STEERING PHYSICS IN VIDEO WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Video world models learn representations that support prediction and planning, and a growing body of work finds that these representations capture physics and dynamics. For example, velocity and acceleration can be read from the representations of V-JEPA 2. However, knowing that a quantity is encoded does not tell us where it is represented, and without that we cannot steer it. In this work, our goal is to edit the representation so that the decoded video carries out a requested physical instruction, such as making a ball move faster. Using simulation datasets, we investigate velocity and acceleration, both linear and angular. We first observe that translation and rotation trace very distinct shapes in the latent space, and that rotation splits along two axes. This structure motivates two steering writers, an additive writer that steers translation, and an orientation field that steers rotation. Measured from the motion in the decoded video, the additive writer controls both the direction and the speed of translation, with direction errors below 10° and speed correlations of 0.750 in 2D and 0.861 in 3D. For rotation, the orientation field sets how fast the object spins, and the requested and decoded spin rates correlate at $\rho = 0.942$, holding at 0.945 when the object's appearance changes. The same field, fitted only on objects spinning at a constant rate, also makes objects speed up or slow down their spin, with no refitting. Finally, we use an edited representation to steer a simulated robot arm, which strikes a ball within 2% of the requested speed on all 200 test commands. Together, these results show that a video model's latent space can serve as a two-way interface to physics, in which physical quantities can be measured, set, and used as goals to plan actions in the real world.

1 INTRODUCTION

World models learn how an environment evolves, so that an agent can predict the consequences of its actions and plan with them (Ha & Schmidhuber, 2018; LeCun, 2022; Hafner et al., 2025). V-JEPA 2 learns such representations from video and uses them for prediction and action-conditioned planning (Bardes et al., 2024; Assran et al., 2025). A growing body of work finds that these models also learn physics, and a large generative video model even solves physical tasks it was never trained for (Wiedemer et al., 2025). For V-JEPA 2 in particular, it was shown that physical quantities such as velocity and acceleration can be read from these representations with probes, and the readouts change when the representation is edited (Joseph et al., 2026). However, knowing that a quantity is encoded does not necessarily guide how it can be changed. Here, we consider a stricter question: after steering the latent, can we decode to recover the scene with a targeted physical change? Concretely, we have a ball moving at velocity a; can we steer the latent so that after decoding it moves at velocity b? If we can achieve this, the representation becomes an interface through which motion can be both measured and modified. As an application, we show that a robot could then perceive the edited representation as its goal and choose the action that makes the object move as desired.

With this goal in mind, we study linear and angular velocity and acceleration using simulated datasets. We start by examining the shape of these quantities in the latent space. By applying principal component analysis (PCA), we find that translation and rotation yield very different geometric structures. To understand the rotational paths, we train a linear probe to predict the object's orientation. This reveals that an object rotating at any angular velocity passes through the same orientations and differs only in how quickly it moves through them. When the predicted orientation is unwrapped, that is, tracked as a continuous angle that accumulates across full turns, the unfolded shapes of angular velocity/acceleration resemble those of linear velocity/acceleration. Rotation therefore separates

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

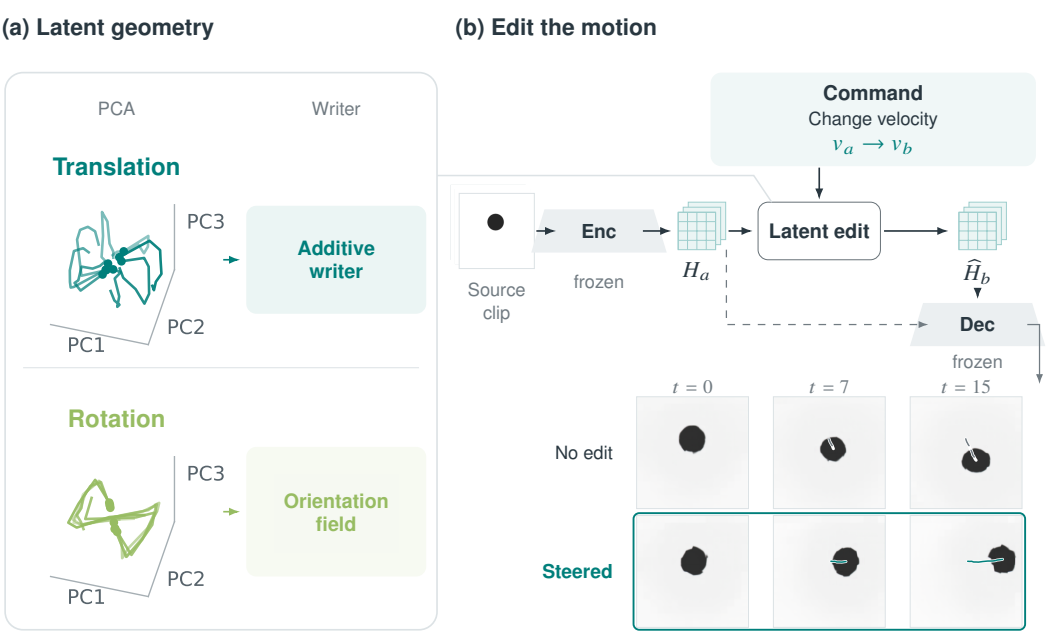


Figure 1: **Latent motion structure guides physical steering.** (a) PCA views of frozen ViT-L features motivate an additive writer for translation and an orientation field for rotation. (b) A source representation and physical command produce an edit without a target clip. A decoder renders the result; independent tracking measures the motion. The filmstrip shows the ViT-L additive writer with magnitude axes on one held-out 2D test pair, with the same decoder before and after steering.

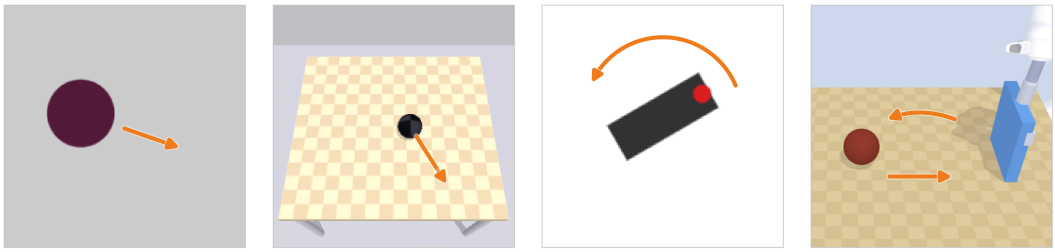
into the path of orientations the object passes through and the rate at which it moves along that path. This separation motivates two different steering methods, which we call *writers*, each with an edit parameterization suited to its motion. For rotation, the writer acts on the orientation path and the rate separately, so it can change how fast an object rotates without changing which orientations it passes through.

Specifically, translation is steered by an additive writer, which shifts the representation along learned directions. When these directions are learned from labeled speeds, they capture speed changes that the directions found by PCA do not. Rotation is controlled by an orientation field, a writer defined over orientation, which specifies how the representation should change at each orientation. Because this writer depends on orientation rather than time, it can be applied when the object rotates at a new rate. We compare these writers across 6 motion types and evaluate the resulting edits by decoding them into video with a separately trained, frozen decoder and measuring the motion with an object tracker (Figure 1). At test time, a writer receives only a source representation and a physical command, with no clip of the target motion. We also test transfer to angular acceleration, robustness to object appearance, and composition of edits.

Finally, we use an edited representation as the goal for a simulated robot arm that strikes a ball. Given a requested outgoing speed, the additive writer turns the representation of the scene before contact into a latent goal for the scene after contact, and a separately trained policy chooses a strike from that goal. On all 200 test commands, we find that the ball leaves within 2% of the requested speed.

2 Related Work

JEPAs and V-JEPA 2 learn video representations for prediction and planning (Bardes et al., 2024; Assran et al., 2025), while physically structured objectives and probes study the dynamics latent features encode (Liu et al., 2026; Zeng et al., 2026; Joseph et al., 2026). Feature interventions, controllable video generation, and latent planners offer other ways to influence model behavior or motion (Turner et al., 2023; Wang et al., 2024; Zhou et al., 2025). We edit the frozen representation of



(a) 2D Moving Ball (b) 3D Moving Ball (c) Spinning Bar (d) Robotic Paddle Strike

Figure 2: **Datasets.** (a) A 2D moving ball. (b) A moving ball in a rendered 3D MuJoCo scene. (c) A marked spinning bar. (d) A simulated robotic paddle strike. The first three provide controlled clips for probing and steering; the last tests whether a controller can use a steered representation as a goal. Orange arrows illustrate motion.

an observed clip using a physical command, measure the resulting motion in decoded video, and use the edited representation as a latent goal for control. Further related works are given in Appendix A.

3 Experimental Setup

Model. We use the Joint Embedding Predictive Architecture model V-JEPA 2 as our world model. Most experiments use the pretrained ViT-L (300M) checkpoint, and we repeat key experiments on ViT-H (600M) and ViT-g (1B) to test robustness. For each model we record the residual stream at four layers placed at similar relative depths, namely layers 6/12/18/23 for ViT-L, 8/16/24/31 for ViT-H, and 10/20/30/38 for ViT-g.

Datasets. For our experiments we use synthetic environments. Each clip follows the V-JEPA 2 input format of 16 frames at 256×256 resolution and 4 fps. Figure 2 shows the four main environments, a two-dimensional (2D) ball moving, a three-dimensional (3D) ball rolling on a table, a rotating bar with a red marker at one end, and a simulated robotic paddle strike. For these environments we also vary ball color, radius, and background color to ensure robustness of our steering methods. We introduce the steering methods in Section 5, after the analysis that motivates them. Further details are described in Appendix B.2. We organize the data into scenes. A *scene* consists of eight clips that share their initial physical conditions and appearance and differ only in one physical quantity q, and we call each value of q a *motion setting*. Let $x(s, q)$ be the clip from scene s with motion setting q, let E_ℓ denote the frozen encoder up to layer ℓ, and let $H_\ell(s, q) = E_\ell(x(s, q))$ be the clip's layer-ℓ representation.

Probes and decoders.

- **Probe.** Before steering we also probe to read off physical quantities from the latents. We mainly use linear probes, which is a ridge regression from the latent representation to the physical quantity, and its output is a *readout*. Probes predict velocity from pooled tokens.
- **Decoder.** To see whether an edit changes the motion, we decode the edited representation back into video with a separately trained, frozen decoder, a 12-layer transformer (208.7M parameters) that reads the four recorded layers and renders all 16 frames. We then track the object in the decoded frames to measure the motion it actually carries out. Besides pixel losses, the decoder is trained with a term that places the object at its true position in every frame, because with pixel losses alone even an unedited representation decodes to a ball moving about four times too slowly. Each dataset has its own decoder with the same architecture and training, and Appendix B.3 gives the details.

4 Latent Representation of Physical Quantities

Joseph et al. (2026) showed that physical quantities can be recovered from V-JEPA 2 representations with probes. We confirm this in our controlled 2D tasks, where probes of frozen ViT-L features recover

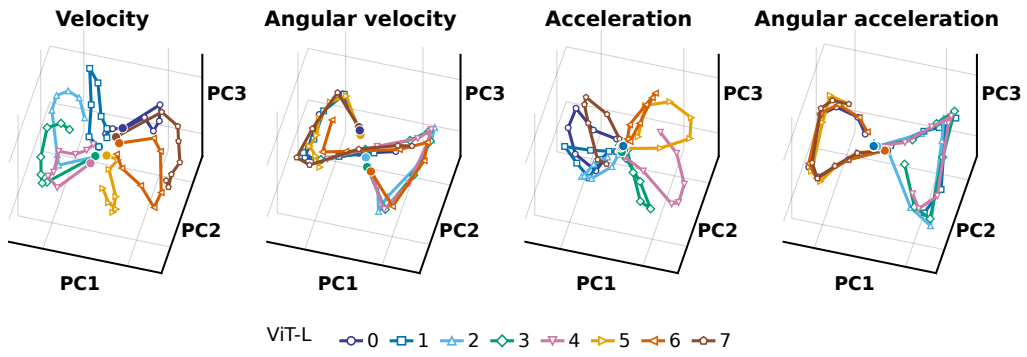


Figure 3: **PCA results.** Each panel varies one quantity over eight motion settings (colors and markers). Lines join a clip's eight temporal tokens, and filled circles mark the first.

linear and angular velocity and acceleration with held-out R^2 between 0.78 and 0.995 (Table 1). These scores measure how well each quantity can be read, which says nothing yet about whether it can be controlled in decoded video. Appendix C.4 gives separate position and orientation readouts used to interpret the latent trajectories. We next examine the structure of the four motion quantities in the representation to guide steering.

Table 1: Held-out probe R^2 for motion quantities in frozen ViT-L layer-12 features.

