Single Agent Architecture as Field Theory: Technical TLDR

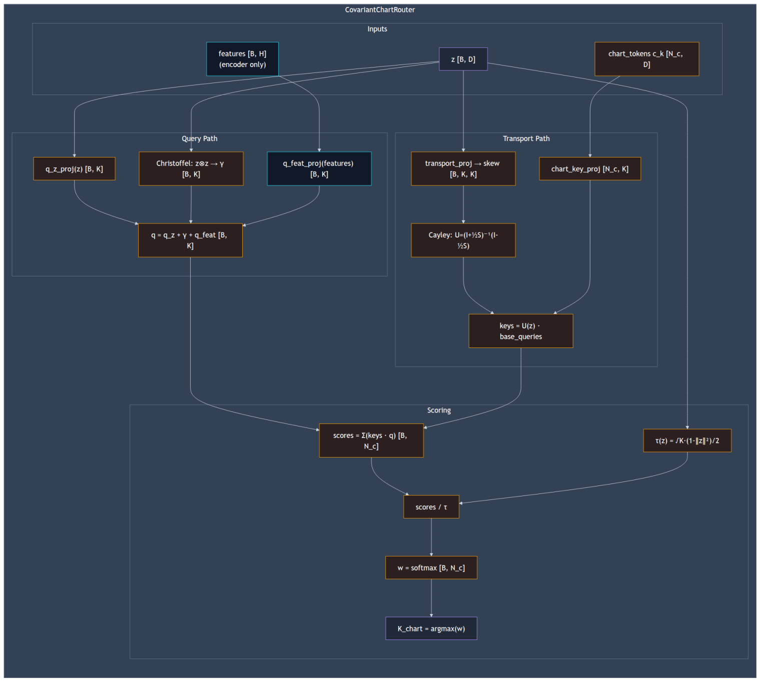
Guillem Duran Ballester, Jan 2026

0. Architecture

The following diagrams illustrate the current implementation architecture of the TopoEncoder system, showing how observations are encoded into the split-latent space (K, z_n, z_{tex}) and decoded back to reconstructions. These diagrams mirror the code in `src/fragile/core/layers/atlas.py` and `src/experiments/topoencoder_2d.py`.

