Cooperative Multi-Agent Mobile Manipulation using Learned Dynamical Systems

Saray Bakker[1], Wendelin Böhmer[2] and Javier Alonso-Mora[1]

***Abstract*—We propose a framework for safe and interpretable multi-agent collaboration in mobile manipulation leveraging independently learned single-agent dynamical systems. Each robot is equipped with a stable task-space vector field defined on $SE(3)$, which policy we preserve, while enabling collaboration through velocity scaling. Unlike prior collaborative learning approaches that rely on predicting key poses and delegating safety-critical reasoning to low-level planners, our method incorporates whole-body collision avoidance directly into the policy while remaining close to the original dynamical systems. We formulate multi-agent coordination as a finite-horizon optimization problem that adapts execution speed, without modifying motion direction, to satisfy task constraints and ensure collision-free behavior. The proposed framework is parallelized with JAX and evaluated in simulation on two mobile manipulators, demonstrating safe execution and improved coordination across both illustrative and randomized scenarios.**

***Index Terms*—Multi-agent Collaboration, Learned Dynamical Systems, Mobile Manipulation**

I. Introduction

Mobile manipulators are versatile platforms capable of assisting humans in household tasks [1], or performing labor-intensive jobs, such as operating in greenhouses [2] or construction sites [3]. In these domestic and professional environments, efficiency rises when robots collaborate with robots and humans. For instance, consider a mobile manipulator lifting an object to allow another agent to clean the surface beneath it. Such scenarios demand that each robot understands the tasks and intentions of its collaborators, adapting its behavior proactively to ensure both task effectiveness and safety. The movements of these robots must therefore be not only efficient in a multi-agent context but also intuitive and predictable for all agents involved.

Learning from Demonstration (LfD) has emerged as a powerful framework for transferring human skills to robots, with successful applications in manipulators [4], [5], dual-arm systems [6], and mobile manipulators [7], [8], [9]. However, extending these skills from single-agent to multi-agent collaborative movements presents several challenges. First, collecting multi-agent demonstrations is inherently complex, and especially for mobile manipulators even a few demonstrations can be challenging to obtain. Second, real-world execution is rarely perfect: other agents may deviate from their intended policies due to disturbances, and any learned policy is inherently only an approximation of the true objective. Nevertheless, many dual-manipulator systems rely on centralized control with perfect state knowledge of all agents [6]. Finally, many learned policies focus on providing end-effector key poses or velocities, delegating whole-body collision avoidance to low-level planners [8]. In multi-agent settings with mobile manipulators, this reliance on low-level planning for whole-body feasibility can become limiting, as control actions may drive the system into off-distribution regions of the state space, where the learned policy may produce unreliable actions, thereby reducing both interpretability and safety guarantees.

Rather than depending on extensive multi-agent demonstrations, recent approaches have explored composing multi-agent behavior from single-agent demonstrations. In [10], collision avoidance between manipulator end-effectors is addressed by combining attraction to learned trajectories with collision avoidance terms and a timing heuristic, though this does not guarantee collision avoidance for all manipulator links. Furthermore, single-agent demonstrations can be combined with multi-agent cost functions to learn coordinated dual-manipulator diffusion policies without requiring joint demonstrations [11]. Other methods rely on multi-manipulator demonstrations during training, but focus on ensuring global consistency during decentralized execution [12], [13]. Learning-free task and motion planning (TAMP) methods, such as those using temporal logic [14] or long-horizon planning frameworks [3], also offer solutions for collaborative tasks incorporating whole-body collision avoidance. However, these TAMP approaches often fall short of real-time execution required for physical platforms.

In this work, we consider safety-critical multi-agent collaboration settings in which whole-body collision avoidance and interpretability are essential. We assume that each robot is equipped with a single-agent learned dynamical system, represented as a vector field, that achieves its task while guaranteeing convergence to the final goal state. In particular, we focus on policies defined on $SE(3)$ with formal convergence properties, such as those in [5], [15], [16]. Our goal is to preserve these demonstrated single-agent behaviors as much as possible while adapting them for collaborative execution.

Rather than learning joint multi-agent policies from large demonstration datasets, we assume that high-level task assignments and temporal constraints (e.g., "Robot 1 must lift an ob-

The authors are with the (1) Department of Cognitive Robotics and (2) the Department of Electrical Engineering, Mathematics & Computer Science, Delft University of Technology, 2628 CD, Delft, The Netherlands.

This project has received funding from the European Union through ERC, INTERACT, under Grant 101041863. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Corresponding author: Saray Bakker (*s.bakker-7@tudelft.nl*).

ject before Robot 2 cleans the surface") can be extracted from a small number of multi-agent demonstrations or videos [17]. We then propose a collaborative framework that modifies only the execution velocity of the single-agent dynamical systems, enabling whole-body collision avoidance, and task-consistent behavior without altering the underlying motion structure. This preserves the interpretability and stability properties of the original policies while enabling safe collaboration.

