Reactive Dual-Arm Cube Stacking with Learned Dynamical System Motion Primitives

Nalini Jain, Shivank Gupta, Thomas Stephen Felix
University of Pennsylvania
MEAM 6230: Learning and Control for Adaptive and Reactive Robots

github.com/naljain/ds_stacking

***Abstract*—This paper presents a hybrid dynamical systems (DS) architecture for dual-arm cube stacking in Isaac Sim using two Franka Panda arms sharing a common workspace. The initial approach, learning a Neural DS for all five task primitives, failed due to noisy demonstration labels, unsegmented primitive boundaries, and spurious attractors in the learned fields. The final system assigns DS controllers only to `reach` and `transport`, where free-space reactivity is beneficial, and uses scripted Lula IK for the contact-dependent primitives `grasp`, `lift`, and `place`. We compare a joint-space Neural DS with Lyapunov regularization against a 2-D Cartesian LPV-DS transport baseline, and use Huber–Billard–Slotine modulation to deflect each arm around the other during shared-stack access. With modulation enabled, the system stacked all 6 blocks in 14.4 s of simulated time; with modulation disabled, it timed out before placing any block.**

I. Introduction

The goal of this project was to stack cubes using two robot arms with learned motion primitives. Each primitive, reaching to a cube and transporting it to the stack, is implemented as a DS controller: a vector field

$$\dot{\mathbf{q}} = f(\mathbf{q})$$

that maps the current joint state to a velocity command. Because the command depends on state rather than a preplanned time index, the controller can recover from initial-condition perturbations without replanning.

The all-learned approach did not succeed. Noisy velocity labels, unsegmented primitive transitions, and learned fields with spurious attractors caused the arm to stall or converge to incorrect configurations. A spurious attractor is a fixed point of the learned field that is not the task goal; it manifests during deployment as stalling or confident motion to the wrong state.

The final architecture assigns DS learning only where it provides measurable benefit. `reach` and `transport` use learned DS controllers because they are long free-space motions with variable start states. `grasp`, `lift`, and `place` are scripted with Lula IK because they are short, contact-bounded, and do not require generalization over initial conditions.

This report is organized as follows: Section III describes the architecture and data pipeline; Sections IV–VI detail the DS formulations; Section VII describes the task coordinator; Section VIII reports training diagnostics and deployment results.

II. Background

a) Stable DS.: Methods such as SEDS [1] and LPV-DS [2] learn vector fields subject to conditions that guarantee convergence to the task attractor. LPV-DS provided a stable-by-construction reference for debugging the Neural DS.

b) Neural DS.: A Neural DS parameterizes the vector field with a neural network. Compared to classical formulations, this increases expressivity but introduces sensitivity to data quality. The goal equilibrium is enforced structurally, and a Lyapunov loss penalizes violations of the energy-decrease condition during training.

c) Spurious attractors.: A spurious attractor is a learned fixed point distinct from the task goal. It arises when demonstrations are noisy or when the saved goal label $\mathbf{q}^*$ does not correspond to the terminal state of the demonstrated trajectory.

d) Modulation.: Huber et al. [3] construct a state-dependent matrix $\mathbf{M}(\mathbf{x})$ that attenuates velocity components directed into an obstacle and amplifies tangential components. We apply this framework to the dual-arm case, treating each arm's end-effector as a moving spherical obstacle for the other.

e) Dual-arm coordination.: The task coordinator assigns blocks, reserves stack slots, and gates simultaneous descent. All reactive behavior is produced by the DS and modulation layers, not the coordinator.

III. System Architecture

The system consists of four layers: (1) the Isaac Sim scene with two Franka Panda arms and a shared table; (2) a task coordinator that sequences primitives per arm; (3) primitive controllers [Neural DS, LPV-DS, or Lula IK] depending on the motion type; and (4) a safety layer implementing Cartesian modulation, sampled-link holds, stack-slot reservation, and staggered starts. Fig. 1 shows the full structure.

A. Simulation Environment

The scene (Fig. 2) contains two Franka Panda arms sharing a table. Each arm has dedicated source blocks, but both stack to the same central goal, creating intentional conflict in the stack region. The scene includes dynamic colored cubes, visual stack goal markers, a table-side camera, and optional kinematic carry. Lula IK is used consistently for both data collection and deployment, ensuring that training targets and deployment targets are computed identically.