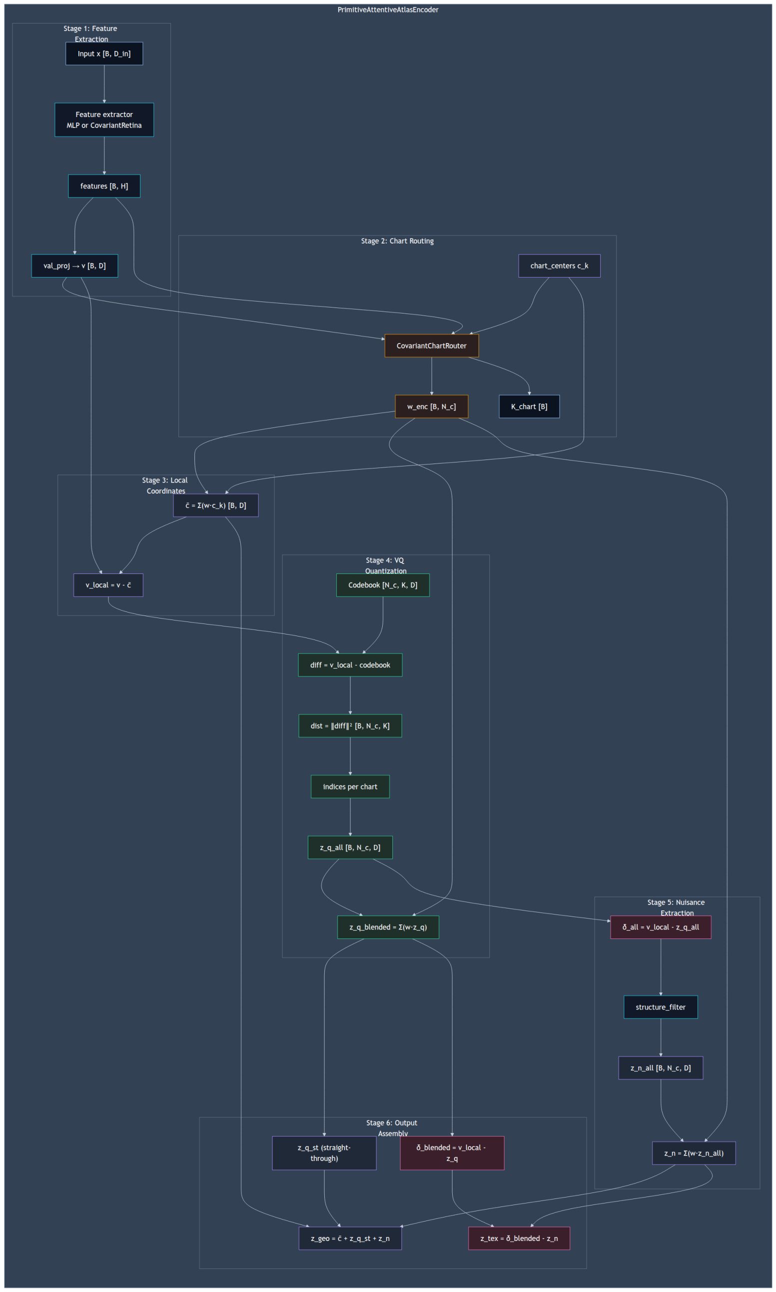
0.1 CovariantChartRouter

The chart router is shared by both encoder and decoder. It performs metric-aware, covariant chart assignment using: - Geodesic query terms (linear + quadratic Christoffel correction) - Wilson-line transport via Cayley transform for gauge invariance - Position-dependent temperature scaled by local metric (conformal factor)


CovariantChartRouter
Inputs
features [B, H] (encoder only)
z [B, D]
chart_tokens c_k [N_c, D]
Query Path
q_z_proj(z) [B, K]
Christoffel: z⊗z → γ [B, K]
q_feat_proj(features) [B, K]
q = q_z + γ + q_feat [B, K]
Transport Path
transport_proj → skew [B, K, K]
chart_key_proj [N_c, K]
Cayley: U=(I+½S)⁻¹(I-½S)
keys = U(z) · base_queries
Scoring
scores = Σ(keys · q) [B, N_c]
τ(z) = √K·(1-‖z‖²)/2
scores / τ
w = softmax [B, N_c]
K_chart = argmax(w)

0.2 Encoder (PrimitiveAttentiveAtlasEncoder)

The encoder performs feature extraction, chart routing, VQ quantization per chart, and splits the latent into (z_{geo}, z_n, z_{tex}):

