Make the Generator Its Own Physicist: Latent Conserved Charges for Text-to-Video

Method. Latent Conserved Charges (LCC)

Motivation

Modern text-to-video (T2V) models — diffusion models that turn a sentence into a video — now produce sharp, smooth clips. But they were trained only to make videos LOOK right, frame by frame. They never learned that the things in the video have to obey physics. So when you ask for something the training videos rarely showed, the motion goes wrong: a ball falls upward, an object suddenly speeds up, momentum appears from nothing. This isn't fixed by adding more training videos — the model simply has no internal notion of a physical state that must follow a law.

Everyone's current fix adds the physics from OUTSIDE the generator. NewtonGen trains a separate little physics simulator and pipes its predictions into a frozen video model. WISA feeds in hints about which physical category applies. PhysCorr trains a reward model that scores how physical a clip looks and nudges the generator toward higher scores. PhyT2V just rewrites the prompt. Each only handles the cases its outside helper was built for, and none of them changes the fact that the video model's own internal representation still carries no physical state.

Why this is the moment: open video models (Open-Sora, CogVideoX-2B) now let us reach inside and read/modify their internal per-frame state directly. And a recent line of 'world models' for robot planning — LaWM, Arkhon, PH-Dreamer — showed that you can build a special update rule on a model's internal state that makes physics hold automatically, by construction, rather than scoring a finished video. That idea has not yet been brought into open-domain text-to-video. Our move: make the video model's own internal state BE the physical state — read a small conserved quantity off it, use a special 'symplectic' update step that keeps that quantity constant automatically, and measure physics directly from the state (no trained scorer), learning it all from ordinary video.

Method

Background

Unchanged scaffolding: expose the backbone, build the foreground mask, generate/evaluate, and run attribution ablations.

1. Run the pretrained video generator and attach hooks that capture, at a late denoising step where the scene has stabilized, its internal per-frame latent tensors and its text-to-image attention maps (averaged over attention heads and steps to one map per word, at the latent's resolution). Don't change any weights yet.

   Why: We want to enforce physics on the model's OWN internal state, so first we have to be able to see and touch it.

2. Find which word in the prompt names the moving object — the grammatical subject's main noun, or the word the model attends to most — then build a per-frame mask by normalizing that word's attention map to 0–1, matching it to the latent's spatial grid, thresholding at 0.5, and keeping the single biggest connected blob.

   Why: Physics should be measured on the thing that moves, and the model already tells us where that is.

3. At generation time just generate as usual — conservation is now baked in — and record the residual and the charge trajectory for each clip. Score a clip as physically passing if its residual stays inside a

small tolerance (set from the calibration noise level) for every frame and its motion matches the textbook solution; for the projectile/collision tests, make ground-truth trajectories by rendering synthetic clips in a physics engine. Evaluate on PhyWorldBench, T2VPhysBench, and that subset.

Why: Because conservation is built into the update, the frames obey the law, and ρ_t is a free per-generation physics readout.

4. Run ablations that pin each effect to a specific part. Negative control: swap the symplectic step for an ordinary learned update with the same number of parameters and inputs, keeping everything else — physics should fall back to the base generator, showing it was the structure (not extra capacity) that helped. Positive control: keep the integrator and readout but drop the contrastive loss — most of the trajectory-error gain should survive. Also add a baseline that builds the same pairs but scores them with a plain inner product instead of charge-distance.

   Why: These pin the gains to the right cause: the by-construction integrator for conservation, the charge-distance similarity for readability.

Latent Physical State

Turn the generator's own per-frame latent into a measurable physical-state vector (the conserved charge).

5. Add a small linear readout R and define the charge as $q(z_t) = R \cdot pool(z_t \odot m_t)$, where pool is the mask-weighted average of the latent over the foreground (sum the masked latent, divide by the mask area) — a 6-number vector per frame. Calibrate R once by least-squares against center-of-mass and momentum estimated from optical flow, on a few hundred clips, so the numbers mean something physical. 【author decision: a 6-number charge meant as 3D position + 3D momentum cannot be calibrated from ordinary 2D video, where depth and absolute scale are unobservable and optical flow only gives in-plane motion — the authors must either keep the charge to in-plane quantities (2D position + 2D momentum plus angular momentum/energy) calibrated against 2D flow, or add a monocular-depth estimator to recover 3D and accept its error; this choice of what the 6 numbers mean is theirs to make.】 *The latent conserved charge q is a learned linear readout R over the masked spatial average of the per-frame denoising latent z_t; its 6 components are 3 center-of-mass position coefficients and 3 generalized-momentum components.*

$$q(z_t) = R\,\mathrm{pool}(z_t \odot m_t), \quad R \in \mathbb{R}^{d_q \times d},\ d_q = 6 \tag{1}$$

   Why: This turns the model's internal state into an actual physical-state vector we can conserve and measure.