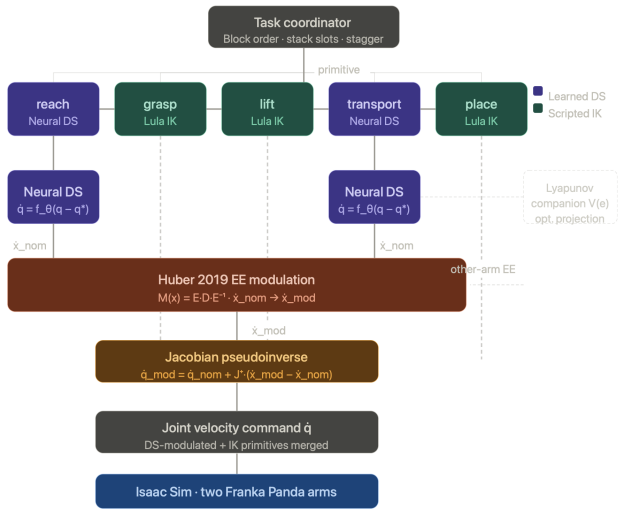

Fig. 1. System architecture. The task coordinator sequences five primitives per block. `reach` and `transport` are driven by the learned DS; `grasp`, `lift`, and `place` are scripted Lula IK. During transport, Huber modulation deflects each arm's Cartesian velocity around the other arm's EE safety sphere before the Jacobian pseudoinverse maps back to joint velocities.

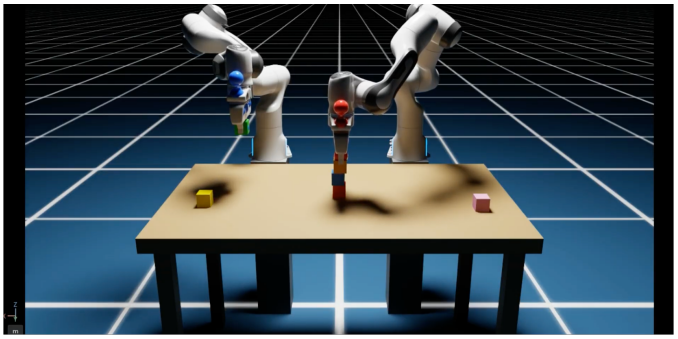

Fig. 2. Isaac Sim setup during active pick-and-place.

B. Primitive Architecture

Each block-stacking cycle decomposes into five primitives in fixed order (Table I). DS learning is restricted to `reach` and `transport`; `grasp`, `lift`, and `place` are scripted because they are short, constrained by contact events, and do not benefit from learned generalization.

TABLE I
PRIMITIVE DECOMPOSITION AND ASSIGNED CONTROLLER.

Primitive	Controller	Description
`reach`	Neural DS (joint)	Hover above source block
`grasp`	Scripted Lula IK	Descend to surface; gripper closes
`lift`	Scripted Lula IK	Raise to transport altitude
`transport`	Neural DS or LPV-DS	Move above stacking goal
`place`	Scripted Lula IK	Descend onto stack; gripper opens

C. Data Collection

Demonstrations are collected with a Lula IK joint-space controller. At each primitive start, Lula IK computes a joint-space goal $\mathbf{q}^* \in \mathbb{R}^7$ from a Cartesian target; the arm then tracks $\mathbf{q}^*$ with a clamped joint-space controller at reduced velocity, yielding cleaner finite-difference velocity labels than high-speed collection. Each timestep records $(\mathbf{q}, \dot{\mathbf{q}}, \ell, \mathbf{q}^*, \mathbf{p}_{\text{ee}})$, where ℓ is the primitive label.

Three pipeline changes were required to produce usable training data:

- **Settling pauses.** The arm pauses between primitives without recording, preventing the DS from learning zero-velocity behavior at non-goal states.
- **Dynamic stack clearance.** Transport targets are computed from the current stack height at collection time, matching deployment behavior.
- **MAD-based demo cleaning.** Demonstrations are filtered by Median Absolute Deviation on path length ($\sum \|\Delta \mathbf{q}\|$) and smoothness ($\sum \|\Delta \dot{\mathbf{q}}\|^2$); IK branch-jump outliers beyond 2.5 MADs are rejected.