Our main contributions are as follows:

- We propose a collaborative framework for multi-agent mobile manipulators that composes single-agent learned dynamical systems into safe and interpretable joint behavior, while preserving the convergence guarantees of the original policies.
- A predictive velocity-scaling optimization scheme enforces whole-body collision avoidance and high-level task constraints over a finite horizon. By adapting execution speed rather than motion direction, the method minimizes deviations from demonstrated behaviors.
- We implement the framework in JAX with FRAX [18], a kinematics-tailored version of MuJoCo XLA (MJX), and validate it in simulation with two mobile manipulators, demonstrating effective multi-agent collaboration through task-consistent coordination and collision-free execution.

II. Preliminaries

A. Single-agent Dynamical Systems

Dynamical systems (DSs) provide an effective framework for encoding motion behaviors either from human demonstrations [19] or through manual design by combining multiple sub-behaviors into a single DS [20]. A motion is represented as an autonomous continuous-time DS of the form,

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)), \tag{1}$$

where $\boldsymbol{x}(t)$ denotes the system state and $\boldsymbol{x}^*$ is the equilibrium point corresponding to the task goal, such that $f(\boldsymbol{x}^*) = \mathbf{0}$, with $\mathbf{0}$ denoting the zero vector of appropriate dimension.

The learned behavior is typically defined in the end-effector state space, which enables adaptation to changes in goal pose and the robot's embodiment. In this work, we represent the end-effector pose as $\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{\xi}]$, where $\boldsymbol{p} = [x, y, z]$ denotes the Cartesian position and $\boldsymbol{\xi} = [\xi_w, \xi_x, \xi_y, \xi_z]$ denotes the orientation represented as a quaternion. The output of the DS, $\dot{\boldsymbol{x}}$, therefore corresponds to the end-effector twist.

Stability of the learned DS can be enforced in several ways. In [16], stability is achieved by learning a bijective mapping from the task-space dynamics to a simple stable system in a latent space. Other approaches enforce stability directly through the training objective [5] or by guaranteeing convergence toward a geometric curve learned from demonstrations [15]. These formulations are particularly suitable for encoding robot motions on $SE(3)$, as they provide well-defined derivatives, convergence guarantees, and efficient parallel implementations. More generally, our framework is compatible with any learned vector field whose derivatives are well-defined and whose dynamics converge reliably to the task goal.

III. Collaborative multi-agent behavior generation

In this section, we formulate collaborative multi-agent behavior generation as an optimization problem built using independently learned single-agent dynamical systems. Each robot is assumed to have a stable task-space policy for its assigned task, and our objective is to preserve this behavior as much as possible while enabling safe and coordinated execution. To achieve this, we adapt only the velocity of each dynamical system over a finite horizon, allowing robots to satisfy high-level task constraints and ensure whole-body collision avoidance while maintaining interpretability and convergence guarantees.

A. Multi-agent Optimization problem

For a robotic system, the configuration of a robot is defined as $\boldsymbol{q} \in \mathcal{C}$ with time derivatives $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$. Each agent $i \in 1, \ldots, N$ is capable of executing M tasks, where each task is learned via a single-agent dynamical system,

$$\dot{\boldsymbol{x}}^i(t) = f_{\text{DS}}^m(\boldsymbol{x}^i(t)),\ \forall m \in 1, \ldots, M. \tag{2}$$

In this work, we assume that a high-level decision maker is responsible for task assignment, determining which robot is most suitable for executing each task, as well as defining the associated constraints, such as positional or timing requirements, necessary for successful task completion. For a two-robot scenario, we can therefore assume that Task 1 is executed by Robot 1 and Task 2 by Robot 2.

Our objective is to minimize deviations from each robot's single-agent trajectory while satisfying task constraints and ensuring whole-body collision avoidance. To obtain interpretable motions, we scale only the velocity output of the single-agent DS without altering its direction. Specifically, we optimize the velocity scaling factors $\boldsymbol{\alpha} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^N]^\top$ for N robots, where $\boldsymbol{\alpha}^i = [\alpha^i(0), \ldots, \alpha^i(K)]^\top$ represents the scaling factors over a horizon of length K.

To generate meaningful collaborative motions, we assume knowledge of the other robot's task and reason about the behavior of the other robot(s) $j \neq i$ in two ways: when robot j is cooperative, it adapts its velocity scaling $\boldsymbol{\alpha}^j$ using the same collaborative policy; when robot j is non-cooperative, it follows its original single-agent policy without adaptation, i.e., $\boldsymbol{\alpha}^j = \mathbf{1}$.

In the following, we describe how forward predictions are generated over the optimization horizon and how collision avoidance and task constraints are incorporated. The complete optimization problem is summarized in Eq. 9.