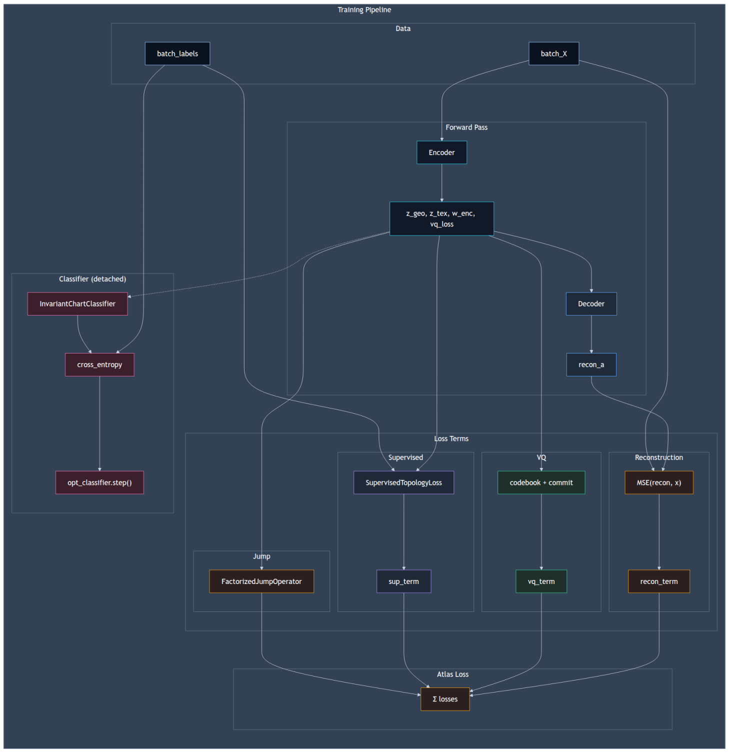


0.3 Decoder (PrimitiveTopologicalDecoder)

The decoder performs chart-weighted reconstruction from the geometric latent z_{geo} and adds the texture residual z_{tex}: - Geometric path: Chart projectors → Chart-weighted mixing → Renderer - Texture path: Independent residual network - Final output: Base reconstruction + scaled texture residual

0.4 Experiment Wiring (Training Losses)

Optional training components available in the experiment configuration: - Learned precisions for automatic loss balancing (reconstruction, VQ, supervised) - SupervisedTopologyLoss for semantic topology learning - FactorizedJumpOperator for chart transition consistency - InvariantChartClassifier (detached readout head with separate optimizer)



1. Latent Space Decomposition

Split-Latent Structure:

- Total latent: $Z = (K, z_n, z_{\text{tex}})$ where each component has distinct geometric/physical role
- $K \in \{1, \dots, N_c\}$: Discrete macro state (VQ codebook index) - chart assignment on topological atlas
- $z_n \in \mathbb{R}^{D_n}$: Continuous nuisance latent - local coordinates within chart (gauge-invariant position)
- $z_{\text{tex}} \in \mathbb{R}^{D_t}$: Texture residual - holographic boundary degrees of freedom
- Decomposition satisfies $z_e = z_q + z_n + z_{\text{tex}}$ where z_e is raw encoder output, z_q is VQ-quantized macro

Encoder Architecture (AttentiveAtlasEncoder):

- Feature extraction: Conv layers (64→128→256 channels) → hidden_dim projection
- Cross-attention routing: $w_k = \text{softmax}(\langle K_k, Q(x)\rangle/\sqrt{D})$ where K_k are learnable chart queries
- Query projection: $Q(x) = \text{key\_proj}(\text{features}(x))$ with LayerNorm stabilization
- Chart assignment: $K = \arg\max_k w_k(x)$
- VQ per chart: N_c independent codebooks, each with K_c codes, vectorized quantization
- Nuisance extraction: Structure filter $z_n = f_{\text{struct}}(z_e - z_q)$ removes VQ-residual structure
- Texture: Holographic residual $z_{\text{tex}} = (z_e - z_q) - z_n$ orthogonal to both macro and nuisance.
- **Texture Firewall:** Dynamics depend on z_{macro} and z_n, screening out only z_{tex}.

Training Objectives for Coarse-Graining: To enforce this split-latent structure and ensure valid coarse-graining, we minimize:

$$\mathcal{L}_{\text{latent}} = \mathcal{L}_{\text{VQ}} + \mathcal{L}_{\text{closure}} + \mathcal{L}_{\text{slowness}} + \mathcal{L}_{\text{disentangle}}$$

1. **VQ Loss:** $\|z_e - z_q\|^2 + \beta \|z_q - \text{sg}[z_e]\|^2$. Stabilizes the discrete macro state K.
2. **Causal Enclosure:** $-\log p(K_{t+1}|K_t, a_t)$. Ensures macro dynamics are self-contained (predictable without micro details).
3. **Slowness (Anti-Churn):** $\|e_{K_t} - e_{K_{t-1}}\|_G^2$. Penalizes rapid flickering of the macro state to ensure temporal stability.
4. **Disentanglement:**
   - *Nuisance KL:* $D_{\text{KL}}(q(z_n|x)\|\mathcal{N}(0, I))$. Minimal sufficient nuisance.
   - *Texture KL:* $D_{\text{KL}}(q(z_{\text{tex}}|x)\|\mathcal{N}(0, I))$. Texture should contain no macro info.