D. Early Failure Analysis

The initial Neural DS was trained on uncleaned demonstrations with unsegmented primitive boundaries and was asked to cover all five primitives. The learned fields appeared locally reasonable but failed consistently during rollout, typically due to spurious attractors: the arm would reduce its error for some steps, then stall or curve toward an incorrect configuration.

Two lessons emerged from these failures. First, a stable DS can be made reliable when the state space and training data are appropriately constrained. Second, per-arm convergence is insufficient for dual-arm tasks: the velocity field must be modulated when the other arm occupies the shared workspace.

IV. Neural Dynamical System

A. Formulation

The Neural DS operates in joint-error coordinates. Let $\mathbf{e} = \mathbf{q} - \mathbf{q}^* \in \mathbb{R}^7$ denote the error between the current joint configuration and the primitive attractor:

$$\dot{\mathbf{q}} = f_\theta(\mathbf{e}), \quad \mathbf{e} = \mathbf{q} - \mathbf{q}^*. \tag{1}$$

Operating in error space makes the attractor explicit and avoids runtime IK inversion in the inner control loop. Lula IK is called once at each primitive transition to compute $\mathbf{q}^*$; the DS runs entirely in joint space thereafter.

B. Architecture

The model is parameterised as:

$$f_\theta(\mathbf{e}_n) = \text{net}_\theta(\mathbf{e}_n) - \text{net}_\theta(\mathbf{0}) - \gamma_{\text{skip}}\, \mathbf{e}_n, \tag{2}$$

where $\mathbf{e}_n$ is the normalised error, net_θ is a three-layer MLP with Tanh activations, and $\gamma_{\text{skip}} > 0$ is a fixed skip-connection gain. Subtracting $\text{net}_\theta(\mathbf{0})$ enforces $f_\theta(\mathbf{0}) = \mathbf{0}$ structurally, independent of training outcome. The skip term $-\gamma_{\text{skip}}\, \mathbf{e}_n$ provides a linear convergent prior; the network learns only the nonlinear residual correction.

C. Lyapunov Network

A companion network $g_\phi : \mathbb{R}^7 \to \mathbb{R}^k$ provides a positive-definite scalar Lyapunov candidate:

$$V(\mathbf{e}) = \|g_\phi(\mathbf{e}) - g_\phi(\mathbf{0})\|^2 + \varepsilon \|\mathbf{e}\|^2 . \tag{3}$$

Both terms vanish at $\mathbf{e} = \mathbf{0}$ and are strictly positive elsewhere. Setting $\varepsilon = 0.5$ guarantees positive definiteness even if g_ϕ produces a degenerate output.

D. Training

The total loss combines imitation and a soft stability hinge:

$$\mathcal{L} = \underbrace{\|f_\theta(\mathbf{e}) - \dot{\mathbf{q}}_{\text{demo}}\|^2}_{\text{imitation}} + \lambda_{\text{stab}} \underbrace{\left[\dot{V}(\mathbf{e}) + \alpha V(\mathbf{e})\right]_+}_{\text{stability hinge}}, \tag{4}$$

where $\dot{V}(\mathbf{e}) = \nabla_{\mathbf{e}} V \cdot f_\theta(\mathbf{e})$ is computed via autograd, $\alpha > 0$ controls the exponential decay rate of V, and $\lambda_{\text{stab}} = 0.05$. The hinge activates only when the Lyapunov decrease condition is violated and becomes sparse as training progresses. A scale factor $s = v_{\text{scale}}/\sigma_e$ maps the stability constraint from normalised to real-time coordinates.

E. Safe Projection at Deployment

An optional hard projection (enabled with `--use_safe`) enforces $\dot{V} \leq 0$ at every deployment step:

$$\mathbf{v}_{\text{safe}} = \mathbf{v} - \frac{[\nabla V \cdot \mathbf{v} + \alpha V]_+}{\|\nabla V\|^2} \nabla V, \tag{5}$$

where $\mathbf{v} = f_\theta(\mathbf{e})$. When the training constraint is satisfied, $[\cdot]_+ = 0$ and $\mathbf{v}_{\text{safe}} = \mathbf{v}$; otherwise, the minimum correction along ∇V is applied. This yields two ablation modes: soft stability (training hinge only) and hard stability (runtime guarantee).

