty* arises when the requested object class is absent from the detection set or marked as uncertain by the depth projector; the planner avoids acting on uncertain detections, but extended runs of uncertain observations, collapse object search and navigation to degenerate plans. (iii) *Safety-validation aborts*, where an action is rejected because it would breach the altitude limit or the horizontal geofence; these are by design but propagate to the mission failure.

V. Implications and Future Directions

We have presented an agent in which mission planning is decoupled from control, and in which every action that reaches the low-level controller is structured and safety constrained by an independent service. This design point is distinct from code-generating planners [14], [15], which emit executable artifacts for every mission and thus expand the action space at run time , and from end-to-end vision–language–action policies [12], [13], [9], [10], which embed the action space in the network weights and offer no inspectable trace of what the model decided.

The separation between the language model and the control path has three practical implications. First, mission plans become auditable: the planner's reasoning, the structured mission plan, and the per-step results are recoverable from the mission log, which makes offline analysis of failure modes possible. Second, an off-the-shelf LLM is sufficient for the planner; in-context-learning composition of natural-language tasks over the tool set requires no domain-specific fine-tuning, in contrast to domain-specific models [9], [10], [12], [13]. Third, safety constraints are verified outside the planner, so violations are caught even when the planner emits a plan that is otherwise well-formed. Moreover, the adoption of a planner–executor paradigm reduces LLM invocations to a single call, which is substantially lower than ReAct-based agents that typically require an LLM invocation after each tool interaction.

Three limitations are visible in the failure modes of Section IV. Perception-level uncertainty manifests because the world model is the only feedback channel between the scene and the planner, so a sequence of degenerate snapshots can lead to a degenerate plan that no recovery mechanism below the granularity of a tool failure can correct. Execution-level state-machine interactions, particularly between off-board control and arming, indicate that idempotency at the tool layer is required: each tool should re-establish its preconditions internally rather than depend on the planner to interleave them.

Two directions follow from the design above. The first is a sim-to-real deployment on physical UAV hardware. The second concerns dynamic scenes whose state changes within a mission: the current open-loop planning assumption is brittle in such settings, since a target object that moves between planning and execution, or an obstacle that appears mid-mission, is only handled at the granularity of a tool failure. The accumulated mission traces, consisting of plans paired with perception snapshots and outcomes, are suitable for distilling a compact vision–language–action model that can complement the language-conditioned planner with a reactive, visually-grounded channel for dynamic environments.

[1] https://huggingface.co/datasets/erdemuysalx/uav-mission-prompts

Acknowledgment

We thank the EU Commission and the State Secretariat for Education, Research, and Innovation (SERI) for funding the project InnoGuard, Marie Sktodowska-Curie Actions Doctoral Networks (HORIZON-MSCA-2023-DN), the Swiss National Science Foundation (SNSF) for funding the "SwarmOps" project (No. 200021_219732).

References

[1] Y. Tian, F. Lin, Y. Li, T. Zhang, Q. Zhang, X. Fu, J. Huang, X. Dai, Y. Wang, C. Tian *et al.*, "Uavs meet llms: Overviews and perspectives towards agentic low-altitude mobility," *Information Fusion*, vol. 122, p. 103158, 2025.

[2] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "Uavs meet agentic ai: A multidomain survey of autonomous aerial intelligence and agentic uavs," *arXiv preprint arXiv:2506.08045*, 2025.

[3] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *The eleventh international conference on learning representations*, 2022.

[4] Y.-C. Tang, P.-Y. Chen, and T.-Y. Ho, "Defining and evaluating physical safety for large language models," *arXiv preprint arXiv:2411.02317*, 2024.

[5] X. Wang, D. Yang, Z. Wang, H. Kwan, J. Chen, W. Wu, H. Li, Y. Liao, and S. Liu, "Towards realistic uav vision-language navigation: Platform, benchmark, and methodology. arxiv 2024," *arXiv preprint arXiv:2410.07087*, 2024.

[6] A.-C. Cheng, Y. Ji, Z. Yang, Z. Gongye, X. Zou, J. Kautz, E. Bıyık, H. Yin, S. Liu, and X. Wang, "Navila: Legged robot vision-language-action model for navigation," *arXiv preprint arXiv:2412.04453*, 2024.