Quantity	Linear	Angular
Velocity	0.995	0.925
Acceleration	0.918	0.783

Following Musa et al. (2026), we project latent trajectories onto their top three principal components. We take the eight clips of one scene and extract the frozen representation at layer 12. Each clip yields 8 temporal tokens on a 16×16 spatial grid, so the scene gives 64 points in total. We rank the spatial tokens by their activation variance across these 64 points, summed over channels, and keep the top 16. At each point we concatenate the 1024-channel activations of those 16 tokens, fit PCA to the 64 resulting vectors, and plot each clip's projection onto the first three components in temporal order. Figure 3 shows the results, and we draw two observations from it.

Translation and rotation follow different latent trajectories. We observe that clips with different velocities spread into an open fan of paths, while clips with different angular velocities follow overlapping curved paths. Velocity and acceleration share the same latent structure, as do angular velocity and angular acceleration. The same shapes appear for ViT-H and ViT-g at similar relative depths (Appendix C.2). These differences suggest that translation and rotation need distinct steering methods.

The PCA view suggests that rotating objects repeatedly visit a curved region of the representation, and that the motion setting changes how quickly they move through it. The projection alone does not tell us what the curved path represents. We therefore ask whether the bar's orientation provides a useful coordinate for the full representation, and how that coordinate evolves over time.

Unwrapping rotation reveals its orientation schedule. To see whether orientation is readable beyond the PCA projection, we fit linear probes on the full representation, holding out each motion setting in turn. Figure 4b,e shows their $(\cos 2\theta, \sin 2\theta)$ predictions. We use 2θ because the bar's silhouette repeats after a half turn, even though its end marker distinguishes full turns in the video. The predicted *wrapped phase*, $\phi_t = \operatorname{atan2}(\widehat{\sin 2\theta_t}, \widehat{\cos 2\theta_t}) \in (-\pi, \pi]$, jumps between π and $-\pi$ as the bar keeps rotating.

We *unwrap* each clip by adding or subtracting 2π at these jumps, which makes its phase continuous. The change from the first token then estimates twice the accumulated rotation. Panels (c,f) subtract

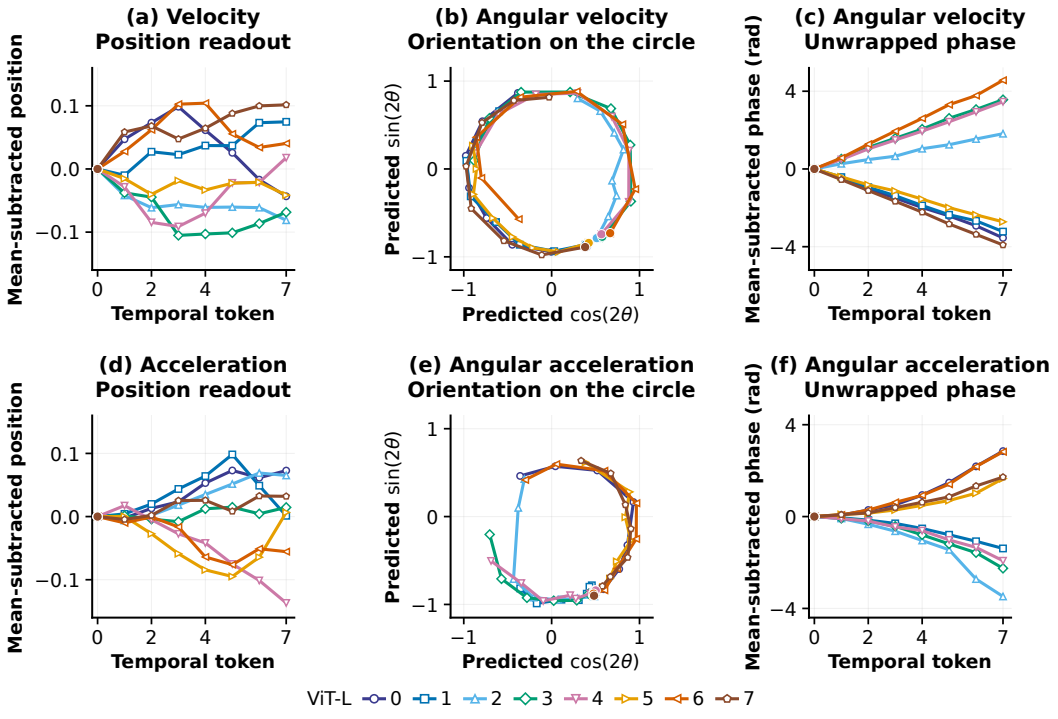


Figure 4: Position and orientation readouts from the ViT-L representation. Each curve holds out one of eight motion settings during probe fitting. (a,d) Predicted x position versus temporal token. (b,e) Predicted $\cos(2\theta)$ and $\sin(2\theta)$. (c,f) Phase unwrapped from those same predictions and plotted against time. The time plots subtract the mean across settings at each token, then shift each curve to start at zero. Colors and markers identify settings; filled circles mark the first token. Lines join predictions without smoothing. These are linear-probe readouts of position and orientation, not PCA projections or latent edits. Appendix C.4 gives the fitting protocol and scores.

the mean unwrapped phase across settings at each token and shift each trace to start at zero. In these relative traces, measured in radians of 2θ, constant angular velocity gives nearly straight lines and angular acceleration gives curved ones.

These readouts motivate the two parts of the rotation writer. The first is an orientation schedule, set by the initial orientation and the motion command. The second is a field that describes how the representation varies with orientation at each token. The same field can then be queried with a linear schedule for constant angular velocity or a quadratic schedule for constant angular acceleration.

5 From Latent Structure to Physical Edits

We turn the observed structure into two writers, where a writer is a learned map from a source representation and a physical command to an edited representation. The additive writer handles translation, and the orientation field handles rotation. Both are trained on pairs of clips from the same scene. For two such clips of scene s with motion settings q_a and q_b, the training target is the difference between their layer-ℓ representations, $\Delta H^\star_\ell = H_\ell(s, q_b) - H_\ell(s, q_a)$, which we call the *true edit*. A writer's estimate of it is the *predicted edit*. Adding the true edit to the source representation gives the target clip's representation at every token.

At test time, the writer receives source representation, a source value q_a, and a requested value q_b:

$$\widehat{H}_\ell(s, q_b) = \mathcal{T}_\ell(H_\ell(s, q_a), q_a, q_b). \tag{1}$$

We abbreviate $H_\ell(s, q_a)$ and $H_\ell(s, q_b)$ as H_a and H_b, and drop ℓ when the layer is clear. We note that the writer never sees the target clip or its representation. For rotation, we also supply the object's

center and initial orientation in the source clip. For linear and angular acceleration, the initial velocity or angular velocity stays the same as in the source clip. We fit a separate writer at each of the four recorded layers and apply all four edits together, since the decoder reads all four layers.

5.1 TRANSLATION: WRITING DIRECTION AND MAGNITUDE

The additive writer predicts a low-dimensional latent displacement:

$$\widehat{\Delta H} = gUW\phi(\mathbf{q}_a, \mathbf{q}_b), \qquad \widehat{H}_b = H_a + \widehat{\Delta H}. \tag{2}$$

The columns of U are directions in which the writer can change the representation. Ridge regression, a regularized linear fit, learns the command map W from the true edits of the training scenes. The command features (ϕ) include the velocity components, the speeds, and the unit headings of the source and requested velocities, and each feature is divided by its root-mean-square value over the training set before the fit. A scalar gain g is selected on validation scenes.

The output basis must retain magnitude as well as heading. The leading principal components of the true edits in the training set capture heading well but can miss changes in magnitude, and even projecting the true edit onto the PCA basis lowers the magnitude correlation (Table 8). We therefore append three *magnitude axes*, directions fitted with supervision on the change in speed, so the writer can recover what the PCA basis misses. Linear acceleration uses the same construction, with acceleration commands and magnitude axes fitted to the change in acceleration magnitude. Appendix D.1 describes the fit, and Appendix D.2 reports the basis ablation.

5.2 ROTATION: EDITING ALONG AN ORIENTATION FIELD

The unwrapped phase view in Section 4 suggests controlling rotation by specifying the orientation the object should have at each time. The writer builds this schedule from the source clip's initial orientation and the physical command. It then converts the difference between the source and requested orientations at each time into a change in the latent representation.

We first shift the 16×16 token grid so that the object's center sits at the grid center, and write $\widetilde{H}_t$ for this recentered representation at token t. At each temporal token t, we fit the orientation field $C_t\varphi_K(\theta)$, a harmonic function of orientation, where C_t is a coefficient matrix fitted by ridge regression to $\widetilde{H}_t$ and

$$\varphi_K(\theta) = [1, \cos\theta, \sin\theta, \ldots, \cos K\theta, \sin K\theta]^\top, \qquad K = 4. \tag{3}$$

For constant angular velocity, the continuous schedule is $\theta_\omega(t) = \theta_0 + \omega\tau_t$, where θ_0 is the source clip's initial orientation and τ_t is the time associated with token t. We write θ_a and θ_b for the schedules at the source and requested angular velocities. The edit after recentering is

$$\widehat{\Delta\widetilde{H}}_t = gC_t[\varphi_K(\theta_b(t)) - \varphi_K(\theta_a(t))]\,. \tag{4}$$

The same change in angular velocity can therefore produce different latent displacements at different starting orientations. The orientation field captures how the representation looks at each orientation, and the schedule specifies when each orientation occurs. After computing the edit, we undo the recentering, so that the edit applies at the object's original location, before decoding. We fit the field on clips with constant angular velocity and select the scalar gain g on validation scenes. Section 6 tests whether the same field can be reused under a quadratic schedule, which corresponds to constant angular acceleration.

5.3 TESTING THE EDIT PARAMETERIZATION

We test these choices by fitting both kinds of writer to every type of motion. The first is the additive writer, which predicts the edit from the command alone. The second is a *configuration-conditioned writer*, or *conditioned writer* for short, which learns a function of position or orientation and takes as its edit the value of that function at the target configuration minus its value at the source configuration. Six controlled motion types cover both translation and rotation, each with first-order and second-order motion. Both writers train on the same scenes, and we score each by how closely its predicted edit aligns with the true edit on every ordered pair of clips in held-out scenes, before any decoding. The

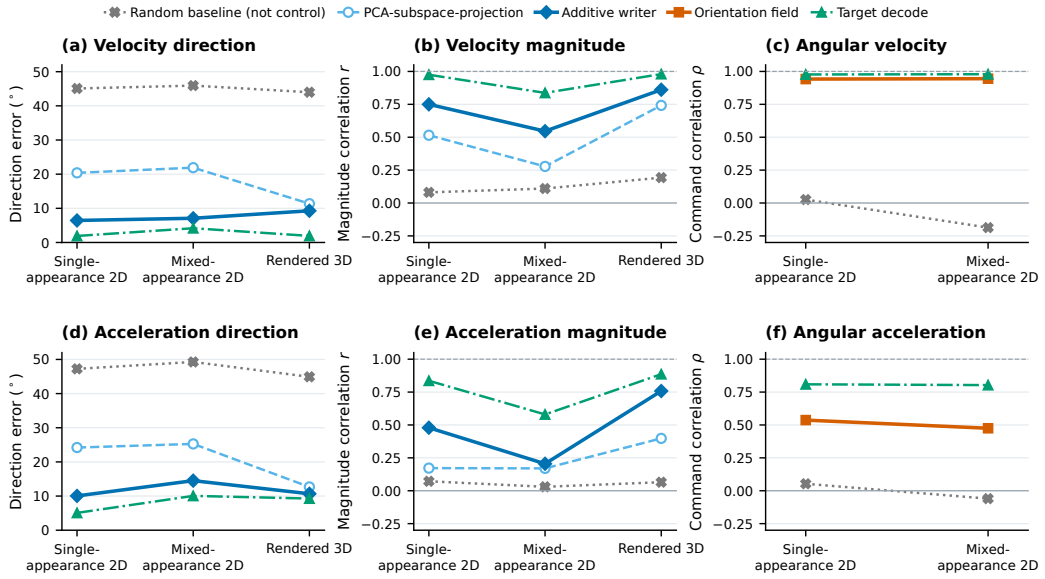


Figure 5: Decoded motion from held-out edits. Top: velocity direction error, speed correlation, and angular-velocity correlation; bottom: the corresponding acceleration measures. Translation uses 50 test scenes per setting. Rotation writers and shuffled-command baselines use 105 test pairs; rotation target decodes pool 210 validation and test pairs. The translation random and PCA-subspace projections require the target edit, and target decodes require the target representation. Lower direction error and higher correlation are better.

conditioned writer aligns better for both rotational motion types and the additive writer aligns better for all four translational ones, as the PCA and phase views suggested. Table 5 in Appendix C.1 gives the scores, and Appendix C.3 gives the fitting protocol. We next test whether the edits produce the requested motion after decoding.

6 Steering physical quantities in decoded video

We now test the writers end to end (Figure 1). A command produces a latent edit, the frozen decoder renders the edited representation as video, and a tracker measures the resulting motion. We also decode the representation of the actual target clip, which we call the *target decode*. It shows what the decoder can render when given the correct input. Training, gain-selection, and test scenes are disjoint. Appendix B.3 validates the decoder and trackers, and the ranking of methods holds under a second decoder trained with a different seed (Appendix B.6).