V. LPV-DS Reference Experiment

A. Formulation

After the initial Neural DS failures, the LPV-DS served as a reference to isolate whether a stable convergent field could be obtained in a simpler setting. We implement a Linear Parameter-Varying Dynamical System [2] for `transport` in 2-D Cartesian end-effector space $\mathbf{x} \in \mathbb{R}^2$ (the xy plane at constant transport height):

$$\dot{\mathbf{x}} = \sum_{k=1}^{K} h_k(\mathbf{x})\, \mathbf{A}_k\, (\mathbf{x} - \mathbf{x}^*), \tag{6}$$

where $\mathbf{x}^*$ is the Cartesian transport goal, $h_k(\mathbf{x})$ are GMM responsibilities satisfying $\sum_k h_k = 1$, and $\mathbf{A}_k \in \mathbb{R}^{2\times 2}$ are gain matrices.

Global asymptotic stability at $\mathbf{x}^*$ is enforced by the SDP constraint:

$$\mathbf{A}_k + \mathbf{A}_k^\top \prec 0 \quad \forall k. \tag{7}$$

B. Design Choices

a) 2-D xy only.: During transport, z-variation is negligible ($\sigma_z \approx 0.007\,\text{m}$ vs. $\sigma_{xy} \approx 0.12\,\text{m}$). Including z caused the GMM to partition components by height rather than horizontal direction, distorting the velocity field. Isotropic normalisation with scalar σ_x preserves velocity directions in the two relevant dimensions.

b) GMM model selection.: The number of components K is selected by BIC over $K \in \{1, \ldots, 5\}$ on the demonstrated transport distribution.

c) IK wrapping at deployment.: The Cartesian velocity $\dot{\mathbf{x}}_{\text{des}}$ is integrated via:

$$\mathbf{q}_{t+1} = \text{LulaIK}(\mathbf{x}_t + \dot{\mathbf{x}}_{\text{des}}\, \Delta t), \quad \dot{\mathbf{q}} = (\mathbf{q}_{t+1} - \mathbf{q}_t) \,/\, \Delta t. \tag{8}$$

This is equivalent to $\mathbf{J}^+ \dot{\mathbf{x}}$ but uses Lula IK for singularity robustness without perturbing the simulation state.

C. Neural DS vs. LPV-DS

Table II summarises the key trade-offs between the two formulations.

TABLE II
NEURAL DS VS. LPV-DS COMPARISON.

Property	**Neural DS**	**LPV-DS**
State space	Joint error (7-D)	Cartesian xy (2-D)
Stability	Training hinge + optional projection	SDP certificate
Flexibility	High (nonlinear MLP)	Moderate (piecewise-linear)
Training	Sensitive to data & $\mathbf{q}^*$ labels	GMM + one SDP solve
Spurious attractors	Possible if noisy	None by construction
Deployment	Direct $\dot{\mathbf{q}}$ output	IK-based integration

VI. Cartesian DS Modulation

A. Huber 2019 Modulation Framework

In the dual-arm task, both arms may attempt to enter the stack region concurrently. We address this by treating each arm's end-effector as a moving spherical obstacle of radius R for the other. A state-dependent modulation matrix reshapes the Cartesian end-effector velocity near the safety boundary [3]:

$$\mathbf{M}(\mathbf{x}) = \mathbf{E}(\mathbf{x})\, \mathbf{D}(\mathbf{x})\, \mathbf{E}(\mathbf{x})^{-1}, \tag{9}$$

where $\mathbf{E}(\mathbf{x})$ is an orthonormal basis with obstacle-reference direction $\hat{\mathbf{r}} = (\mathbf{x} - \mathbf{x}_{\text{obs}}) / \|\mathbf{x} - \mathbf{x}_{\text{obs}}\|$ as its first column, and $\mathbf{D} = \text{diag}(\lambda_r, \lambda_t, \lambda_t)$:

$$\lambda_r = 1 - \tfrac{1}{\Gamma}, \quad \lambda_t = 1 + \tfrac{1}{\Gamma}, \quad \Gamma = \left(\tfrac{\|\mathbf{x} - \mathbf{x}_{\text{obs}}\|}{R}\right)^p. \tag{10}$$

$\Gamma > 1$ outside the safety sphere and $\Gamma = 1$ on its boundary. As the EE approaches the obstacle, $\lambda_r \to 0$ (radial damping) and $\lambda_t \to 2$ (tangential boost), deflecting motion around rather than into the sphere.

a) Tail-effect gating.: When $\mathbf{v} \cdot \hat{\mathbf{r}} \geq 0$ (the arm is already moving away), $\lambda_r := 1$ and radial damping is suppressed, allowing natural departure from the contention zone.