1) Forward prediction: As we optimize $\boldsymbol{\alpha}$ along a prediction horizon, realistic forward simulations are required for the whole-body system. Given the joint configuration of each robot at the prediction-step k, $\boldsymbol{q}^i(k)$, we leverage forward kinematics (FK) to obtain the end-effector poses,

$$\boldsymbol{x}^i(k) = FK(\boldsymbol{q}^i(k)), \quad \forall i \in [1, \ldots, N]. \tag{3}$$

The end-effector poses are fed into the single-agent dynamical systems in Eq. 2, where the desired velocity outputs are scaled by $\alpha^i(k)$,

$$\dot{\boldsymbol{x}}_{\mathrm{d}}^i(k) = \alpha^i(k) f_{\mathrm{DS}}^i(\boldsymbol{x}^i(k)), \quad i \in [1, \dots, N], \tag{4}$$

where $\dot{\boldsymbol{x}}_{\mathrm{d}}^i$ indicates the scaled desired end-effector velocity. The desired positional and rotational velocities of the end-effectors, or so-called twists, are mapped towards joint velocities by a differentiable low level controller (LLC). In this work, we leverage differential null-space inverse kinematics [21],

$$\dot{\boldsymbol{q}}_{\mathrm{d}}^i = \boldsymbol{J}^\top \left(\boldsymbol{J}\boldsymbol{J}^\top + \boldsymbol{D}\right)^{-1} \dot{\boldsymbol{x}}_{\mathrm{d}}^i + \left(\boldsymbol{I} - \boldsymbol{J}^\dagger \boldsymbol{J}\right) \boldsymbol{K}_n \left(\boldsymbol{q}_0 - \boldsymbol{q}^i\right) \tag{5}$$

with Jacobian $\boldsymbol{J}(\boldsymbol{q}^i)$, damping matrix $\boldsymbol{D}$, null-space gain $\boldsymbol{K}_n$ and homing configuration $\boldsymbol{q}_0$. The joint velocity generated by the LLC is integrated to obtain the configuration at the next predicted state with time-step Δt,

$$\boldsymbol{q}^i(k+1) = \boldsymbol{q}^i(k) + \Delta t \dot{\boldsymbol{q}}_{\mathrm{d}}^i(k), \tag{6}$$

given that $\boldsymbol{q}^i(0)$ is the current robot configuration $\boldsymbol{q}_0^i$.

2) Collision avoidance: To achieve whole-body collision avoidance, the distance between any pair of points on the robots' body must exceed a predefined threshold $d_{\min}$. For two robots, this constraint is expressed as:

$$d_{\min} - d\left(f_{\mathrm{coll}}(\boldsymbol{q}^1(k)), f_{\mathrm{coll}}(\boldsymbol{q}^2(k))\right) \leq 0, \tag{7}$$

where the function f_{coll} maps the robot's configuration $\boldsymbol{q}(k)$ to a geometric representation of the robot bodies, modeled as a collection of geometric primitives such as capsules or spheres. Using simple geometric primitives ensures that the distance function $d(\cdot)$ remains well-defined and differentiable.

3) Task constraints: To obtain coordinated behavior between multiple agents, we assume that a high-level decision maker can distill this information from a single or small set of multi-human demonstrations or multi-robot demonstrations [17]. For this work, we will only define task constraints with respect to the relative timing and pose between agents. For example, a constraint could encode that agent 1 should achieve a (sub-)goal pose, $\boldsymbol{x}_{\mathrm{g}}^1$, at least Z prediction steps before the other agent j arrives at a (sub-)goal pose, $\boldsymbol{x}_{\mathrm{g}}^2$,

$$k_{\mathrm{g}}^i = \min\left\{k \;\middle|\; \|\boldsymbol{x}^i(k) - \boldsymbol{x}_{\mathrm{g}}^i\| \leq d_{\mathrm{g}}\right\} \tag{8a}$$

$$k_{\mathrm{g}}^j = \min\left\{k \;\middle|\; \|\boldsymbol{x}^j(k) - \boldsymbol{x}_{\mathrm{g}}^j\| \leq d_{\mathrm{g}}\right\} \tag{8b}$$

$$k_{\mathrm{g}}^i + Z \leq k_{\mathrm{g}}^j \tag{8c}$$

4) Multi-agent optimization problem: Having introduced the single-agent dynamical systems, the forward prediction model, and the collision and task constraints, we now combine these components into a unified multi-agent formulation. For a system of N robots with velocity scaling variables $\boldsymbol{\alpha}$, the resulting optimization problem integrates predictive behavior over the horizon with both task achievement and safety requirements, and can be written as follows:

$$\min_{\boldsymbol{\alpha}} \quad \sum_i^N \left(w_\alpha \sum_0^K (\alpha^i(k) - 1)^2 \right) \tag{9a}$$

$$s.t. \quad \dot{\boldsymbol{x}}_{\mathrm{d}}^i = \alpha^i(k) f_{\mathrm{DS}}^i(\boldsymbol{x}^i(k)), \tag{9b}$$