Figure 5 summarizes the results on the 2D, mixed-appearance 2D, and rendered 3D datasets. For translation we compare against two projections of the target edit, one onto a random subspace and one onto the PCA subspace, and for rotation against shuffled commands. The projections and the target decode all see target information that the writer never receives, so they show how much of the motion can be recovered with that information. Correlation alone can hide a weak response, since decoded motion can follow the command while realizing only a fraction of it. We therefore also report the *response slope* from regressing achieved on commanded magnitude, where a slope of one means the full command is realized. Appendix B.4 defines every metric and the gain selection.

Every writer is also evaluated on mixed-appearance scenes, where ball color, radius, and background vary. Direction and angular-velocity control carry over, and only the response slope drops (Appendix F.1). Steering also works with ViT-H and ViT-g encoders and their own decoders, but larger encoders do not reliably improve control (Appendix F.3).

Velocity direction and speed. The additive writer follows the commanded heading in all three settings, with direction errors between $6.45°$ and $9.28°$. Its speed response depends more on the

setting. In rendered 3D the response slope is 0.918, so the writer realizes most of each requested change in speed, and in mixed-appearance 2D it is 0.354, about a third of the requested change. Speed correlation alone therefore does not show whether the writer achieves the commanded magnitude (Figure 13).

Adding supervised speed axes to the PCA basis improves speed tracking in every setting with little change in heading accuracy. Because the command features and selected gains match those of the PCA-only writer, this ablation isolates the effect of retaining magnitude directions in the edit basis (Table 7). The mixed-appearance writer still under-steers after this addition. Appendix D.2 gives the gain and basis ablations. Velocity edits also compose. Applying two edits in succession gives nearly the same decoded velocity as a single direct edit, with discrepancies under 4% of the steering effect, and each edit is substantial, cutting direction error from 172° to 9° (Appendix F.2).

Angular velocity. The orientation field gives reliable directional control. Every test edit rotates in the commanded direction, and the correlation between commanded and decoded angular velocity stays high under both appearance conditions, at 0.942 and 0.945 (Figure 5). Mixed appearance still lowers the response slope from 0.741 to 0.668, so the nearly unchanged correlation hides a weaker change in rotation rate. With shuffled commands the correlation falls to near zero, which shows that the decoded motion follows the requested command (Table 11). Appendix E.2 gives the harmonic ablations and decoder checks.

Linear acceleration. Acceleration direction remains steerable, with errors between 10.04° and 14.51°, and the size of the decoded acceleration follows the command less closely than it does for velocity, particularly under mixed appearance (Figure 5). Adding magnitude axes raises the magnitude correlation from 0.359 to 0.479 in single-appearance 2D and from 0.671 to 0.758 in rendered 3D, and leaves it almost unchanged under mixed appearance, at 0.205 (Table 7). On those mixed-appearance scenes the target decode itself reaches only 0.580, which points to the decoder as part of the limit. The gap between 0.205 and 0.580 still leaves room to improve the writer. Appendix D.3 tests temporal structure, and Appendix D.4 probes the frozen decoder with per-scene search.

Angular acceleration from a velocity-trained field. Changing only the schedule produces a measurable angular-acceleration response, without fitting a new field on accelerating clips. The correlation between commanded and decoded angular acceleration reaches 0.537 for single appearance and 0.475 for mixed appearance, against 0.053 and -0.060 for shuffled commands, so the decoded second-order motion follows the command. The target decode reaches 0.809 and 0.803, so the transferred control remains incomplete (Figure 5). A field refitted on accelerating clips reaches 0.512 for single appearance, which is no better than transfer (Table 12). Table 13 gives the harmonic ablations.

6.1 From steered physics to action

An edited representation can also specify a goal for a controller. In a simulated head-on strike, the additive writer takes the representation of the scene before contact and a requested outgoing ball velocity, and constructs a *latent goal*, which is an edited representation of the scene after contact. A separately trained inverse policy maps the source representation and the latent goal to a continuous paddle action. We execute that action and measure the outgoing ball velocity. These experiments use the additive writer without magnitude axes.

On the actuated Franka arm, all 200 latent-goal commands finish within 2% of the requested speed (Figure 6b). Panel (c) tests whether the policy actually uses the latent goal. Swapping goals between trials lowers success to 28%, and giving every trial the same constant goal lowers it to 2.5%, so the policy depends on the specific goal it receives. A numeric goal, where the requested speed is supplied directly as a number, also reaches 100%. The latent goal therefore works as an interface for control on this task, and the task itself can also be solved from an explicit, numeric goal.

At a 5% tolerance, the latent-goal policy reaches 100% on an abstract paddle, a kinematic Franka arm, and an actuated Franka arm. A harder two-dimensional version commands the full outgoing velocity vector, and on the actuated Franka arm it averages 80.5% across five seeds of 40 held-out trials (Appendix G). Off-center strikes that command speed and angular velocity together would test this interface under more varied contact conditions.

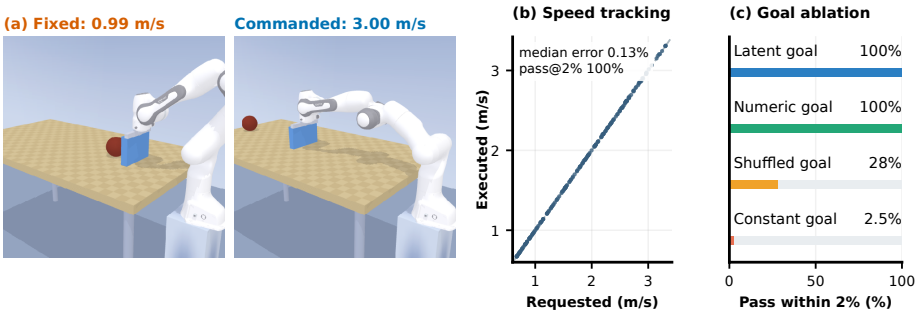


Figure 6: Latent-goal control of a simulated Franka strike. (a) Illustrative snapshots at the same simulation frame, after contact in both rollouts. For a requested outgoing speed of $3.0\,\mathrm{m/s}$, the fixed swing produces $0.99\,\mathrm{m/s}$ and the commanded swing produces $3.00\,\mathrm{m/s}$, measured from simulator state. This example is separate from the benchmark in (b,c). (b) Requested versus executed outgoing ball speed across 200 held-out latent-goal trials on the actuated Franka; the dashed line denotes exact tracking. (c) Fraction of trials with relative speed error at most 2%. Latent and numeric goals each reach 100%; shuffled and constant goals reach 28% and 2.5%, respectively.

7 Conclusion

In this work, we proposed a method for editing the latent states of a frozen video world model to steer the physics it encodes while enforcing strict constraints. For example, to change the velocity of a moving ball from a to b, we require that the decoded video leave the rest of the scene unchanged while the ball moves at velocity b. We first inspect the latent space of 4 physical quantities, linear and angular velocity and acceleration. Using PCA and orientation probes on simulated datasets, we find that translation follows open paths while rotation revisits the same orientations at a rate set by its angular velocity. This structure motivates two writers, an additive writer for translation and an orientation field for rotation. The additive writer raises the correlation between requested and decoded speed from 0.465 to 0.750 in 2D and from 0.733 to 0.861 in rendered 3D, with direction errors below $10°$. The orientation field reached an angular-velocity correlation of 0.942, held it at 0.945 under new appearance, and steered angular acceleration without refitting. Finally, we used an edited representation as the goal for a simulated robot arm, whose strikes matched the requested ball speed to within 2% on all 200 test commands.

Limitations and future work. Our experiments use simulated scenes, decoders trained for each task, and object positions and orientations read from the simulator, which give us exact labels to verify every edit. Natural next steps are to apply the writers to natural video with camera motion and clutter; to estimate the object's position and orientation from the video itself, for example with trained probes, so that the approach extends to harder scenes such as uneven ground or transparent objects; and to test the edits with decoders built independently of our pipeline. Improving the decoders would also help close the remaining gap in the size of the decoded response, especially for acceleration under varied appearance, where the target decode itself sets the ceiling. Carrying the latent-goal interface from simulation to a physical robot is a further direction, which would add sensing noise, contact uncertainty, and latency that our simulated strikes do not include.

8 AI usage statement

We used generative AI tools (Claude) for writing dataset-generation code, debugging parts of the experiment code and plotting figures. We did not use generative AI tools for hypothesis proposal, the conceptual framework (the additive writer and orientation field) or result interpretation. All datasets were rendered by a physics simulator (such as MuJoCo). Proofs and qualitative data analysis do not apply to this work. We also used generative AI tools to edit the text for clarity and to revise figure captions. The authors reviewed and tested all AI-assisted code and checked every AI-edited passage against the experimental results. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

9 REPRODUCIBILITY STATEMENT

All code for this work is available in an anonymized repository at https://anonymous.4open.science/r/vjepa-steering-physics-7521. The repository follows the same order as the paper. It starts with the procedural scene generators for all four environments, then the wrappers that load the frozen V-JEPA 2 encoders, the code that fits the probes, the additive writer, and the orientation field, and the code and configurations that train every decoder in Table 3. It also includes the trackers, the inverse policy, and the evaluation scripts, together with a test suite and a step-by-step protocol for the control experiment.

We do not distribute the datasets. Every clip is a deterministic function of a seed and a scene index, so running the generators reproduces each clip exactly (Appendix B.2). The encoders are the public V-JEPA 2 ViT-L, ViT-H, and ViT-g checkpoints, which the code loads from Hugging Face. Appendices B.1 and B.4 give the evaluation splits, the gain selection, and the definition of every metric we report.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture. In *CVPR*, pp. 15619–15629, 2023. doi: 10.1109/CVPR52729.2023.01499.

Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and Scalable Self-Supervised Learning Without the Heuristics. *arXiv preprint arXiv:2511.08544*, 2025. URL https://arxiv.org/abs/2511.08544.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting Feature Prediction for Learning Visual Representations from Video. *TMLR*, 2024. URL https://openreview.net/forum?id=QaCCuDfBk2.

Hila Chefer, Uriel Singer, Amit Zohar, Yuval Kirstain, Adam Polyak, Yaniv Taigman, Lior Wolf, and Shelly Sheynin. VideoJAM: Joint Appearance-Motion Representations for Enhanced Motion Generation in Video Models. In *ICML*, volume 267, pp. 7595–7616, 2025. URL https://proceedings.mlr.press/v267/chefer25a.html.

Quentin Garrido, Nicolas Ballas, Mahmoud Assran, Adrien Bardes, Laurent Najman, Michael Rabbat, Emmanuel Dupoux, and Yann LeCun. Intuitive Physics Understanding Emerges from Self-Supervised Pretraining on Natural Videos. *arXiv preprint arXiv:2502.11831*, 2025. URL https://arxiv.org/abs/2502.11831.

David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *NeurIPS*, volume 31, pp. 2455–2467, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Control Tasks through World Models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Sonia Joseph, Quentin Garrido, Randall Balestriero, Matthew Kowal, Thomas Fel, Shahab Bakhtiari, Blake Richards, and Michael Rabbat. Interpreting Physics in Video World Models. In *ICML*, 2026. URL https://icml.cc/virtual/2026/poster/63036.

Yann LeCun. A Path Towards Autonomous Machine Intelligence. OpenReview, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf. Version 0.9.2.

Andy Zeyi Liu, Haoran Sun, Lucas Baker, Randall Balestriero, and John Sous. Semigroup-JEPA: Latent Dynamics Consistency for Zero-Shot Physics Generalization. *arXiv preprint arXiv:2609.10464*, 2026. URL https://arxiv.org/abs/2609.10464.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels. *arXiv preprint arXiv:2603.19312*, 2026. URL https://arxiv.org/abs/2603.19312.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do Generative Video Models Understand Physical Principles? In *WACV*, pp. 948–958, 2026. URL https://openaccess.thecvf.com/content/WACV2026/papers/Motamed_Do_Generative_Video_Models_Understand_Physical_Principles_WACV_2026_paper.pdf.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mahmoud Assran, Koustuv Sinha, Michael Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking Dense Features in Video Self-Supervised Learning. *arXiv preprint arXiv:2603.14482*, 2026. URL https://arxiv.org/abs/2603.14482.

Sharon S. Musa, Fereshteh Forghani, Harrish Thasarathan, Sonia Joseph, Matthew Kowal, and Konstantinos G. Derpanis. What, Where, and How: Probing Spatiotemporal Representations in Video Foundation Models. *arXiv preprint arXiv:2609.01551*, 2026. URL https://arxiv.org/abs/2609.01551.