B. Joint-Space Wrapping (Neural DS)

The Neural DS outputs joint velocities. Modulation is applied in Cartesian space via the translational Jacobian $\mathbf{J}_t(\mathbf{q}) \in \mathbb{R}^{3\times 7}$:

$$\mathbf{v}_{\text{cart}} = \mathbf{J}_t(\mathbf{q})\,\dot{\mathbf{q}}_{\text{nom}}, \tag{11}$$

$$\mathbf{v}_{\text{mod}} = \mathbf{M}(\mathbf{x}_{\text{self}}, \mathbf{x}_{\text{other}})\,\mathbf{v}_{\text{cart}}, \tag{12}$$

$$\dot{\mathbf{q}}_{\text{mod}} = \dot{\mathbf{q}}_{\text{nom}} + \mathbf{J}_t^{+}\,(\mathbf{v}_{\text{mod}} - \mathbf{v}_{\text{cart}}), \tag{13}$$

where $\mathbf{J}_t^{+} = \mathbf{J}_t^{\top}(\mathbf{J}_t\mathbf{J}_t^{\top} + \mu^2 I)^{-1}$ is the damped pseudoinverse. The $+\Delta$ form preserves null-space joint motions (e.g., elbow repositioning) unaffected by modulation. For LPV-DS, modulation is applied directly to $\dot{\mathbf{x}}_{\text{des}}$ in 3-D Cartesian space before IK integration.

C. Sampled-Link Safety Hold

EE-sphere modulation protects only the gripper region. A complementary discrete safety hold samples Franka link poses every N steps and issues a hold to the lower-priority arm when any link pair falls below threshold d_{link}. The hold releases once the separation exceeds $d_{\text{link}} + \delta_{\text{hyst}}$.

VII. Task Coordinator and Deployment

A. Task Coordinator

The coordinator manages: (1) block order and primitive sequence, (2) stack-slot reservation via a shared atomic counter incremented on each `place` completion, (3) dynamic stack-height targets, and (4) return-home parking after each arm finishes. It does not learn a policy.

A `can_place()` yield gate prevents simultaneous descent: the hovering arm holds its `transport` position until the other arm's EE clears `yield_radius` around the shared goal.

B. Deployment Flow

Per-block execution follows the sequence in Fig. 3. IK primitives execute synchronously; `transport` steps are interleaved across arms so that modulation is applied at every simulation tick.

C. Staggered Starts

With symmetric starts, both arms converge on the shared stack simultaneously during their first `transport`, saturating the modulation sphere. A configurable right-arm start delay of $N \approx 200$ physics steps shifts this conflict to the second block cycle, by which point the arms have naturally desynchronised. Staggered starts proved more reliable than relying on continuous modulation alone for this geometry.

VIII. Evaluation

Evaluation addresses two questions. First, do the learned fields exhibit DS-consistent behavior: goal-directed orientation, rollout convergence, and decreasing Lyapunov violations? Second, does the full system complete the stacking task?

Diagnostics are drawn from three scripts: `plot_ds.py` for training and checkpoint analysis, `deploy_single_arm.py --log_csv` for per-step deployment telemetry, and `analyze_deployment_logs.py` for offline rollout evaluation.