$$\dot{\boldsymbol{q}}_{\mathrm{d}}^i = \mathrm{LLC}(\dot{\boldsymbol{x}}_{\mathrm{d}}^i(k)), \tag{9c}$$

$$\boldsymbol{q}^i(k+1) = \boldsymbol{q}^i(k) + \Delta t \dot{\boldsymbol{q}}_{\mathrm{d}}^i(k), \tag{9d}$$

$$\boldsymbol{x}^i(k) = \mathrm{FK}(\boldsymbol{q}^i(k)), \tag{9e}$$

$$\boldsymbol{q}^i(0) = \boldsymbol{q}_0^i, \tag{9f}$$

$$\text{Eq. } 7, \tag{9g}$$

$$\text{Eq. } 8, \tag{9h}$$

$$0 < \alpha^i(k) \leq \alpha_{\max}, \tag{9i}$$

$$|\dot{\boldsymbol{x}}_{\mathrm{d}}^i| \leq |v|_{\max}, \tag{9j}$$

We observe that the objective function penalizes deviations from $\alpha^i(k) = 1$ with weight w_α, since this setting corresponds to executing the original single-agent dynamical system (DS) without modification. Equations 9b-9f are used to forward-simulate the system from the current configuration over the prediction horizon using the velocity scaling $\boldsymbol{\alpha}$ together with the learned single-agent DSs. To prevent backward motion along the DS, all elements of $\boldsymbol{\alpha}$ are constrained to be strictly positive and bounded (Eq. 9i). In addition, the magnitude of the desired end-effector velocity $|\dot{\boldsymbol{x}}_{\mathrm{d}}^i|$ is limited by a maximum value $|v|_{\max}$.

5) Notes on stability: The single-agent dynamical system, $f_{\mathrm{DS}}(\boldsymbol{x})$, is designed to ensure convergence to the goal under the assumptions of the underlying methods [5], [15], [16]. To enable multi-agent cooperation, we introduce a velocity scaling factor $\boldsymbol{\alpha}$ that is bounded and strictly positive, which modulates the execution speed without altering the direction of the vector field generated by the DS. Under the assumption that whole-body collision avoidance and task constraints remain non-blocking, and that the Jacobians used in the LLC remain well-defined, the resulting behavior is expected to preserve the stability properties of the original single-agent DS. However, we note that minor deviations from ideal behavior may arise due to discretization effects introduced during forward simulation of the DS.

IV. Experiments

In this section, we evaluate the proposed framework in simulation on mobile manipulation tasks requiring coordinated multi-agent behavior under collision and task constraints. We first describe the simulation and control setup, then present an illustrative example using single-agent DSs in $\mathbb{R}^2$. Finally, we report additional experiments with DSs in $SE(3)$ using randomized initializations to evaluate performance under both cooperative and non-cooperative settings.

A. JAX-Based Simulation and Control Framework

Experiments are conducted in simulation using two mobile manipulators, each consisting of a Clearpath Dingo omnidirec-

tional base and a Kinova Gen3 Lite arm, resulting in a 9-DoF system per robot.

The FK and LLC are parallelized using JAX [22] and FRAX [18], a tailored version of MuJoCo XLA (MJX) [23] for forward kinematics. To obtain well-defined distances and their gradients in FRAX, we represent each robot body as a collection of spheres. While this representation slightly over-approximates the true robot geometry, it enables well-defined collision-avoidance constraints with reliable gradient computation. Note that MJX already supports capsule-based geometric approximations of robot bodies, a capability planned for future integration into FRAX.

As the single-agent DS, we use the curve-induced dynamical system proposed in [15], implemented in a JAX-parallelized form. More generally, any single-agent DS can be used in our framework as long as its gradients are available, including for example neural network-based policies implemented in PyTorch. The inequality constraints in Equation 9 are enforced via a relaxed formulation to ensure smooth and well-defined gradients during optimization.

B. Illustrative Example using $\mathbb{R}^2$ Dynamical Systems

In this example, we consider a scenario in which two mobile manipulators must coordinate their movements given their assigned tasks. Each single-agent dynamical system is learned from demonstrations in $\mathbb{R}^2$, and the corresponding vector fields are defined over the (x, y)-plane. The z-position and orientation are fixed to their final goal values, specifically $z = 0.5\,\text{m}$ and $\boldsymbol{\xi} = [0.0001,\ -0.8220,\ -0.0006,\ -0.5695]$. The task constraint requires Robot 1 to reach the goal region, defined as a 1-meter radius ($d_g = 1\,\text{m}$), $0.5\,\text{s}$ earlier than Robot 2 ($Z = 5$). The optimization is performed over a horizon of $K = 200$ prediction steps with a discretization interval of $\Delta t = 0.1\,\text{s}$. The control parameter $\boldsymbol{\alpha}(k)$ is updated every 10 prediction steps, yielding a piecewise-constant control signal over the optimization horizon and reducing abrupt changes. To solve the resulting optimization problem, we employ the Optax Adam solver from the JAX optimization library [24], using a maximum of 100 solver iterations. The optimization is initialized from three diverse initial guesses ($N_{\text{init}} = 3$), and the solution with the lowest objective value is selected.