Geometric Regularization (Quality Control): To ensure the latent space is well-conditioned (high-fidelity, isometric charts), we add:

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{VICReg}} + \mathcal{L}_{\text{ortho}}$$

5. **VICReg (Self-Supervision):** Prevents collapse without negative pairs.
   - *Invariance:* $\|z - z'\|_G^2$. Robustness to view augmentation.
   - *Variance:* $\max(0, \gamma - \sqrt{\text{Var}(z)})$. Forces code utilization.
   - *Covariance:* $C(z) \approx I$. Decorrelates latent dimensions (whitening).
6. **Orthogonality (Chart Isometry):** $\|W^T W - I\|_F^2$ on encoder weights. Ensures the mapping from observation to latent space is locally isometric (preserves distances), crucial for meaningful geodesic calculations.

2. The Reward Field & Hodge Decomposition

Physical Context: We model the agent as a particle with **Position and Momentum** performing a **Geodesic Random Walk** on the latent manifold. Because utility is harvested via trajectory traversal, the Reward is naturally defined as a differential **1-form** coupled to velocity, rather than a static scalar field.

Constraint: Reward is not a scalar $r(z)$, but a **1-form** $\mathcal{R}$ that depends on direction ($\mathcal{R}_i \dot{z}^i$). This requires a field-theoretic treatment of value.

Hodge Decomposition: The reward 1-form uniquely decomposes into three orthogonal components:

$$\mathcal{R} = \underbrace{d\Phi}_{\text{Gradient}} + \underbrace{\delta\Psi}_{\text{Solenoidal}} + \underbrace{\eta}_{\text{Harmonic}}$$

- **Gradient (Φ):** Optimizable scalar potential (Conservative Value).
- **Solenoidal (Ψ):** Vector potential generating the **Value Curl** (Magnetic Field) $\mathcal{F} = d\mathcal{R} = d\delta\Psi$.
  - *Physics:* Just as a magnetic field B exerts a **Lorentz Force** $v \times B$, the Value Curl $\mathcal{F}$ exerts a velocity-dependent force $\mathcal{F}_{ij}\dot{z}^j$ that steers the agent to **orbit** value cycles rather than converge.
- **Harmonic (η):** Topological cycles (manifold holes).

The Screened Poisson Equation (Conservative Case): When the **Value Curl** vanishes ($\mathcal{F} = 0$), the scalar value function $V(z)$ satisfies the **Helmholtz Equation** on the manifold:

$$(-\Delta_G + \kappa^2)V(z) = \rho_r(z)$$

where Δ_G is the Laplace-Beltrami operator, κ is the screening mass, and ρ_r is the reward source density.

The Composite Navigation Potential (Runtime): The agent navigates a **Composite Potential** Φ_{eff} constructed at runtime by summing learned and intrinsic signals:

$$\Phi_{\text{eff}}(z) = \underbrace{V(z)}_{\text{Learned}} + \underbrace{U(z)}_{\text{Intrinsic}} + \underbrace{\mathcal{B}_{\text{safety}}(z)}_{\text{Fixed}}$$

1. **Control (V): Learned** scalar value. Drives the agent towards high-reward regions.
   - *Loss:* Helmholtz Residual (see below).
2. **Generation (U): Intrinsic** entropy potential (e.g., Hyperbolic expansion $-\log \mathrm{vol}(z)$). Drives exploration away from the origin.
   - *Loss:* None (Fixed geometric prior).
3. **Safety ($\mathcal{B}_{\text{safety}}$): Fixed** safety barrier. Hard constraints (e.g., capacity limits) modeled as high-energy walls.
   - *Loss:* None (Fixed constraint).