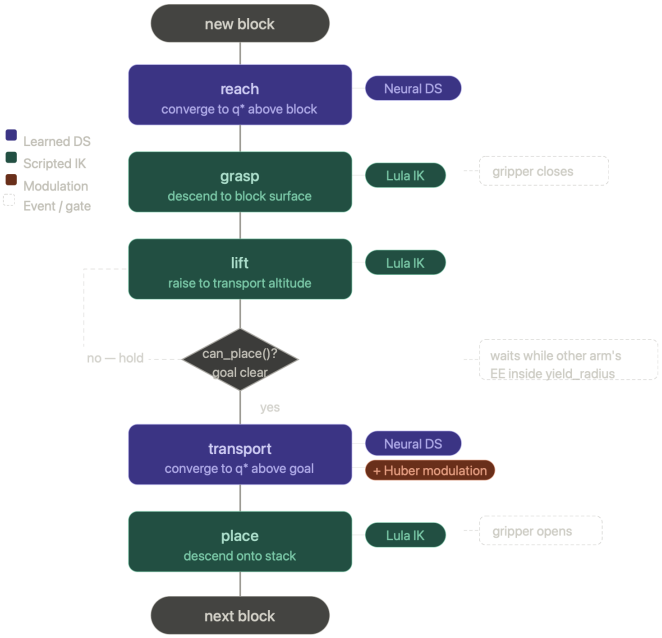

Fig. 3. Per-arm deployment flow. Neural DS runs during `reach` and `transport`; scripted Lula IK handles the three short primitives. The gripper closes after `grasp` and opens after `place`. The `can_place()` gate holds `transport` until the shared goal is clear.

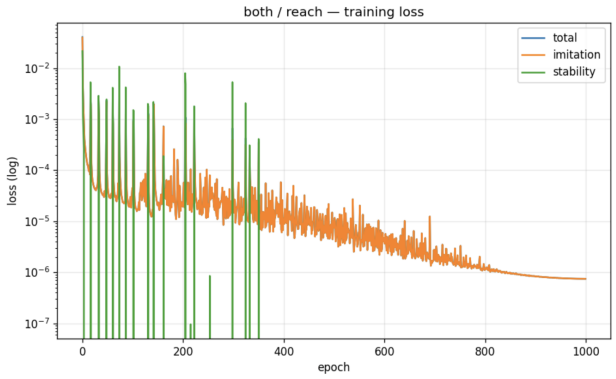

Fig. 4. Training loss for the `reach` Neural DS (1000 epochs). Imitation loss (orange) converges monotonically; the stability hinge (green) is dense early, then becomes sparse as the field satisfies $\dot{V} + \alpha V \leq 0$.

A. Training Diagnostics

Fig. 4 shows training-loss curves for the `reach` Neural DS over 1000 epochs. The imitation loss converges smoothly and monotonically. The stability hinge is dense in early training, when the learned field frequently violates the Lyapunov decrease condition, and becomes sparse after epoch ≈ 400 as the field increasingly satisfies $\dot{V} + \alpha V \leq 0$. Residual spikes beyond epoch 500 reflect intermittent boundary violations outside the training distribution.

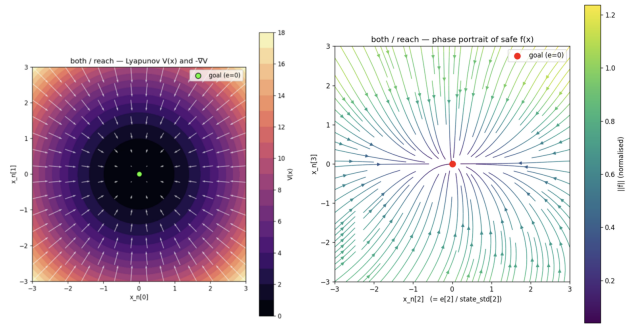

Fig. 5. [Left] Quadratic Lyapunov landscape $V(\mathbf{e}_n) = \|\mathbf{e}_n\|^2$. [Right] Phase portrait of the safe-projected Neural DS in the e_2–e_3 slice. Streamlines converge to the goal (red dot); outward vectors near the boundary indicate the edge of the training distribution.

Table III reports offline checkpoint diagnostics from 100 rollouts per primitive, integrating the DS from $\mathbf{e} \sim \mathcal{U}[-1.5\sigma_e, 1.5\sigma_e]$[7].

TABLE III
OFFLINE NEURAL DS CHECKPOINT DIAGNOSTICS. ROLLOUT RESULTS: 100 ICS, EULER INTEGRATION, $\delta_{\text{done}} = 0.05$ RAD.