As illustrated in Fig. 1, the cooperative multi-agent policy enables both robots to adapt their velocity scaling, resulting in a joint strategy that satisfies the task constraints. As shown in Fig. 1-1 and Fig. 1-2, this scaling affects only the execution speed along the DS trajectories, while the spatial paths remain unchanged. The resulting behavior respects the constraints on $\boldsymbol{\alpha}$ (Fig. 1-3) and the velocity magnitude (Fig. 1-4), while ensuring whole-body collision avoidance (Fig. 1-5). This is particularly important in mobile manipulation settings, where considering only end-effector collisions would be insufficient, as observed in Fig. 2a.

Interestingly, Fig. 1-3 reveals an emergent behavior: Robot 2 waits for Robot 1 to pass first to prevent a collision. This behavior can also be observed in the multi-agent simulations using the proposed strategy (Fig. 2a), while the vanilla strategy with $\boldsymbol{\alpha} = \mathbf{1}$ resulted in a collision (Fig. 2b). Notably, this waiting behavior is not enforced heuristically, allowing for alternative strategies, such as close-proximity manipulation or more sequential task execution, depending on the constraints.

We currently assume that each robot executes the planned trajectory perfectly, and that in a decentralized setting both robots arrive at the same solution. Future work could build on this by relaxing these assumptions and incorporating more realistic execution conditions, while further improving the robustness of the framework for decentralized deployment on real-world systems.

C. Experimental results

We now consider a scenario involving two mobile manipulators, where the dynamical systems for each task are defined in $SE(3)$ and outputs a twist as the control action. In this setup, we evaluate 100 randomized start configurations and goal poses, where following the nominal dynamical systems results in intersecting or near-intersecting trajectories, as illustrated in Fig. 3-Vanilla. The start (x, y)-coordinates of the base are randomized with uniform noise (maximum magnitude of 1.0 m) around default start positions, while the manipulator joint positions are perturbed with noise of up to 0.3 rad. Additionally, random noise with a maximum magnitude of 0.3 m is applied on default goal (x, y)-positions. Forward simulations are conducted with a time step of $\Delta t = 0.1$s over a horizon of $K = 200$ steps, with the control parameter $\boldsymbol{\alpha}$ updated every 10 steps for enhanced motion smoothness. For optimization, we use the Optax Adam optimizer [24]. Each of the 100 randomized scenarios is solved using two different initial guesses in a multi-start scheme from which the solution with the lowest objective value is selected. We therefore perform a total of 200 optimization runs in parallel.

We analyze two cases: (1) a non-cooperative setting, where only one agent adapts its policy while the other follows a fixed strategy, and (2) a cooperative policy, where both agents adapt their velocity scaling. We compare these two cases against a vanilla baseline ($\boldsymbol{\alpha} = \mathbf{1}$), and a common TAMP-style heuristic in which the closest robot moves to its goal first while the other waits, and only starts moving once the first robot has passed. The performance of each policy is assessed using four metrics:

- *Goal Reached:* The Euclidean distance to the goal must be within a $0.3\,\text{m}$ margin after 200 time-steps.
- *Task Constraints:* The constraints in Eq. 8 must be satisfied, with $Z = 0.5$ and $d_{\text{g}} = 1.0\,\text{m}$.
- *Collision Avoidance:* The minimum distance between the two robots, Eq. 7, must be larger than the margin of $0.2\,\text{m}$ during the entire movement from start to goal.
- *Closeness to the DS:* The deviation from the original DSs, measured as the average distance from $\boldsymbol{\alpha} = \mathbf{1}$.

Table I and Fig. 3 show that the proposed policy obtains goal reaching and whole-body collision avoidance in both cooperative and non-cooperative settings, compared to the vanilla baseline and heuristic. In contrast, the heuristic produces inefficient motions and often fails to reach the goal within the time window. In the non-cooperative setting, collisions