Ying Shen, Jerry Xiong, Tianjiao Yu, and Ismini Lourentzou. PHANTOM: Physics-Infused Video Generation via Joint Modeling of Visual and Latent Physical Dynamics. In *CVPR*, pp. 11185–11194, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Shen_PHANTOM_Physics-Infused_Video_Generation_via_Joint_Modeling_of_Visual_and_CVPR_2026_paper.html.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from Reward-Free Offline Data: A Case for Planning with Latent Dynamics Models. In *NeurIPS*, volume 38, 2025. URL https://neurips.cc/virtual/2025/poster/116649.

Kaizhen Tan, Xin Xu, Siru Tao, Yixiao Li, Hanzhe Hong, Yang Feng, and Heqing Du. What Can Latent World Models Know? Physical Parameter Identifiability in Multimodal Predictive Representations. *arXiv preprint arXiv:2607.27017*, 2026. URL https://arxiv.org/abs/2607.27017.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering Language Models With Activation Engineering. *arXiv preprint arXiv:2308.10248*, 2023. URL https://arxiv.org/abs/2308.10248.

Xingyun Wang, Haomin Zheng, Man Yuan, Leqian Yang, and Ziming Liu. A Chosen Future Can Still Be Rewritten: Causal Writability in Video Models. *arXiv preprint arXiv:2609.15980*, 2026. URL https://arxiv.org/abs/2609.15980.

Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. MotionCtrl: A Unified and Flexible Motion Controller for Video Generation. In *SIGGRAPH*, pp. 1–11, 2024. doi: 10.1145/3641519.3657518.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video Models Are Zero-Shot Learners and Reasoners. *arXiv preprint arXiv:2509.20328*, 2025. URL https://arxiv.org/abs/2509.20328.

Haotian Xue, Yipu Chen, Liqian Ma, Zelin Zhao, Lama Moukheiber, Yuchen Zhu, and Yongxin Chen. ACWM-Phys: Investigating Generalized Physical Interaction in Action-Conditioned Video World Models. *arXiv preprint arXiv:2605.08567*, 2026. URL https://arxiv.org/abs/2605.08567.

Xi Zeng, Haojie Ren, and Ziying Song. PhyLatent: Learning Dynamics-Relevant Representations for JEPA World Models. *arXiv preprint arXiv:2608.05720*, 2026. URL https://arxiv.org/abs/2608.05720.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World Models on Pre-Trained Visual Features Enable Zero-Shot Planning. In *ICML*, volume 267, pp. 79115–79135, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation Engineering: A Top-Down Approach to AI Transparency. *arXiv preprint arXiv:2310.01405*, 2023. URL https://arxiv.org/abs/2310.01405.

A Further related work

Joint-embedding predictive architectures (JEPAs). JEPAs learn by predicting the representation of unseen content from the content they observe. I-JEPA applies this idea to images (Assran et al., 2023), V-JEPA applies it to video (Bardes et al., 2024), and V-JEPA 2 extends it to video understanding, prediction, and action-conditioned planning (Assran et al., 2025). V-JEPA 2.1 learns richer dense features by adding supervision across locations and layers (Mur-Labadia et al., 2026). LeJEPA studies representation learning with isotropic Gaussian regularization, and LeWorldModel combines that regularization with latent dynamics prediction to learn end to end from pixels (Balestriero & LeCun, 2025; Maes et al., 2026). In this work we take a pretrained V-JEPA 2 encoder and learn writers for its representation without updating its weights. We call the resulting change to an observed clip counterfactual editing, since a fixed decoder renders the same scene with the requested motion.

Physics-aware latent dynamics. Semigroup-JEPA trains an encoder together with a physics-conditioned predictor through multi-step latent rollouts, and tests prediction and control under gravitational fields it never saw in training (Liu et al., 2026). PhyLatent adds objectives that preserve physical state and separate the consequences of different actions (Zeng et al., 2026). Both methods shape the representation while it is being trained. Our writers instead leave the encoder fixed.

Identifiability of physical parameters. Controlled identifiability studies show how the inputs and the prediction objective determine which physical parameters a video model ends up encoding (Tan et al., 2026). These results concern what the representation contains, which is the starting point for the edits we study.

Probing and steering physical variables. Features from video models pass violation-of-expectation tests of intuitive physics (Garrido et al., 2025), and activation steering changes a model's behavior by intervening on its internal features (Turner et al., 2023; Zou et al., 2023). Joseph et al. (2026) recover physical variables with several probes, steer the representation along the probe directions, and evaluate each edit with a separate held-out probe. Wang et al. (2026) study when corrective motion information can change a generated future. Our writers act on a frozen encoder representation from a physical command, and we evaluate every edit by the motion it produces in the decoded video. Our orientation field, fitted only on constant angular velocity, also edits constant angular acceleration.

Controlled generation and physics benchmarks. MotionCtrl conditions video generation on camera and object motion (Wang et al., 2024), VideoJAM learns joint appearance and motion representations (Chefer et al., 2025), and PHANTOM couples a video generator to a predictor of V-JEPA 2 features (Shen et al., 2026). Physics benchmarks such as Physics-IQ and ACWM-Phys ask whether generated motion is physically correct and not only whether it looks acceptable (Motamed et al., 2026; Xue et al., 2026). Our decoded-motion evaluation applies the same principle to edited representations.

Planning with latent dynamics. DINO-WM and planning methods built on latent dynamics models choose actions by optimizing through learned latent transitions (Zhou et al., 2025; Sobal et al., 2025). In our simulated control experiment we specify the goal by editing a representation and leave the choice of action to an inverse policy, that is, a goal-conditioned policy that maps a source representation and a latent goal to an action.

B Experimental protocol

B.1 Datasets and evaluation splits

The clips within a scene share the same scene content and initial conditions and differ in a single motion command. For acceleration we also hold the initial velocity fixed within a scene, and for angular acceleration we hold the initial angular velocity fixed, so the commanded quantity is the only thing that varies. The 2D simulations use moving balls or marked bars. The 3D simulations render ball motion in MuJoCo, and we measure that motion in the image plane. Table 2 lists, for each experiment, the data we used for fitting and the unit over which we report results. Counts from different rows should not be added together, because several source and target pairs can come from the same scene and are therefore not independent.

Table 2: Data used by the appendix experiments. A scene contains eight motion settings unless stated otherwise. Validation and test scenes are disjoint. Decoder details and writer versions are given with each result.

Experiment	Fitting / calibration	Reported evaluation
Edit parameterization comparison	Four folds; 45 training scenes per fold	60 scenes; 3,360 held-out ordered pairs
Position and orientation readouts	Seven settings of scene 0 per fold	One held-out setting; eight folds
Translation, each dataset	500 writer-training scenes; 50 validation scenes	50 test scenes; one command per scene
Rotation and angular-acceleration transfer	125 field-training scenes; 15 validation scenes	15 test scenes / 105 pairs; target decode uses all 30 evaluation scenes / 210 pairs
Temporal acceleration baselines	50 validation scenes / 350 pairs	50 test scenes / 350 pairs
Decoder-guided search	Optimize each evaluated scene	30 held-out scenes per quantity
Velocity composition	Additive writer without magnitude axes; fixed gain 2	40 held-out 3D scenes
Simulated control	Inverse policy trained on separate scenes	Scalar task 800/800/200 trials; vector task 40 trials per seed and embodiment

B.2 Dataset curation

All datasets share one clip format and one set of encoder settings, so any difference between steering results comes from the scene itself. Clips have 16 frames at 256×256 resolution and 4 frames per second. The encoder is the frozen V-JEPA 2 ViT-L checkpoint pretrained on 64-frame clips at resolution 256, and we cache layers $\{6, 12, 18, 23\}$. With a tubelet size of 2 and a patch size of 16, each layer gives $H \in \mathbb{R}^{8\times256\times1024}$.

Scenes and nuisance variables. A scene contains eight clips that share a start position and appearance, and the commanded quantity takes a different value in each clip. Because position and appearance are shared, the true edit $H_b - H_a$ between two clips of a scene isolates the change in the commanded quantity, and the nuisance variables cancel out. We also decorrelate the nuisance variables explicitly. The eight headings are spaced evenly around the circle from a random offset, and each is jittered by up to 0.2 rad. The eight speeds are drawn uniformly from the speed range, pushed apart to a minimum gap of 5% of that range, and paired with the headings by a random permutation, so that speed carries no information about heading. Acceleration scenes vary the acceleration vector in the same way around an initial velocity that the scene shares, and rotation scenes permute signed

angular velocities with equal numbers of each sign. Mixed-appearance datasets randomize the ball's color and radius and the background's gray level across scenes and hold them fixed within a scene.

Shared start positions. One start position has to keep all eight trajectories of a scene inside the frame. We compute the set of valid start positions for each trajectory and intersect the eight sets. If the intersection is empty, we scale all eight speeds together by a factor of 0.85 and try again, for up to 24 attempts, which preserves the headings and the relative speeds. If every attempt fails, the scene falls back to a centered start at the smallest scale we tried, which leaves the ball nearly static. We chose the speed range of the rendered 3D scene so that no scaling is ever needed.

Splits and reproducibility. Training and evaluation scenes come from different random seeds and never overlap. The training seed yields 500 scenes (4,000 clips), and the evaluation seed yields 100 scenes (800 clips), which we split into validation and test halves. Each clip is a deterministic function of its scene index, so regenerating an index reproduces the clip exactly. We therefore store no clips and regenerate them on demand.

Labels. Labels live in the image plane, in normalized coordinates from 0 to 1 with y pointing down, including for the MuJoCo scenes. The velocity label is the clip's mean displacement between frames, so it is the same quantity the tracker recovers from decoded pixels. We store world-frame quantities as metadata only, because steering a world-frame velocity while measuring with an image-plane tracker would add a projective bias to every direction error. The rotating bar is drawn directly in image coordinates without a camera model, so its angular velocity label is already an image-plane rate.

Measured quantities. We measure each quantity the way its generator defines it. Velocity is the displacement of the object's centroid. Acceleration and gravity come from a parabola fitted to the decoded centroid track, with acceleration equal to twice the quadratic coefficient. We chose their ranges so that speed changes by about a factor of three over 16 frames, which makes the curvature visible while keeping the ball in the frame. Angular velocity uses a rigid bar with a red marker at one end, since the rotation of a uniform disk is invisible, and we cap it at 0.20 rad/frame so that unwrapping the tracked phase is unambiguous.

The rendered 3D scene. The 3D dataset changes only the renderer. A MuJoCo sphere rolls without slipping on a textured table under a fixed perspective camera. Its scenes are constructed exactly as in 2D, and its decoder configuration is identical apart from the training data. The scene therefore tests whether our results reflect a property of V-JEPA 2 or an artifact of the simple 2D renderer. We set three scene parameters in response to failures we measured. At a camera elevation of 32°, foreshortening changed image speed by a factor of 5.6 across headings and collapsed the eight commands onto an ellipse. At 62°, this ratio falls to 1.267. A hard contact shadow pulled the tracked centroid 17 pixels off the ball, so we use ambient-dominant lighting, under which the tracker agrees with the analytic motion to within 0.22 pixels and 0.15°. The speed range of 0.06 to 0.115 m/s is the widest band for which no scene needs its speeds scaled to share a start position (0 of 150 scenes, against 75% of scenes for 0.08 to 0.15 m/s). It also maps onto the image-speed band of the 2D dataset, so the two datasets can be compared directly.

Automated checks. Five checks run automatically with every code change and are enforced by 12 unit tests. They compare the tracker with the labels, confirm that velocity is constant within a clip, measure perspective anisotropy, compare the 3D image-speed band with the 2D reference, and confirm that every scene has a shared start position.

B.3 Frozen encoders, decoders, and trackers

The default encoder is the frozen V-JEPA 2 ViT-L (24 blocks, width 1024). We record its representation at layers 6, 12, 18, and 23. A clip of 16 frames at 256×256 gives eight temporal tokens on a 16×16 spatial grid. We fit a separate writer at each of the four layers and apply their edits together. The decoded ViT-H runs use layers $\{8, 16, 24, 31\}$ and ViT-g uses $\{10, 20, 30, 38\}$. Both use encoder-specific decoders, and Appendix F.3 reports their results separately.

Each steering decoder has 209M parameters, reads layers 6, 12, 18, and 23, and predicts each frame through a patch-level output head. We train it separately from the writers and keep it frozen during all steering experiments. Its training loss combines ten terms, namely a robust pixel loss, a foreground loss, SSIM, MS-SSIM, temporal consistency, state, trajectory, and velocity losses, a per-frame centroid loss, and a spread penalty. A pixel loss on its own can be minimized by a blurred, nearly static reconstruction, and the per-frame centroid term, which carries the largest weight, rules that out by requiring the object's position to be correct in every frame. Each dataset has its own decoder, and the rotation decoder adds a per-frame orientation loss. A separate 296M decoder with a convolutional upsampling head renders the filmstrips and videos in our figures, and no reported number comes from it. Within each dataset we use the same checkpoint for every variant, with one exception. For rendered 3D velocity, the raw-feature variant uses the original decoder checkpoint, and the two standardized variants use a second checkpoint of the same decoder trained with a different random seed. The comparison that adds magnitude axes therefore shares one checkpoint, while the comparison between raw and standardized features also changes the decoder. Table 3 lists the checkpoint behind each result, and Appendix B.6 shows that the two checkpoints preserve the ranking of methods and give statistically indistinguishable errors for the raw-feature writer.