Metric	`reach`	`transport`
$\|f_\theta(\mathbf{0})\|$ (rad/s)	$< 1 \times 10^{-6}$	$< 1 \times 10^{-6}$
Rollout convergence rate (%)	94	91
Rollout median conv. time (s)	1.22	1.41
Stability violation rate (%)	3.1	4.8
Imitation MSE (rad^2/s^2)	6.4×10^{-4}	9.1×10^{-4}

B. Vector Field and Lyapunov Landscape

Fig. 5 shows the quadratic Lyapunov landscape $V(\mathbf{e}_n) = \|\mathbf{e}_n\|^2$ alongside the phase portrait of the safe-projected DS in a 2-D joint-error slice. All sampled streamlines converge to the goal (red dot); no spurious attractors are visible. Outward-pointing vectors near the boundary identify states where the soft training hinge required augmentation by the runtime projection.

C. Closed-Loop Rollout Convergence

Fig. 6 shows closed-loop rollouts of the safe-projected DS from 12 perturbed initial conditions. Most joint-error norms reach $\delta_{\text{done}} = 0.05$ rad within 2–3 s. A subset plateau near zero without diverging, consistent with safe-projection stall at out-of-distribution states. Velocity norms remain below the clip limit throughout.

D. Dual-Arm Deployment Results

Table IV reports task-level metrics for a complete dual-arm stacking run and an ablation with modulation disabled.

With modulation enabled, all six blocks were stacked in 1,729 simulation steps (≈14.4 s, 2.40 s per block). Stack-height slots 0–5 were filled in alternating arm order (left: 0, 2, 4; right: 1, 3, 5), with block z-heights rising from 0.762 m to 0.997 m in uniform 0.047 m increments. No IK abort, DS timeout, or safety-hold event occurred.

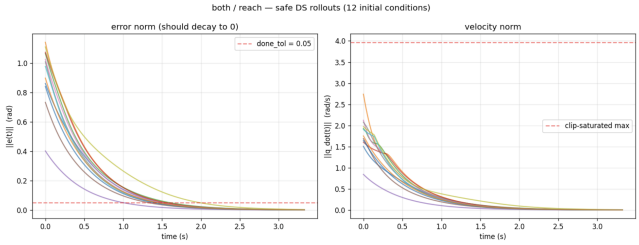

Fig. 6. Closed-loop Neural DS rollouts from 12 perturbed initial conditions (`--use_safe`). [Left] Joint-error norms; most converge below $\delta = 0.05$ (dashed) within 3 s; a small subset plateau, reflecting safe-projection stall outside the training distribution. [Right] Velocity norms remain below the clip limit throughout.

TABLE IV
DUAL-ARM DEPLOYMENT RESULTS. PHYSICS TIMESTEP $\Delta t = 0.00833$ S; 6 BLOCKS TOTAL (3 PER ARM); `TRANSPORT` DRIVEN BY NEURAL DS WITH SAFE LYAPUNOV PROJECTION; SHORT CONSTRAINED PRIMITIVES DRIVEN BY LULA IK.

Metric	Modulation ON	Modulation OFF
Stack completion	**Success**	Failed
Blocks placed	6 / 6	0 / 6
Completion rate	100%	0%
Total sim. steps	1,729	4,500 (timeout)
Approx. simulated time	14.4 s	37.5 s
Avg. time per block	2.40 s	–
Timeout / abort	No	Yes
Failure primitive	–	`left/transport`
Final joint error	–	0.074 rad
Final Cartesian error	–	0.053 m
Safety-held steps	–	0
Return-home	2 / 2 arms	–

With modulation disabled, the system failed during the first simultaneous `transport`. The left arm timed out at 4,500 steps with a joint error of 0.074 rad and Cartesian error of 0.053 m. The absence of safety-held steps rules out the link-safety hold as a contributing factor; the failure is attributable to the unmodulated DS being unable to reach its transport attractor while the other arm occupied the shared workspace. **Interpretation.** With modulation and Lyapunov safe projection enabled, the system completes the task reliably. Without modulation, it fails before placing the first block. The Cartesian modulation layer is a functional necessity for shared-stack access, not an optional safety guard.