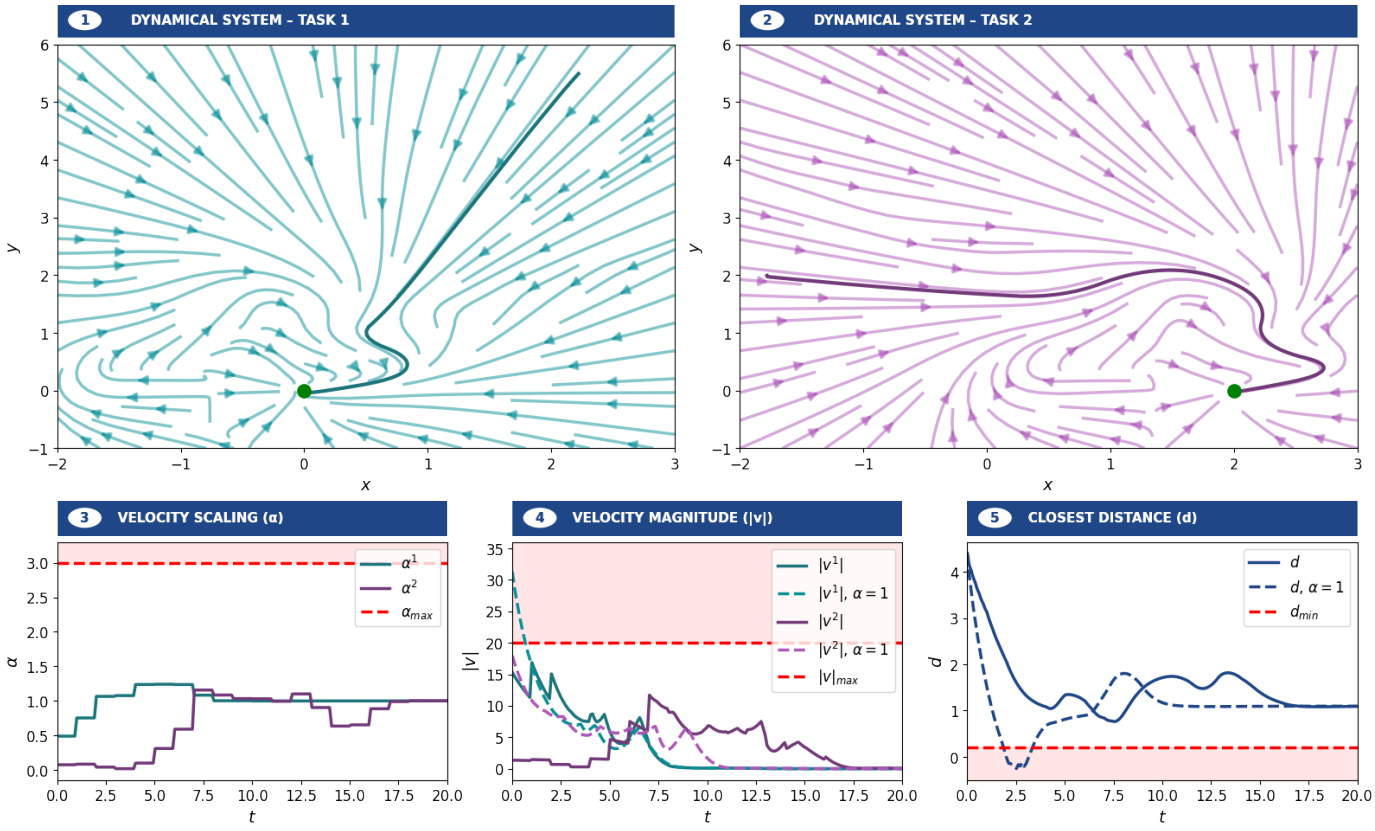

Fig. 1: Figures 2.1–2.2 illustrate the single-agent dynamical systems for Task 1 (executed by Robot 1) and Task 2 (executed by Robot 2), with arrows representing the dynamics. The dark-colored lines shows the trajectories under the proposed velocity scaling in a cooperative multi-agent setting. Figures 2.3–2.5 depict the velocity scaling, velocity magnitude, and whole-body collision distance between the two robots, along with their imposed limits. Dashed lines illustrate the vanilla case without velocity adaptation, i.e. $\boldsymbol{\alpha} = \mathbf{1}$.

(a) Cooperative (ours)

(b) Vanilla ($\boldsymbol{\alpha} = \mathbf{1}$)

Fig. 2: Snapshot of the whole-body movement generated by: (a) the proposed multi-agent cooperative strategy, as observed in Fig. 1, and (b) the vanilla strategy, where each agent executes its learned single-agent policy, resulting in a collision. The positions of the goals are marked by spheres, each color-coded to match its corresponding robot.

still occur in some of the randomized trials compared to the cooperative case (Fig. 3), since robot 2 does not adapt its policy. The non-cooperative setting increases the complexity of the required adaptive maneuver and makes the optimization more susceptible to undesirable local minima. This issue may be mitigated by using more diverse initializations. Task constraint satisfaction is significantly improved over both the vanilla and heuristic baselines, while the proposed method remains closer to the original DSs compared to the heuristic.