All of the main rotation experiments use one decoder, the orientation-supervised `angvel2d_big` ViT-L decoder at training step 7,000. This includes the mixed-appearance and angular-acceleration transfer experiments. The bar carries a marker at one end, which breaks its half-turn symmetry and makes orientation identifiable over a full turn. We measure translation by tracking the object's center in each frame and fitting its motion across frames. We measure rotation from the bar and its marker, then unwrap the resulting angle over time. We first validate each tracker on simulator frames, where the true motion is known, and its error there is below 0.2%. We then apply it to decoded videos. Finally, the target decode checks that the decoder itself can render the requested motion on the same task.

Table 3: Decoder checkpoints. Runs load the saved `last.pt` checkpoint unless a training step is given.

Use	Checkpoint
2D velocity, single appearance	`moving_ball_scene_v2d_decoder_fp`
2D velocity, mixed appearance	`moving_ball_scene_v2d_mixed_decoder_fp`
2D acceleration, single appearance	`moving_ball_scene_accel2d_decoder_fp`
2D acceleration, mixed appearance	`moving_ball_scene_accel2d_mixed_decoder_fp`
3D velocity, raw-feature variant	`rolling_ball3d_decoder_fp`
3D velocity, standardized variants	`rolling_ball3d_decoder_fp_seed1`
3D acceleration	`size_sweep_decoded/rb3d_accel/vjepa2_large`
Rotation, step 7,000	`moving_ball_scene_angvel2d_big_decoder_orient`
Figures only (296M)	`hifi_rb3d_decoder`
Additive writer implementation	`cmd_U8` (historical name)

B.4 Metrics and calibration

For a commanded vector q_i and a tracked vector $\hat{q}_i$, direction error is the mean of $\arccos(q_i^\top \hat{q}_i/(\|q_i\|\|\hat{q}_i\|))$, in degrees. Magnitude correlation is the Pearson correlation between $\|q_i\|$ and $\|\hat{q}_i\|$. We also fit $\|\hat{q}_i\| = b\|q_i\| + c$ with an intercept, and the slope b measures the scale of the response, which correlation alone does not. Translation units are image widths per frame for velocity and image widths per frame squared for acceleration, including in rendered 3D scenes. For rotation, correlation, slope, and mean absolute error use the signed angular velocity or acceleration, in radians per frame or per frame squared. Sign accuracy measures whether the decoded motion has the requested sign.

We select translation gains by minimum mean direction error on the first 50 sorted evaluation scenes and fix them for the remaining 50. The candidate grid is $\{1, 1.5, 2, 2.5, 3\}$. This selects for direction and not for magnitude correlation. Table 7 reports the selected gain for every variant. Rotation gains minimize the mean squared error of the signed outcome on 15 validation scenes, with 15 different scenes held out. Angular-acceleration transfer reuses the constant angular velocity field

without refitting it, but calibrates its gain on angular-acceleration validation scenes. Transfer therefore concerns the learned field and not the absence of calibration on the target task.

All translation references below are recomputed on the same 50 test scenes as their writer. The saved rotational target decode pools 210 pairs from both validation and test scenes, whereas writer scores use 105 test pairs. We label this difference instead of treating those pools as a paired test-only comparison. Random projections use the target representation. Rotational shuffled-command baselines instead permute the command labels while holding the decoded outcomes fixed. Neither is an alternative writer that works from the command alone.

B.5 Probe editing and the layer where the edit is applied

Editing the representation along the direction of a velocity probe changes the probe's prediction as intended, and the predicted velocity increases monotonically with the strength of the edit ($\rho = 1.000$). The decoded speed, however, stays nearly fixed until the edits become large enough to damage the reconstruction. This diagnostic uses a single probe and does not reproduce the multi-probe steering method of Joseph et al. (2026). A separate experiment on 100 rendered-3D scenes asks where the decoder is sensitive to edits. It substitutes the true target representation at one decoder input layer at a time and leaves the other three layers unchanged (Table 4). Layers 6 and 12 support accurate velocity readout ($R^2 \geq 0.991$), yet an edit applied there barely changes the decoded motion, while the same edit at layer 23 produces most of the response. Accurate readout at a layer therefore does not mean the decoder responds to an edit at that layer. The writers in the main results edit all four input layers.

Table 4: Readout and single-layer edits in the rendered-3D velocity experiment. The frozen ViT-L `rolling_ball3d_decoder_fp` decoder receives the true target representation at only the indicated layer, and the other inputs stay unchanged. Scores use 100 held-out scenes, not the 50-scene writer split. The error with no edit is $44.2°$. The last column is the decoded response slope.

Layer	Probe R^2	Direction error (°)	Response slope
6	0.991	44.0	0.028
12	0.997	43.9	0.028
18	0.997	38.0	0.162
23	0.998	7.1	0.822

B.6 Results across two decoder checkpoints

The ranking of methods does not depend on the decoder. Figure 7 scores the same latent edits through the two rendered 3D velocity decoders of Table 3, which share their architecture and training clips and differ only in the random seed used for training. We apply the edits at all four layers and measure direction error on the ball's image-plane velocity tracked in the decoded frames. The writer here is the additive writer with raw features and a rank-8 PCA basis, which is the version that steers direction but not speed (Figure 12). Its gain of 2 was selected with decoder A on 50 of the 100 evaluation scenes and reused for decoder B, and we report it on the other 50. The remaining methods use no gain and are scored on all 100 scenes.

Every method keeps its place under both decoders. Projecting the target edit onto a random eight-dimensional subspace gives about 44°, close to the 44.2° of no edit. A single global edit, meaning one linear map from the velocity change to a fixed latent pattern that ignores the scene, gives about 34°. Projecting the target edit onto the top eight principal directions gives 15.4° and 13.2°, the additive writer 11.4° and 11.7°, and the target decode 2.7° and 2.1°. A paired bootstrap over scenes finds no significant difference between the decoders for the additive writer ($p = 0.13$) or the global edit ($p = 0.12$). The decoders do differ by up to 2.2° on the two projections and the target decode ($p < 0.02$), which is small next to the gaps between methods.

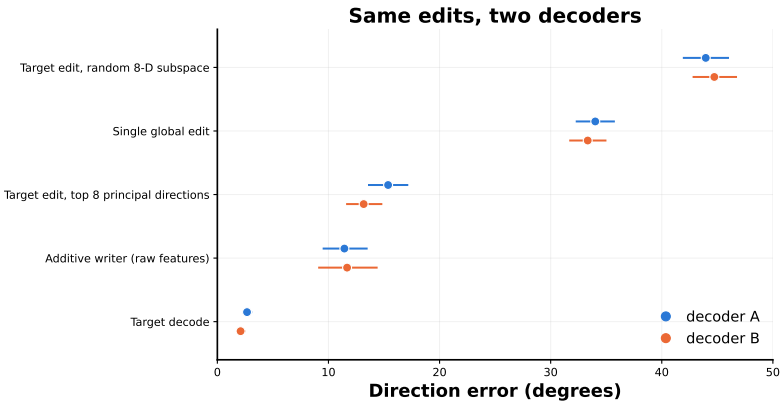


Figure 7: Direction error for the same latent edits decoded by two rendered 3D velocity decoders that differ only in training seed. The additive writer (raw features, rank-8 basis, gain 2) is scored on the 50 evaluation scenes not used for gain selection, and every other method on all 100 evaluation scenes. Both projections receive the target representation. Bars are 95% scene-level bootstrap intervals.

C Geometry and edit parameterization

C.1 Testing the edit parameterization

Section 5.3 compared the additive writer with a conditioned writer, whose edit is a learned function of position or orientation evaluated at the target configuration minus its value at the source configuration. Table 5 gives the layer-12 scores, and Appendix C.3 gives the fitting protocol and the alignment score in Equation 5.

Motion Type	Configuration	Probe R^2	Additive	Conditioned	Cond./Add.
Angular velocity	S^1, closed	.925	.106	**.423**	**4.01**
Angular acceleration	S^1, closed	.783	.037	**.411**	**11.24**
Speed	line, open	.942	**.165**	.076	.46
Acceleration, 1D	line, open	.939	**.185**	.061	.33
Velocity, 2D	$\mathbb{R}^2$, open	.995	**.538**	.466	.86
Acceleration, 2D	$\mathbb{R}^2$, open	.918	**.303**	.080	.26

Table 5: Additive and conditioned writers across six motion types at layer 12. Scores are norm-weighted cosines between predicted and true edits over four scene-level folds (60 scenes and 3,360 held-out pairs per type). Values above one in the last column favor the conditioned writer.

The conditioned writer improves angular-velocity alignment from 0.106 to 0.423 and angular-acceleration alignment from 0.037 to 0.411. The additive writer performs better on all four translational motion types, and this ordering also holds at layer 23 (Figure 11). Latent distance computed over all tokens rises and then falls as the difference in orientation increases, consistent with a recurring orientation path. The translational motion types show no comparable rise and fall (Figure 10).

C.2 Latent trajectories across encoder sizes

Figures 8 and 9 show the PCA views for ViT-H and ViT-g that correspond to Figure 3 for ViT-L. Each view uses scene 0, with eight motion settings and eight temporal tokens per clip, which gives 64 points. We keep the 16 spatial tokens that carry the strongest motion signal, subtract the mean across settings at each time so that the time course shared by all settings is removed, apply a single scalar normalization, and fit a separate three-component PCA for each motion type. We chose the layers at similar relative depth, namely layer 12 of 23 for ViT-L, 16 of 31 for ViT-H, and 20 of 39 for ViT-g. These views are descriptive. The tests below evaluate the edit parameterization in the full representation space, without truncating to three principal components.

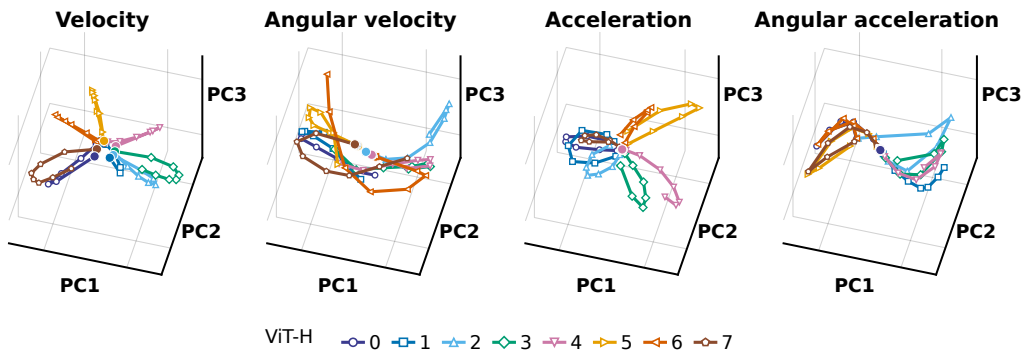


Figure 8: ViT-H latent trajectories at layer 16. The scene, settings, preprocessing, and panel arrangement match Figure 3. Lines follow temporal tokens and filled circles mark their starts. Rotation traces recurring orientations, while translation follows position changes.

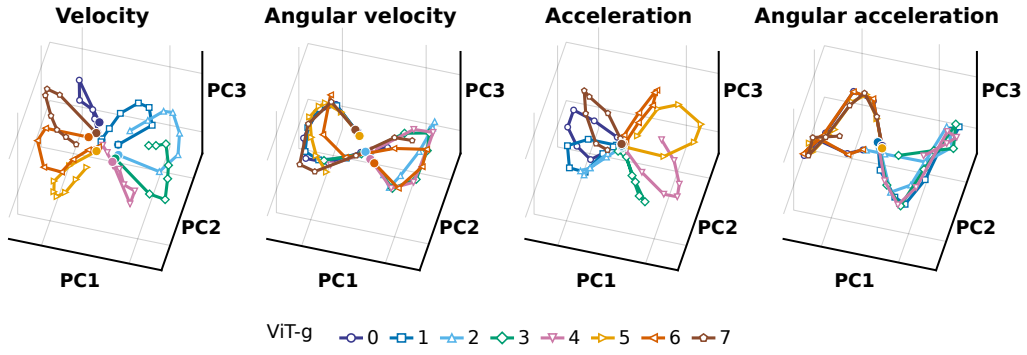


Figure 9: ViT-g latent trajectories at layer 20, using the same scene, motion settings, and preprocessing as the ViT-L/H views. The quantitative comparison of edit parameterizations uses the full representation rather than these three PCs.