[7] M. Adang, J. Low, O. Shorinwa, and M. Schwager, "Singer: An onboard generalist vision-language navigation policy for drones," *arXiv preprint arXiv:2509.18610*, 2025.

[8] M. Lee, J. Park, J. Jeong, and Y. Cho, "Cloi-nav: Open-world vision-and-language navigation via complex, long-horizon ordered instructions," in *IROS 2025 Workshop: Open World Navigation in Human-centric Environments*, 2025.

[9] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[10] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023," *URL https://arxiv. org/abs/2307.15818*, vol. 1, p. 2, 2024.

[11] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[12] A. Lykov, V. Serpiva, M. H. Khan, O. Sautenkov, A. Myshlyaev, G. Tadevosyan, Y. Yaqoot, and D. Tsetserukou, "Cognitivedrone: A vla model and evaluation benchmark for real-time cognitive task solving and reasoning in uavs," *arXiv preprint arXiv:2503.01378*, 2025.

[13] V. Serpiva, A. Lykov, A. Myshlyaev, M. H. Khan, A. A. Abdulkarim, O. Sautenkov, and D. Tsetserukou, "Racevla: Vla-based racing drone navigation with human-like behaviour," *arXiv preprint arXiv:2503.02572*, 2025.

[14] G. Chen, X. Yu, N. Ling, and L. Zhong, "Typefly: Flying drones with large language model," *arXiv preprint arXiv:2312.14950*, 2023.

[15] A. Lykov, S. Karaf, M. Martynov, V. Serpiva, A. Fedoseev, M. Konenkov, and D. Tsetserukou, "Flockgpt: Guiding uav flocking with linguistic orchestration," in *2024 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, 2024, pp. 485–488.

[16] B. Yu, Q. Yuan, K. Li, H. Kasaei, and M. Cao, "Co-navgpt: Multi r obot cooperative visual semantic navigation using vision language models," *IEEE Robotics and Automation Letters*, vol. 11, no. 2, pp. 2122–2129, 2025.

[17] O. Sautenkov, Y. Yaqoot, M. A. Mustafa, F. Batool, J. Sam, A. Lykov, C.-Y. Wen, and D. Tsetserukou, "Uav-codeagents: Scalable uav mission planning via multi-agent react and vision-language reasoning," *arXiv preprint arXiv:2505.07236*, 2025.

[18] Y. Liu, F. Yao, Y. Yue, G. Xu, X. Sun, and K. Fu, "Navagent: Multi-scale urban street view fusion for uav embodied vision-and-language navigation," *arXiv preprint arXiv:2411.08579*, 2024.

[19] A. Anwar, J. Welsh, J. Biswas, S. Pouya, and Y. Chang, "Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 2838–2845.

[20] A. Majumdar, K. Yadav, S. Arnaud, J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, T. Wu, J. Vakil *et al.*, "Where are we in the search for an artificial visual cortex for embodied intelligence?" *Advances in Neural Information Processing Systems*, vol. 36, pp. 655–677, 2023.

[21] S. Kuroki, M. Nishimura, and T. Kozuno, "Multi-agent behavior retrieval: Retrieval-augmented policy training for cooperative push manipulation by mobile robots," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 671–12 678.

[22] J.-J. Hwang, R. Xu, H. Lin, W.-C. Hung, J. Ji, K. Choi, D. Huang, T. He, P. Covington, B. Sapp *et al.*, "Emma: End-to-end multimodal model for autonomous driving," *arXiv preprint arXiv:2410.23262*, 2024.

[23] Y. Wang, W. Luo, J. Bai, Y. Cao, T. Che, K. Chen, Y. Chen, J. Diamond, Y. Ding, W. Ding *et al.*, "Alpamayo-r1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail," *arXiv preprint arXiv:2511.00088*, 2025.

[24] V. Strobel, M. Dorigo, and M. Fritz, "Llm2swarm: robot swarms that responsively reason, plan, and collaborate through llms," *arXiv preprint arXiv:2410.11387*, 2024.

[25] H.-S. Liu, S. Kuroki, T. Kozuno, W.-F. Sun, and C.-Y. Lee, "Language-guided pattern formation for swarm robotics with multi-agent reinforcement learning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 8998–9005.