In Fig. 1, the solution using the DSs in $\mathbb{R}^2$ requires $4.1\,\mathrm{s}$ of computation time on a laptop equipped with an Intel Core i7 CPU and 16 GB RAM, using 100 solver iterations and $N_{\text{init}} = 3$. Since reactive execution demands faster inference, reducing computation time is an important direction for future

	Goal reached	Task constraints	Minimum distance	Closeness to DS
	$\rightarrow 1$	$\rightarrow 1$	$\uparrow$	$\rightarrow 0$
Vanilla	1	0.340	-0.193 $\pm$ 0.124	0
Heuristic	0.770	0.330	1.119 $\pm$ 0.206	0.5
Non-coop.	1	0.810	0.393 $\pm$ 0.294	0.120 $\pm$ 0.019
Coop.	1	0.890	1.012 $\pm$ 0.195	0.071 $\pm$ 0.012

TABLE I: Statistics for 20 scenarios with two mobile manipulators. We compare a heuristic, a non-cooperative setting, where one robot adapts its policy while the other follows its DS, and a cooperative setting, where both robots jointly adapt their policy through velocity scaling. The vanilla policy serves as a baseline without velocity adaptation. The arrows indicate the desired direction of each metric.

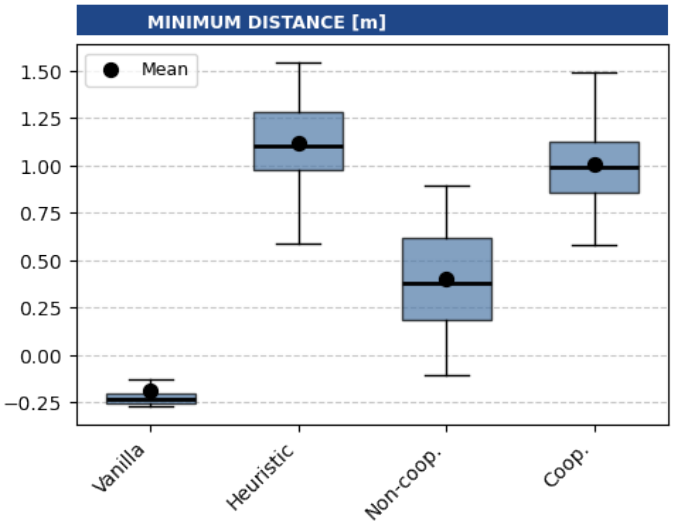

Fig. 3: Minimum distance between the robots' bodies, modeled as sets of capsules.

improvements. In this context, the number of solver iterations and the quality of initializations strongly influence the required computation time: when diverse and effective initial guesses are available, fewer iterations may be sufficient without degrading solution quality. For example, fixing the number of solver iterations to 10, results in a computation time of 0.55 s for $N_{\text{init}} = 3$ or 1.16 s for $N_{\text{init}} = 10$, whereas using 100 solver iterations increases computation time with a factor of 10 in both cases. As observed in Fig. 4, the computation time scales approximately linearly with the horizon length, number of solver iterations, and batch size when solved on CPU, which could be further improved when leveraging GPU. The dominant computational bottleneck lies in repeatedly querying the learned DS, forward kinematics and collision distances during forward predictions. We also observe this bottleneck from the fact that the non-cooperative solution, with half the decision variables compared to the cooperative solution, does not decrease computation time, as it still requires the same amount of forward steps (Fig. 4). When the learned DSs is less involved to query, this significantly decreases computation time. For example, when using the simpler $\mathbb{R}^2$-DSs, 200 parallelized runs takes 20 seconds, compared to two minutes when using $SE(3)$-DSs, given $K = 200$ and 10 solver iterations.

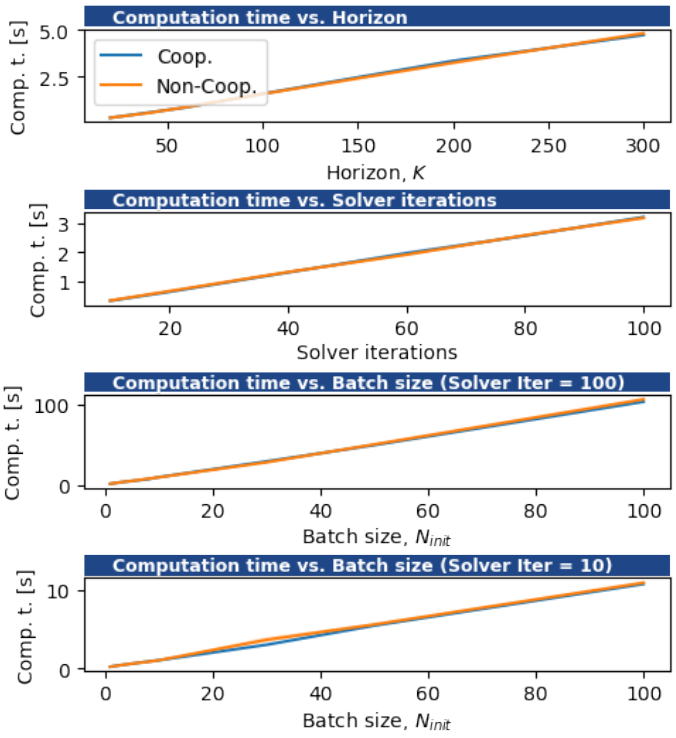

Fig. 4: Computation times using DSs in $\mathbb{R}^2$. Each subplot varies one parameter, while the others are held constant at $K = 200$, maximum solver iterations $= 100$, and $N_{\text{init}} = 2$.