C.3 FULL-SPACE MEASUREMENTS AND WRITER COMPARISON

The comparison across six motion types uses 60 scenes with eight motion settings each, and the ViT-L representation at layers 12 and 23. Every scene contributes 64 points, one for each of its eight clips at each of eight temporal tokens. We work in an orthonormal basis for the subspace spanned by these 64 points, which preserves all distances and angles among them exactly. The distance measurement uses all tokens of the representation. The edit-agreement measurement and the writer fits use the 24 spatial tokens with the strongest motion energy, after shifting the object's center (or its starting token) to the center of the grid.

Figure 10 shows the two measurements separately. The distance measurement asks how far apart two representations are as a function of how far apart the object's configurations are. We compare every pair of points within a scene, divide each latent distance by that scene's mean pairwise latent distance, and plot the median in each bin. For rotation, the horizontal axis is the total difference in orientation, without folding it into $[0, \pi]$. The agreement measurement asks whether the same command produces the same latent edit at different configurations. We compare pairs of edits whose commands point in nearly the same direction (cosine above 0.9) and have similar size (magnitude ratio between 0.8 and 1.25), and we plot the mean cosine between the two edits in each bin. Here the horizontal axis is the separation between the configurations at the midpoints of the two edits, and rotational separation is wrapped. The two measurements therefore differ in the tokens they use and in how they summarize each bin, as the figure labels.

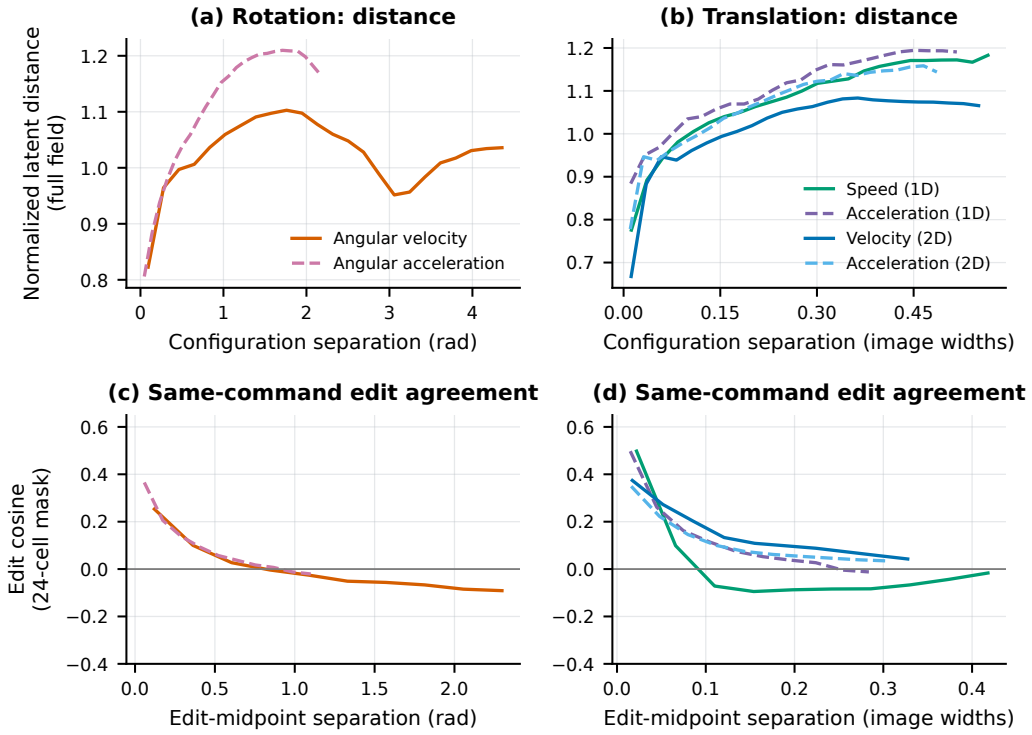


Figure 10: Geometry beyond a three-PC view (ViT-L, layer 12, 60 scenes per motion type). The top row shows latent distance over all tokens, normalized within each scene. The bottom row shows agreement between similar-command edits in the recentered 24-token mask. Native physical units are retained. Rotational distance rises and then falls as orientation changes, while translational distance rises and then often flattens. Agreement weakens in both groups, so the curves alone do not establish which writer is better. Figure 11 tests that choice.

We compare an additive writer with a conditioned writer. For rotation the conditioned writer uses harmonic order 4 in orientation, and for translation it uses separable harmonic order 8 in position. Four scene-level folds fit on 45 scenes and evaluate all ordered pairs in the other 15, which gives 3,360 held-out pairs across folds. The score weights the cosine of each pair and token by the norm of

its true edit,

$$A = \frac{\sum_{p,t} \|\Delta H^{\star}_{pt}\| \cos(\widehat{\Delta H}_{pt}, \Delta H^{\star}_{pt})}{\sum_{p,t} \|\Delta H^{\star}_{pt}\|}. \tag{5}$$

Figure 11 repeats the layer-12 comparison from Table 5 at layer 23. At both depths the conditioned writer is better for the two rotational motion types, and the additive writer is better for all four translational motion types. Absolute alignments remain moderate even with the better parameterization, so neither writer recovers the entire latent change. This test scores predicted latent edits directly and involves no decoder. It is separate from the decoded steering experiments in Section 6.

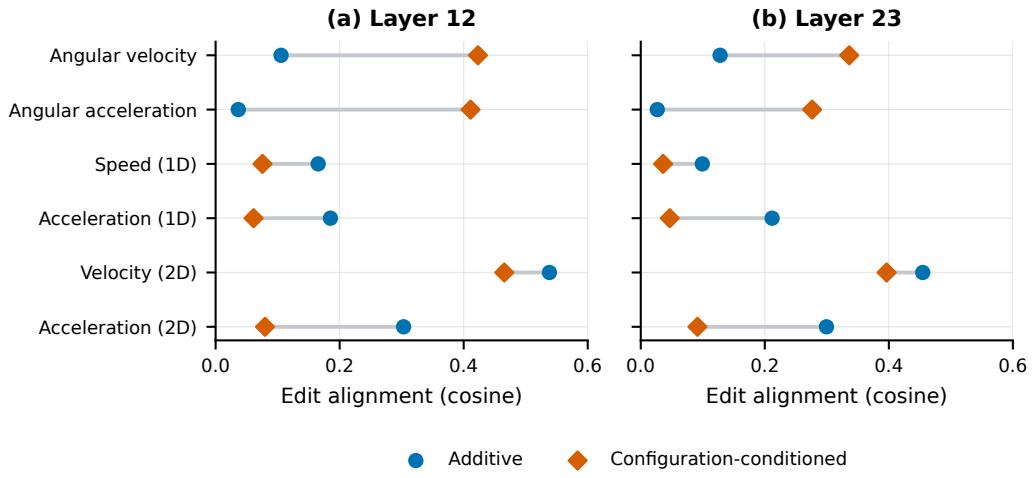


Figure 11: Edit parameterization at two ViT-L depths. Each point is a norm-weighted edit cosine over 3,360 held-out pairs, and segments connect the two writers for the same motion type. The preference for the conditioned writer on rotation and the additive writer on translation persists at layer 23. No decoder is used.

C.4 Held-out position and orientation readouts

The readouts in Figure 4 are fitted on the full coordinates of the 64 points, in the lossless basis described above. We use the three-component PCA for visualization alone. A ridge readout trains on seven motion settings (56 points), predicts the eighth (eight points), and the procedure repeats until every setting has been held out once. We compute centering and scalar normalization from the training settings alone, and we fix the ridge penalty at 0.01. The readout receives neither time nor the motion command as input. The one step that sees the held-out setting is token selection, which uses all eight settings in the scene without their labels, so that preparation is transductive.

Translation targets are the x, y position. Rotation targets are $\cos(k\theta)$ and $\sin(k\theta)$ for $k = 1, \ldots, 4$. The main figure displays the second harmonic and unwraps $\operatorname{atan2}(\widehat{\sin 2\theta}, \widehat{\cos 2\theta})$ over time. Because this is the second harmonic, the plotted phase changes by twice the physical change in orientation. We subtract the shared time course and each curve's initial offset after prediction, purely for display, and we fit no line or parabola to the traces. Table 6 scores the predictions before this display centering, with R^2 measured relative to the mean training target at each temporal token. These tests hold out motion settings within a single scene. They do not test transfer across scenes, and they do not provide independent evidence of circular topology.

Table 6: Readout R^2 for position and orientation at corresponding relative depths, pooled over eight held-out-setting folds of scene 0. Translation columns score x, y and angular columns score the second-harmonic cosine and sine. No decoder is used.

Encoder (layer)	Velocity	Angular velocity	Acceleration	Angular acceleration
ViT-L (12)	0.518	0.982	0.612	0.933
ViT-H (16)	0.352	0.997	0.443	0.905
ViT-g (20)	0.457	0.990	0.568	0.951

D VELOCITY AND ACCELERATION STEERING

D.1 WRITER CONSTRUCTION AND MAGNITUDE AXES

For each of the 500 training scenes, we pair one source clip with the other seven settings, which gives 3,500 true edits $d_i^\star = \mathrm{vec}(H_{b,i} - H_{a,i})$. At each layer, a PCA basis U_k supplies the targets $U_k^\top d_i^\star$ in that basis. The command map uses the 13 features

$$\phi_{\mathrm{raw}}(q_a, q_b) = [1, q_b, q_a, q_b - q_a, \|q_b\|, \|q_a\|, u_b, u_a], \qquad u = q/\|q\|,$$

where q is a two-component velocity or acceleration vector. Ridge regression with penalty 1 predicts the basis coefficients. The standardized fit divides each feature column by its root-mean-square value over the training set. The columns are scaled and left uncentered. After fitting, we map the coefficients back to the units of the raw features, so the writer takes unscaled commands as input. The scaling matters because ridge regression penalizes large coefficients. A feature with small numerical values, such as a speed measured in image widths per frame, needs a large coefficient to influence the prediction, and the penalty would suppress it for that reason alone. Scaling every column to unit root-mean-square value removes this dependence on units.

To add magnitude axes, a second standardized ridge fit predicts the full true edit from $[1, \Delta s, \Delta s\, u_{b,x}, \Delta s\, u_{b,y}]$, where $\Delta s = \|q_b\| - \|q_a\|$. We orthogonalize the three nonconstant coefficient vectors against U_k and against previously accepted axes, then normalize and append them. After the three axes are appended, we refit the command map W in the enlarged basis, using the same features and the same ridge penalty. We keep all three axes in every reported dataset. The PCA basis has rank 8 for single-appearance 2D velocity and for all acceleration datasets, and rank 16 for mixed-appearance 2D velocity and 3D velocity, so the enlarged ranks are 11 and 19 (Table 7). The implementation name in Table 3 is historical and does not indicate the rank actually used.

The final writer receives the source representation, the source physical value, and the requested value, but not a target clip. Held-out commands are drawn continuously from the same generator and command range as training. These experiments therefore test control in new scenes with commands inside the training range. We leave extrapolation to commands outside that range untested.

D.2 FEATURE SCALING AND MAGNITUDE AXES

Table 7 separates the effect of feature scaling from the effect of adding magnitude axes. It compares three variants, a writer with raw features, a writer with standardized features, and a writer with standardized features plus the three magnitude axes. Each variant selects its own gain on the same validation scenes. The second and third variants select the same gain in all six datasets, so the difference between them comes from the basis alone. Within a dataset, every score and every target decode uses the same 50 test scenes and the same frozen ViT-L decoder trained for that dataset, apart from the rendered 3D velocity exception in Appendix B.3.

Adding the magnitude axes raises the velocity magnitude correlation to 0.750, 0.546, and 0.861 for single 2D, mixed 2D, and 3D. Figure 13 shows both the correlation and the scale of the response. Mixed-appearance velocity still realizes only part of the command, with a response slope of 0.354. For acceleration, the correlations are 0.479, 0.205, and 0.758, with slopes of 0.438, 0.226, and 0.735. Mixed-appearance acceleration barely improves, and the target decode on the same test scenes reaches only 0.580. The decoder therefore limits this result as well as the writer, and the weak mixed-acceleration number should not be read as evidence that the encoder lacks acceleration information.

Table 7: Translation ablations on 50 test scenes per dataset. Each result is direction error (°) / magnitude correlation, and g is selected on 50 separate validation scenes. All variants are additive writers with the same command features. "+ axes" is the final variant with magnitude axes, for velocity and acceleration alike. The rank column gives the PCA rank → the rank with magnitude axes.