By-Construction Conservation

Replace the temporal latent update with a symplectic integrator so the charge is conserved automatically, relocating physics enforcement inside the generator.

6. Swap the model's frame-to-frame update for a leapfrog (symplectic) integrator that splits the latent channels into a 'position' half and a 'momentum' half and steps them with a small learned force, in the half-kick / drift / half-kick order; the step size is one over the frame rate. 【author decision: just calling half the channels 'position' and half 'momentum' does not make the dynamics conserve anything — leapfrog only conserves the charge if the force is the slope of a single scalar energy and the split is a true position/momentum pairing; whether the existing latent allows such a split or needs a learned change of coordinates, and which symmetry gives which conserved number, is the core design choice the whole 'conserved by construction' claim depends on and must be decided by the authors.】 If the step is genuinely symplectic, the charge stays exactly constant along the video (when no external field is present) no matter what the force outputs, and that carries through to the decoded frames. *The temporal latent update is a structure-preserving (leapfrog/symplectic) integrator Φ driven by a*

learned generalized force; because it exactly conserves its discrete momentum map regardless of the force field, the charge q is invariant by construction when the prompt specifies no external field.

$$z_{t+1} = \Phi_{\Delta t}\big(z_t, f_\theta(z_t, c)\big), \qquad q(z_{t+1}) = q(z_t) \;\text{ when }\; F_t = 0 \tag{2}$$

Why: A symplectic step automatically keeps its momentum quantity constant no matter what the force is — so the charge q stays conserved by construction, not by hoping the model behaves.

Proxy-Free Signal & Objective

Compute the conservation residual directly from the latent and train the charge to be linearly readable from ordinary video via a charge-contrastive objective.

7. Read any gravity cue from the prompt (default to none, i.e. a closed system) and turn it into the expected per-step change; the mass scale is already set by the step 5 (Add a small linear readout *Rand*...) calibration. Then compute the residual as the change in the charge from one frame to the next minus that expected change, straight from the latents — it is exactly zero when the physical law holds, and needs no learned reward model. *The conservation residual is the discrete derivative of the charge minus the parsed external-field update; computed entirely from latents and a parsed constant, it equals zero exactly when the physical law holds, with no learned reward.*

$$\rho_t = q(z_{t+1}) - q(z_t) - \Delta t\, F_t, \qquad F_t = m\, g \tag{3}$$

Why: ρ_t is exactly zero when the physical law holds, giving a direct physics check that needs no trained scorer that could be wrong on new motions.

8. Train with a contrastive loss built around the charge. Positive pairs are versions of a clip that should keep the same charge (retiming, rigid camera moves); negative pairs are clips that look the same but have a velocity jump spliced in (re-time the second half). Encode every clip through the generator to get its charge, and measure similarity as the negative distance between charges (divided by a temperature set to the typical charge gap in the batch), not as an inner product — this forces the geometry where the charge can be read off linearly. *The charge-contrastive objective replaces the default inner-product similarity with the negative charge-distance scaled by temperature, so the only loss-minimizing geometry is one in which the charge is linearly decodable and conservation-violating trajectories are pushed apart.*

$$\mathrm{sim}(z, z') = -\,\|q(z) - q(z')\|/\lambda_c \tag{4}$$

Why: Using charge-distance as the similarity forces the representation to make the charge cleanly readable — the one thing that minimizes the loss is encoding q.

9. Fine-tune the generator with lightweight LoRA adapters (rank 16 on its attention and MLP layers), training the readout and force field together, using the charge-contrastive loss plus a small dose (weight 0.1) of the model's normal denoising loss so video quality is preserved — all on ordinary video, not a physics-curated set.

   Why: One trained system on ordinary video — strictly fewer moving parts than NewtonGen's separate simulator plus a physics-clean dataset.