However, the fact remains that forward predictions using the learned DSs will result in non-neglectable computation times. This highlight that learning diverse and informative initial guesses, or directly learning the effect of velocity scaling on the resulting trajectory, thereby avoiding costly rollouts, has potential to further reduce computation times and improve reactive performance.

V. Conclusions and Future Work

We presented a collaborative multi-agent framework that builds on single-agent learned dynamical systems and enables safe interaction through predictive velocity scaling. By preserving the structure of the original vector fields and modifying only their execution speed, the method maintains interpretability and leverages existing convergence guarantees while incorporating whole-body collision avoidance and task-level constraints. Experiments in simulation show that the approach enables effective coordination between mobile manipulators, while preventing whole-body collisions.

Future work will focus on deriving high-level task constraints from a small set of human–human or multi-robot interaction demonstrations. In addition, the current optimization procedure is computationally demanding, as discussed in Section IV, and could be further improved in efficiency. Its performance is thereby sensitive to the quality of the initial solution, suggesting that learning-based approaches could be used to predict more effective and diverse initializations.

References

[1] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 1087–1102, 2023.

[2] J. A. Sánchez-Molina, F. Rodríguez, J. C. Moreno, J. Sánchez-Hermosilla, and A. Giménez, "Robotics in greenhouses. scoping review," *Computers and Electronics in Agriculture*, vol. 219, p. 108750, 2024.

[3] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 239–252, 2022.

[4] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *Proc. of Robotics: Science and Systems*, 2023.

[5] R. Pérez-Dattari, C. Della Santina, and J. Kober, "Puma: Deep metric imitation learning for stable motion primitives," *Advanced Intelligent Systems*, vol. 6, no. 11, p. 2400144, 2024.

[6] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *Proc. of Robotics: Science and Systems*, 2023.

[7] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," in *Conference on Robot Learning (CoRL)*, 2024.

[8] P. Sundaresan, R. Malhotra, P. Miao, J. Yang, J. Wu, H. Hu, R. Antonova, F. Engelmann, D. Sadigh, and J. Bohg, "Homer: Learning in-the-wild mobile manipulation via hybrid imitation and whole-body control," *arXiv preprint arXiv:2506.01185*, 2025.

[9] J. Dong, L. Zhang, L. Zhang, Y. Ling, Y. Fu, K. Bai, Z.-C. Márton, Z. Bing, Z. Chen, A. C. Knoll *et al.*, "M4diffuser: Multi-view diffusion policy with manipulability-aware control for robust mobile manipulation," *arXiv preprint arXiv:2509.14980*, 2025.

[10] S. Singh, T. Xu, and Q. Chang, "Collaborative motion planning for multi-manipulator systems through reinforcement learning and dynamic movement primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 3369–3375.

[11] L. Peters, "Game-theoretic motion planning for multi-agent interaction," Ph.D. dissertation, TU Delft, Delft, The Netherlands, 2026.

[12] C. He, G. S. Camps, X. Liu, M. Schwager, and G. Sartoretti, "Latent theory of mind: A decentralized diffusion architecture for cooperative manipulation," *Proc. Conference on Robot Learning (CoRL)*, 2025.

[13] D. Dong, M. Bhatt, S. Choi, and N. Mehr, "Mimic-d: Multi-modal imitation for multi-agent coordination with decentralized diffusion policies," *arXiv preprint arXiv:2509.14159*, 2025.

[14] S. Saha and A. A. Julius, "Task and motion planning for manipulator arms with metric temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 379–386, 2017.

[15] S. Bakker, M. Schonger, T. Löw, J. Alonso-Mora, and S. Calinon, "Curve-induced dynamical systems on riemannian manifolds and lie groups," *arXiv preprint arXiv:2603.05268*, 2026.

[16] J. Urain, D. Tateo, and J. Peters, "Learning stable vector fields on Lie groups," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 569–12 576, 2022.

[17] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," in *Proc. Conference on Robot Learning (CoRL)*, 2024.

[18] D. Morton and M. Pavone, "frax: Fast robot kinematics and dynamics in jax," *arXiv preprint arXiv:2604.04310*, 2026, submitted to the ICRA 2026 Workshop on Frontiers of Optimization for Robotics.

[19] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for adaptive and reactive robot control: a dynamical systems approach*. MIT Press, 2022.

[20] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv preprint arXiv:1801.02854*, 2018.

[21] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," 2004.

[22] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin *et al.*, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/jax-ml/jax

[23] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[24] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López *et al.*, "Efficient and modular implicit differentiation," *Advances in neural information processing systems*, vol. 35, pp. 5230–5242, 2022.