IX. DISCUSSION

a) Data quality determines DS quality.: The most impactful improvements were pipeline decisions, not architectural ones. Settling pauses, consistent Lula IK usage across collection and deployment, and MAD-based demo filtering each contributed to cleaner training distributions. High-speed collection and RMPflow-based labeling produced inconsistent velocity labels that caused spurious attractors regardless of network architecture.

b) Hybrid primitive assignment is principled, not pragmatic.: Attempting to learn DS models for all five primitives introduced instability. `grasp`, `lift`, and `place` are bounded

by contact events, not smooth convergence; their learned fields did not generalize. Restricting DS learning to `reach` and `transport` was the single most impactful architectural decision.

c) The Lyapunov safe projection changes the failure mode.: Without `--use_safe`, the field can diverge outside the training distribution. With projection, the arm stalls rather than flails, and the deployment trace satisfies $\dot{V} \leq 0$ at every step. The offline rollout experiment confirms this: projection replaces diverging trajectories with plateauing ones.

d) The Jacobian round-trip is a functional necessity.: The dual-arm ablation demonstrates this directly: with modulation, all 6 blocks were stacked in 14.4 s; without it, the system timed out at 37.5 s with 5.3 cm Cartesian error on the first transport. Zero safety-held steps in the failed run rule out the link-safety hold as the cause. The correction $\|\Delta\mathbf{v}_{\text{proj}}\|$ is small and intermittent in successful runs, confirming that modulation-induced Lyapunov violations are transient and bounded by the $+\Delta$ form.

e) $\cos(\dot{\mathbf{q}}, -\mathbf{e})$ *is the most informative deployment metric.:* Per-step directional alignment between the commanded velocity and the goal direction is a more sensitive indicator of controller quality than convergence time alone. A field maintaining $\cos(\dot{\mathbf{q}}, -\mathbf{e}) > 0.7$ throughout a primitive is reliably converging; values near zero or negative indicate stall or field divergence. This metric is computable directly from the deployment CSV.

f) LPV-DS as a diagnostic tool.: The LPV-DS provides an SDP-certified convergence guarantee with no Jacobian round-trip, at the cost of 2-D Cartesian scope that precludes capturing elbow-trajectory structure. Its value in this project was diagnostic: it helped decouple the problem of learning a convergent single-arm field from the problem of modulating it for dual-arm conflict. In the `transport` primitive, both methods produced comparable convergence; the Neural DS advantage should be more pronounced in workspaces with significant joint redundancy.

X. Conclusions

A hybrid architecture combining learned DS controllers for free-space primitives with scripted Lula IK for contact-bounded primitives successfully completed a 6-block dual-arm stacking task in simulation. The primary failure mode of the all-learned approach was spurious attractors introduced by noisy demonstrations and poor primitive segmentation. Cleaning the data pipeline and restricting DS learning to `reach` and `transport` resolved this.

Key findings:

- Demonstration label quality is as consequential as network architecture.
- Spurious attractors are the dominant failure mode; vector-field phase portraits are a more reliable diagnostic than training loss alone.
- $\cos(\dot{\mathbf{q}}, -\mathbf{e})$ from the deployment CSV provides a lightweight field-quality indicator requiring no additional tooling.
- Offline rollout convergence from 100 random initial conditions characterises the DS without requiring Isaac Sim.
- The Lyapunov safe projection converts divergence failures to stall failures - a safer deployment outcome.
- Cartesian modulation with $+\Delta$ Jacobian wrapping is required for shared-stack access: without it, 0/6 blocks were placed; with it, 6/6 were placed in 14.4 s.
- LPV-DS is competitive for `transport` but cannot represent the joint-space redundancy structure learned by the Neural DS.

Future work should integrate safe projection and modulation into a unified framework, extend the safety hold to continuous link-mesh representations, and evaluate whether joint-space elbow structure confers measurable advantages in more constrained workspaces.

References

[1] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[2] N. Figueroa and A. Billard, "A physically-consistent Bayesian non-parametric mixture model for dynamical system learning," in *Proc. Conf. on Robot Learning (CoRL)*, 2018.

[3] L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of convex and concave obstacles with convergence ensured through contraction," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1462–1469, 2019.

[4] A. Abyaneh, M. Guzmán-González, H.-C. Lin, O. S. Oguz, and S. Calinon, "Globally stable neural imitation policies," in *Proc. IEEE ICRA*, 2024.

[5] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation — a survey," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340–1353, 2012.