			Raw features		Standardized		Standardized + axes	
Quantity	Dataset	Rank	g	Error / r	g	Error / r	g	Error / r
Velocity	2D single	$8 \to 11$	2	5.55 / 0.117	2	5.76 / 0.465	2	6.45 / 0.750
Velocity	2D mixed	$16 \to 19$	2.5	5.69 / -0.085	2.5	6.43 / 0.330	2.5	7.09 / 0.546
Velocity	Rendered 3D	$16 \to 19$	1.5	9.67 / 0.214	1.5	8.55 / 0.733	1.5	9.28 / 0.861
Acceleration	2D single	$8 \to 11$	2	11.21 / 0.025	2	10.16 / 0.359	2	10.04 / 0.479
Acceleration	2D mixed	$8 \to 11$	1.5	19.72 / 0.064	2	15.48 / 0.201	2	14.51 / 0.205
Acceleration	Rendered 3D	$8 \to 11$	1.5	13.77 / 0.244	1.5	10.29 / 0.671	1.5	10.67 / 0.758

Table 8: Projection baselines and target decodes, recomputed on the same 50 test scenes as Table 7. Cells are direction error (°) / magnitude correlation. Every column uses the target representation at unit gain. Random projection has the base rank k, and the PCA projections with and without magnitude axes have ranks $k+3$ and k. They test what each basis retains and not command prediction. The target decode uses the full target representation.

Quantity	Dataset	Random	PCA	PCA + axes	Target decode
Velocity	2D single	45.08 / 0.081	20.39 / 0.515	20.27 / 0.697	1.91 / 0.975
Velocity	2D mixed	45.93 / 0.110	21.90 / 0.278	21.65 / 0.337	4.17 / 0.837
Velocity	Rendered 3D	43.99 / 0.193	11.36 / 0.742	10.85 / 0.889	1.93 / 0.980
Acceleration	2D single	47.24 / 0.072	24.20 / 0.172	23.76 / 0.233	5.98 / 0.830
Acceleration	2D mixed	49.26 / 0.030	25.27 / 0.170	24.71 / 0.183	10.05 / 0.580
Acceleration	Rendered 3D	44.91 / 0.065	12.67 / 0.398	12.32 / 0.565	9.27 / 0.886

Where the magnitude is lost. Figure 12 follows the speed response through the three variants of Table 7. With raw features the writer barely changes speed, and most of its intervals include zero. Standardizing the features before the ridge fit recovers most of the response, and adding the magnitude axes raises the speed correlation to 0.75, 0.55, and 0.86 for velocity and 0.48, 0.21, and 0.76 for acceleration in single-appearance 2D, mixed-appearance 2D, and rendered 3D. The speed information was therefore present in the representation, and the raw fit simply failed to use it. The magnitude axes cost less than one degree of direction accuracy for velocity, and for mixed-appearance 2D acceleration the final writer is about 5° more accurate than the raw fit. The remaining gap comes from the basis. When we project the true edit onto the first eight principal directions, the decoded speed correlation stays low, and adding the three magnitude axes raises it, for example from 0.60 to 0.88 for rendered 3D velocity. Mixed-appearance 2D acceleration is the weak case. Its correlation moves only from 0.20 to 0.21, and the target decode itself reaches only 0.58.

D.3 Temporal baselines for acceleration

This experiment asks whether the temporal structure of an acceleration edit matters. It uses an earlier version of the acceleration writer, without magnitude axes, which assigns a separate coefficient to each of the eight temporal tokens, together with the mixed-appearance 2D ViT-L acceleration decoder. The 100 evaluation scenes each contain seven pairs that go from a source clip to a target clip. We select gains on 50 scenes (350 pairs) and report results on the other 50 (350 pairs). Table 9 reports the baselines. The writer with eight temporal coefficients reaches a direction error of $13.97°$. Replacing them with one coefficient shared across time raises the error to $24.44°$, and shuffling their temporal order raises it to $44.32°$, even though each baseline selects its own gain on the validation scenes. The timing of the edit therefore carries information that the decoder uses. These baselines apply to the earlier writer, and we did not repeat them for the final writer with magnitude axes. The table also includes a profile-only baseline, which averages the true edit over spatial tokens and broadcasts the resulting temporal profile back across the grid. Its error of $18.06°$ exceeds the $11.24°$ of the full target decode, which shows that the spatially averaged profile discards part of what the decoder uses.

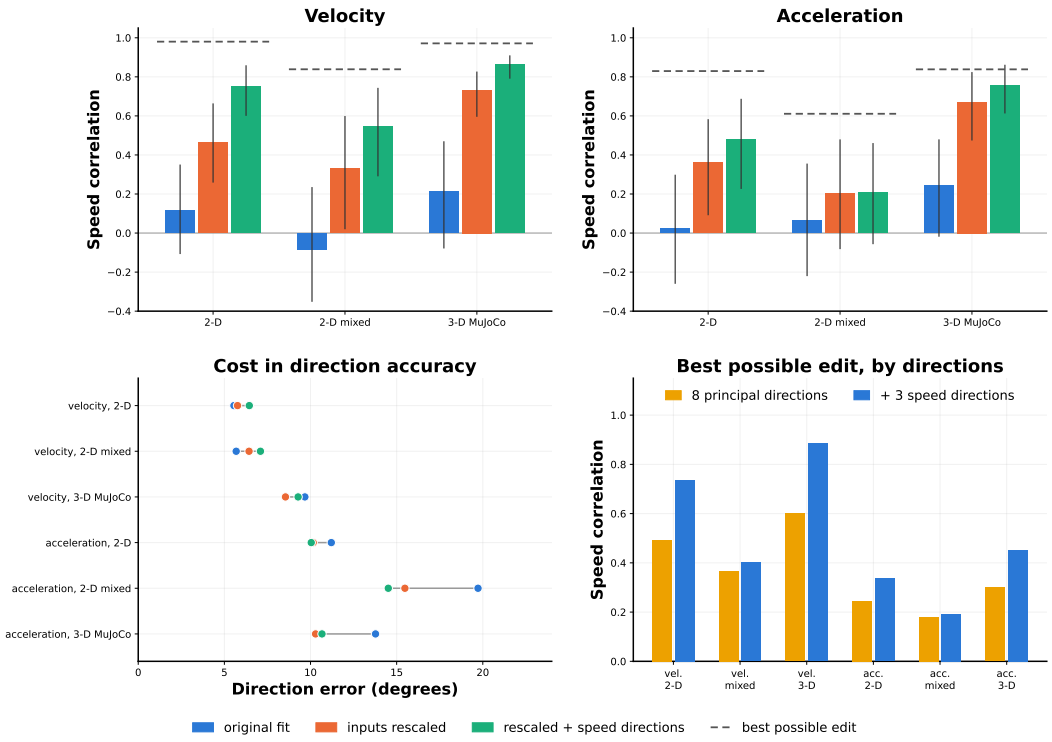


Figure 12: Recovering the commanded magnitude. The velocity and acceleration panels show speed correlation for the three writer variants, with the target decode for reference. The direction panel shows what each variant costs in direction error. The last panel projects the true edit onto the first eight principal directions, with and without the three magnitude axes. It uses eight directions for every dataset, so its values differ from Table 8, which uses 16 for mixed-appearance 2D velocity and rendered 3D velocity. Bars are 95% scene-level bootstrap intervals over the 50 test scenes.

Table 9: Temporal acceleration baselines on 350 held-out pairs from 50 mixed-appearance 2D scenes, using the frozen ViT-L acceleration decoder. Each writer's gain is selected on the other 350 pairs. The target decode is decoded at unit gain.

Writer or baseline	Gain	Direction error (°)
Eight-token temporal writer	2.5	13.97
True temporal profile, broadcast spatially (oracle)	2.5	18.06
One global temporal coefficient	6	24.44
Shuffled temporal order	6	44.32
True-target decode	1	11.24

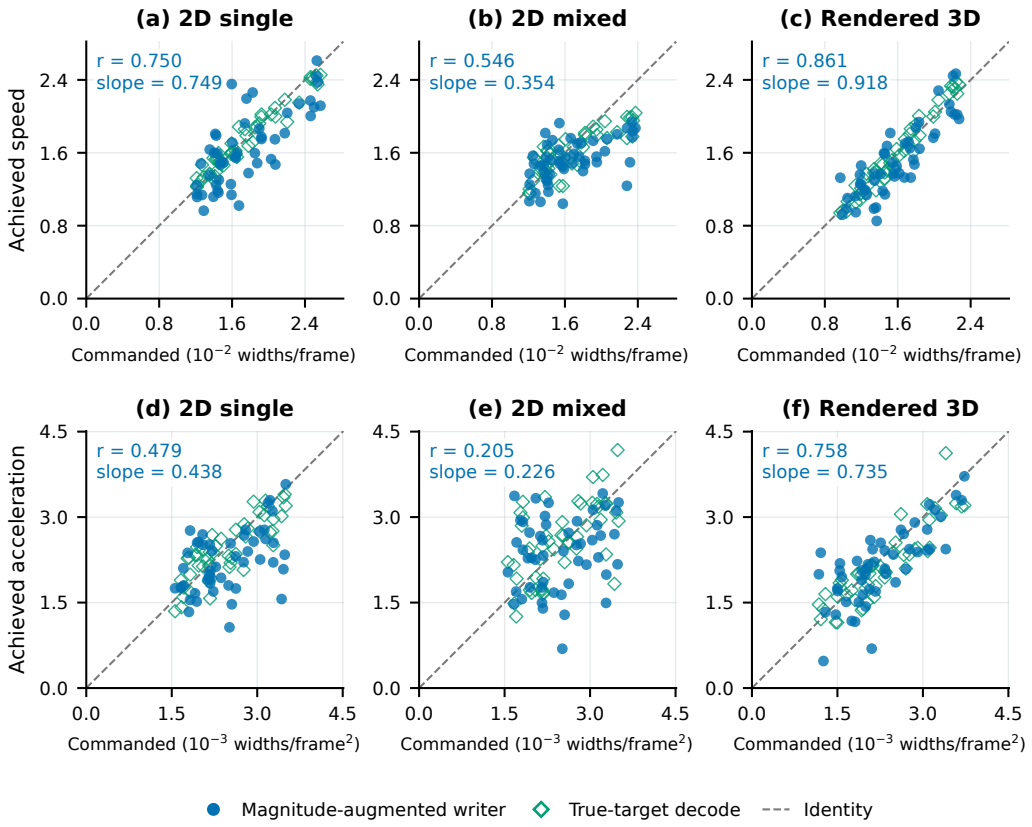


Figure 13: Commanded versus achieved magnitude for the final ViT-L writer with magnitude axes. Columns show single-appearance 2D, mixed-appearance 2D, and rendered 3D, and rows show velocity and acceleration. Each panel contains the same 50 test commands for the writer and its target decode. The dashed line is a perfect response. Annotations give the writer's Pearson correlation and regression slope, with an intercept fitted. Gains and ranks are in Table 7, and both axes use the units shown below each panel.

D.4 DECODER-GUIDED SEARCH

Per-scene optimization measures how much control the frozen decoder allows when we spend more computation on each edit. For every test scene, we optimize a latent edit directly by gradient descent. The edit is free apart from its rank, and the loss is a differentiable motion objective that compares the decoded motion with the supplied command, with gradients passing through the frozen decoder.

This search spends far more computation than the writer and shows what the decoder can render when the edit is unconstrained. After about 300 steps on 30 held-out scenes, it reaches a mean acceleration-direction error of 5.07° and a magnitude correlation of 0.94. The same procedure applied to gravity commands, which set the downward acceleration acting on the ball, reaches 0.87° and 0.97.

Table 10 reports runs of 300 steps on 30 held-out scenes per quantity, along with the edit rank and learning rate. Acceleration uses the mixed-appearance 2D ViT-L acceleration decoder, and gravity uses a separately trained ViT-L gravity decoder. These scenes form a different evaluation set from the one used for the shared writer, so the two sets of numbers are not directly comparable. The search also spends optimization steps on every test scene, while the shared writer predicts each edit in a single pass.

Table 10: Decoder-guided search. Cells are direction error (°) / magnitude correlation before and after 300 optimization steps. Each row uses 30 held-out scenes and its task-specific frozen decoder.

Quantity	Edit rank	Learning rate	Initial	Optimized
Acceleration	16	0.05	32.68 / -0.043	5.07 / 0.943
Gravity	128	0.08	3.48 / -0.066	0.87 / 0.970

E ROTATION AND ANGULAR-ACCELERATION TRANSFER

E.1 DECODED ANGULAR VELOCITY AND ACCELERATION

The orientation field is a harmonic function of order $K = 4$ that we fit separately at each temporal token and each cached layer, using ridge regression with penalty 10. We train it on 125 single-appearance scenes with constant angular velocity, after recentering the object on the token grid. To produce an edit, we evaluate the field along the source schedule and along the requested schedule and take the difference, as in Equation 4. We then undo the recentering and decode with the frozen orientation-supervised ViT-L decoder described in Appendix B.3.

Table 11 reports a target decode for each appearance setting, computed on scenes of that appearance. The angular-velocity correlation is nearly the same for single and mixed appearance, while the response slope drops from 0.741 to 0.668. The direction of rotation is correct on all 105 test pairs in both settings. Mixed appearance therefore weakens the size of the response and leaves the ordering of the decoded angular velocities almost intact.

Table 11: Decoded rotation with the ViT-L orientation field. Writer and shuffled-command scores use 105 pairs from 15 held-out scenes, and target decode scores use 210 pairs from all 30 evaluation scenes. The latter are appearance-specific references and not paired test-only ceilings. MAE is in rad/frame for velocity and rad/frame^2 for acceleration. All gains are selected on the separate 15-scene validation half.

			Writer			Shuffle	Target decode	
Quantity	Appearance	g	ρ	Slope	MAE	ρ	ρ	Slope
Angular velocity	Single	3	0.942	0.741	0.0436	0.026	0.977	0.900
Angular velocity	Mixed	3	0.945	0.668	0.0486	-0.187	0.979	0.914
Angular acceleration	Single	3	0.537	0.531	0.0082	0.053	0.809	0.626
Angular acceleration	Mixed	3	0.475	0.606	0.0096	-0.060	0.803	0.687

Figure 14 plots the angular-velocity rows of Table 11. The orientation field, fitted only on single-appearance scenes, carries over to mixed appearance without refitting. Its rank correlation stays at

0.942 and 0.945, close to the target decode at 0.977 and 0.979, while the shuffled-command baseline stays near zero at 0.026 and -0.187. The response slope falls from 0.74 to 0.67 while the target decode rises from 0.90 to 0.91. The loss of magnitude therefore lies with the field, since the decoder renders mixed-appearance rotation at least as well.

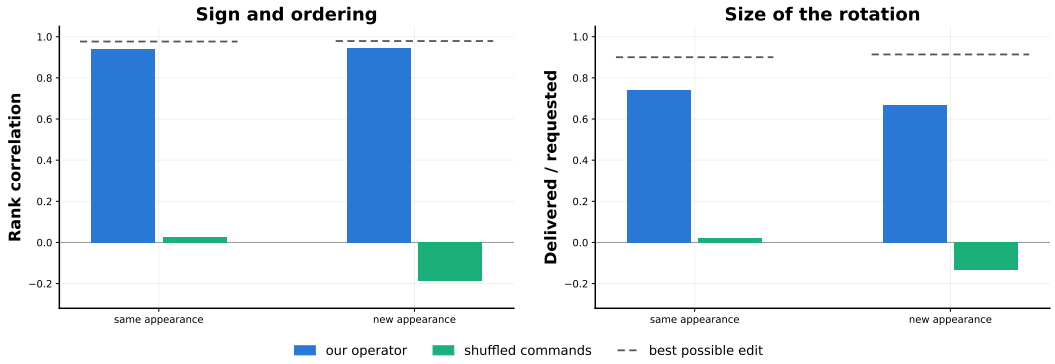


Figure 14: Angular-velocity steering on single-appearance scenes and on new mixed-appearance scenes. The left panel shows rank correlation between commanded and decoded angular velocity. The right panel shows the response slope, the fraction of the requested rotation that is realized. The orientation field and the shuffled-command baseline use 105 pairs from 15 test scenes, and the target decode uses 210 pairs from all 30 evaluation scenes.

For angular acceleration, the field stays the same and the schedule becomes quadratic, $\theta_\alpha(t) = \theta_0 + \omega_0\tau_t + \frac{1}{2}\alpha\tau_t^2$, where ω_0 is the initial angular velocity of the source clip. No accelerating clips are used to fit the field. The gain is the one quantity we calibrate on accelerating validation clips, and the selected value is $g = 3$. The transferred field reaches $\rho = 0.537$ on single-appearance test scenes and 0.475 under mixed appearance. A field refitted on accelerating clips reaches 0.512 in the single-appearance setting, so refitting gives no improvement over transfer (Table 12).

Table 12: Transfer versus refitting. Both use the same orientation-supervised ViT-L decoder and 105 single-appearance test pairs. Only the clips used to fit the field change, and gains use the validation half.

Writer	Field-fitting clips	ρ	Slope
Transferred field	Constant spin	0.537	0.531
Refitted field	Angular acceleration	0.512	0.535

E.2 Harmonics and decoder limits

Table 13 tests the transferred field before decoding, on 210 accelerating pairs from the 30 evaluation scenes. We measure all variants at layer 6. Fixing the layer in advance avoids picking the most favorable layer for each variant. With no orientation terms ($K = 0$), the field is constant in orientation and the predicted edit is zero. Even harmonics alone retain most of the alignment (0.493 against 0.504 for the full order-4 field), and odd harmonics alone reach only 0.096. This matches the half-turn symmetry of the bar's silhouette discussed in Section 4. Skipping recentering lowers the alignment to 0.348. All of these are alignment scores in latent space, and none of them measures decoded acceleration. Raising the order from 4 to 8 improves the latent score from 0.504 to 0.564, which on its own does not establish better rendered control.

Decoder quality limits the rendered test independently of the edit parameterization. In the ViT-H angular-acceleration transfer run, the target decode gives $\rho = -0.257$ on the 210-pair check, so the decoder fails to render angular acceleration even from the correct representation. That run therefore cannot test rendered acceleration, and we do not read it as a failure of the encoder to represent acceleration. Similarly, older experiments with a linear rotation writer used a different decoder with a weak target decode response, so they do not provide a like-for-like comparison with the orientation field.

Table 13: Orientation-field ablations transferred to angular acceleration. All fields train on 125 constant angular velocity scenes. Layer-6 cosine alignment is measured on 210 evaluation pairs, and no decoder is used. The baseline shuffles the command used for the latent prediction.

Field variant	Edit cosine	Shuffled cosine
No orientation terms ($K = 0$)	0.000	0.000
Order 2	0.428	0.018
Order 4 (used)	0.504	0.061
Order 8	0.564	0.100
Even harmonics only ($K = 4$)	0.493	0.061
Odd harmonics only ($K = 4$)	0.096	0.012
Without center registration	0.348	0.031

F Appearance, composition, and model scaling

F.1 Appearance and composition

The orientation fields for angular velocity and angular acceleration, fitted on single-appearance clips, keep their control in mixed-appearance scenes, and the mixed-appearance 2D and rendered 3D benchmarks test the translation writers across varied object appearance and rendering conditions. Appendix F.2 tests whether successive edits compose, and Appendix F.3 reports steering with ViT-H and ViT-g encoders.

F.2 Composition in decoded velocity

The composition tests use the additive velocity writer without magnitude axes, at gain 2, together with the frozen ViT-L `rolling_ball3d_decoder_fp` decoder on 40 held-out 3D scenes. Four tests check whether successive edits behave consistently. The identity test compares an edit whose requested velocity equals the source velocity with applying no edit at all. The inverse test applies a command followed by its reverse and compares the result with the unedited clip. The composition test applies two commands in succession and compares the result with a single command that goes directly to the final velocity. The commutativity test changes the two velocity components one at a time and compares the two possible orders. Table 14 reports each discrepancy as a difference in tracked velocity. We normalize by the size of the steering effect itself, which we define as the median change in decoded velocity between a direct command and no edit, namely 0.01182 image widths per frame.

Table 14: Decoded composition errors for the additive velocity writer without magnitude axes on 40 held-out 3D scenes. Absolute error is a median vector difference, in 10^{-3} image widths/frame. The last row compares two independent random edits of equal norm. It measures decoder sensitivity and not a random writer satisfying the same algebraic relation.

Test	Absolute error	Error / command effect
Identity	0.374	0.032
Inverse	0.278	0.024
Composition	0.425	0.036
Commutativity	0.262	0.022
Random Sensitivity Control	0.884	0.075

Small composition errors would mean little if the edits barely changed the motion, so we confirm that they do. On these same scenes, a single direct edit reduces the mean direction error from $171.9°$ to $8.8°$. The checks therefore show that decoded outcomes compose approximately under edits that substantially change the motion. We observe this behavior in the results, and we impose no group structure on the latent edits.

F.3 Decoded steering across encoders

Table 15(a) compares additive writers across encoder sizes. All rows use standardized features and a PCA basis without magnitude axes. Every row selects its gain on 50 validation scenes and reports results on 50 separate test scenes, using a decoder trained for that encoder and task. The ViT-L rows therefore correspond to the standardized variant of Table 7 and are weaker than the main method, which adds magnitude axes. For velocity, ViT-g raises the magnitude correlation from 0.465 to 0.703 relative to ViT-L and increases the direction error from $5.76°$ to $6.89°$. For acceleration, the larger encoders have nearly twice the direction error of ViT-L ($18.50°$ for ViT-H and $19.83°$ for ViT-g against $10.16°$), and the magnitude correlation of ViT-g falls to 0.142. A larger encoder therefore does not reliably give better decoded control.

Table 15: Decoded steering across encoders. (a) Single-appearance 2D translation with standardized additive writers and PCA bases without magnitude axes. Each row uses its encoder's frozen task decoder and 50 test scenes, and the target decode uses the same scenes. Decoder checkpoints are listed in Table 3. (b) Orientation fields for angular velocity with encoder-specific orientation decoders, using 105 test pairs for writers and shuffled commands and 210 evaluation pairs for target decodes. The main ViT-L rotation result is reported separately in Table 11.

(a) Translation with standardized writers

			Writer		Target decode	
Quantity	Encoder	Gain	Error (°)	r	Error (°)	r
Velocity	ViT-L	2	5.76	0.465	1.91	0.975
Velocity	ViT-g	2.5	6.89	0.703	1.22	0.970
Acceleration	ViT-L	2	10.16	0.359	5.98	0.830
Acceleration	ViT-H	2.5	18.50	0.372	6.61	0.752
Acceleration	ViT-g	2.5	19.83	0.142	14.78	0.480

(b) Angular velocity with orientation fields

Encoder	Appearance	Gain	Writer ρ	Slope	Target decode ρ	Shuffle ρ
ViT-H	Single	2	0.632	0.298	0.886	-0.040
ViT-H	Mixed	1	0.608	0.326	0.810	-0.110
ViT-g	Single	2	0.535	0.438	0.831	-0.033

The ViT-H and ViT-g rotation results show that decoded angular-velocity steering also works with other encoders (Table 15b). The writer correlations are 0.632 and 0.608 for ViT-H under single and mixed appearance and 0.535 for ViT-g, while the shuffled-command correlations lie between -0.110 and -0.033. The response slopes are low, between 0.298 and 0.438, so these writers realize well under half of each requested change in angular velocity. Each result should be read next to its target decode, because every encoder is paired with its own decoder. Differences across rows therefore reflect the encoder and decoder together and cannot be attributed to encoder size alone.

G Simulated control

G.1 Controller and goal baselines

The control experiment uses the additive writer without magnitude axes, on a frozen ViT-L representation. From the source state and the requested outgoing ball velocity, the writer produces a latent goal, that is, an edited post-contact representation. A separately trained inverse policy receives the source representation and the latent goal and predicts a continuous paddle action. The simulator executes the action, and the outgoing ball velocity gives the score. No video decoder is involved in scoring.

The scalar head-on task is approximately affine, with $v_{\text{out}} = 1.89953 v_p - 0.89855 v_{\text{in}}$ and a linearity residual of 0.106% of the output range. The paddle and the kinematic Franka agree to numerical precision. We use 80 physics substeps per frame and a solver tolerance of 10^{-10} to limit contact-discretization error. Table 16 reports the numeric-goal baseline and the two baselines that remove the correct goal. A trial is a scene–command pair. A scene can contribute multiple trials, so the trial counts are not counts of independent scenes. Evaluation scenes are held out from policy training.

Table 16: Scalar strike success within 2% of the requested speed, in percent. The learned policy uses a latent goal. The numeric-goal baseline supplies the physical command directly, and the shuffled and constant goals test whether the learned interface uses the requested goal. Outcomes are measured after simulated execution, without a video decoder.

Embodiment	Trials	Latent goal	Numeric goal	Shuffled goal	Constant goal
Paddle	800	100.0	100.0	32.4	5.1
Kinematic Franka	800	100.0	100.0	32.2	5.1
Actuated Franka	200	100.0	100.0	28.0	2.5

All three embodiments can execute the scalar command from the latent goal. The numeric-goal baseline succeeds as well, so this task establishes a working latent-goal interface and shows no advantage over explicit physical goals. The shuffled and constant goals show that the inverse policy depends on the goal, since the source state alone is not enough to solve the task.

G.2 Individual vector-task seeds

The 2D task commands an outgoing velocity vector. Table 17 reports the results for each seed. Each cell uses 40 held-out trials. The actuated Franka averages 80.5% success with the nonlinear inverse policy and 20.5% with the linear policy. Both policies solve the abstract paddle task. These are execution results with the same latent-goal interface. We did not evaluate the final translation writer with magnitude axes in this setting.

Table 17: Vector-task success within 5% relative velocity error, in percent. Each seed has 40 test trials per embodiment and compares linear and nonlinear inverse policies. No decoder is used for scoring.

	Paddle		Actuated Franka	
Seed	Linear	Nonlinear	Linear	Nonlinear
1	100.0	100.0	17.5	77.5
2	100.0	100.0	27.5	87.5
3	100.0	100.0	20.0	82.5
4	100.0	100.0	17.5	80.0
5	100.0	100.0	20.0	75.0