Neural Hodge Decomposition (Implementation): We approximate the Hodge components using a **Multi-Head Network** sharing a common feature backbone:

1. **Scalar Head (Φ):** Outputs scalar $V(z)$.
   - *Loss:* Helmholtz Residual on the symmetric part of the reward.
   - $\mathcal{L}_\Phi = \|V(z) - (r_{\text{sym}} + \gamma V(z'))\|^2$.
2. **Solenoidal Head (Ψ):** Outputs vector potential $A(z) \in \mathbb{R}^D$ (where $\mathcal{F} = dA$).
   - *Loss:* Residual reconstruction on the antisymmetric part.
   - $\mathcal{L}_\Psi = \|\langle \mathcal{R}, v\rangle - (\langle \nabla\Phi, v\rangle + \langle \nabla \times A, v\rangle)\|^2$. The curl absorbs the non-integrable reward residual.
3. **Harmonic Head (η):** A set of **learnable constant 1-forms** η_k per chart k.
   - *Mechanism:* Captures global topological currents (net flux through manifold holes) that are locally constant.
   - *Loss:* Projected residual after removing Gradient and Curl components.

Value Function Objectives: To define the value field $V(z)$ as the solution to the Screened Poisson Equation, we minimize:

$$\mathcal{L}_{\text{critic}} = \underbrace{\|V(z) - (r + \gamma V(z'))\|^2}_{\text{Helmholtz Residual (TD)}} + \lambda_{\text{geo}} \underbrace{\|\nabla_G V\|^2}_{\text{Smoothness}}$$

1. **Helmholtz Residual:** Enforces the PDE source term (approx. Bellman error).
2. **Geometric Regularization:** Enforces field smoothness with respect to the manifold metric ($\|\nabla_G V\|^2 = G^{ij}\partial_i V \partial_j V$).

3. The Policy Network (Latent Action)

The Policy acts as an **External Force Field** $u_\pi(z)$ pushing the agent through the latent manifold. It operates in a **Latent Action Space** (the Tangent Bundle $T\mathcal{Z}$), decoupling low-level motor commands from high-level intent.

A. The Policy Model (π_ϕ):

- **Role:** Symmetry-breaking control field. Converts potential energy into kinetic motion.
- **Input:** Latent State $z_t \in \mathcal{Z}$ (Position).
- **Output:** Latent Force/Action $u_t \in T_{z_t}\mathcal{Z}$ (Tangent Vector).
  - *Note:* This latent force is subsequently decoded into boundary motor torques action a_t by the Motor/Action Decoder (Neumann Condition).
- **Latent Action Space:** The Tangent Bundle $T\mathcal{Z}$. Actions are vectors "pushing" the state along geodesics.

B. Training Losses (Tier 1):

1. **Task Loss ($\mathcal{L}_{\textbf{task}}$):** Standard Policy Gradient / Reinforce objective to maximize expected returns.
2. **Entropy Bonus (Ergodicity):** $\mathcal{L}_{\text{ent}} = -H(\pi)$. Penalizes low entropy distributions to prevent premature mode collapse and ensure thermodynamic equilibrium.
3. **Zeno Penalty (Temporal Smoothness):** $\mathcal{L}_{\text{zeno}} = D_{\text{KL}}(\pi_t \| \pi_{t-1})$. Penalizes infinite-frequency oscillations (Zeno behavior) to ensure physically realizable trajectories.

4. The World Model (Covariant Integrator)

We define the World Model not as a generic RNN, but as a **Neural Integrator** that approximates the **Lorentz-Langevin SDE**:

$$dz^k = \underbrace{\left(-G^{kj}\partial_j \Phi + u_\pi^k\right) ds}_{\text{Gradient + Policy}} + \underbrace{\beta G^{km}\mathcal{F}_{mj}\dot{z}^j ds}_{\text{Lorentz Force}} - \underbrace{\Gamma^k_{ij}\dot{z}^i\dot{z}^j ds}_{\text{Geodesic Drift}} + \underbrace{\sqrt{2T_c}(G^{-1/2})^{jk} dW^j}_{\text{Thermal Noise}}$$

This equation unifies:

1. **Gradient Descent** (on the Value Landscape)
2. **Magnetic Steering** (from Value Curl)
3. **Geodesic Motion** (on the curved Manifold)
4. **Stochastic Exploration** (Langevin Dynamics)

The integration step is modeled as a **Covariant Cross-Attention** layer (Multi-Head Transformer).

A. Architecture: Covariant Cross-Attention

- **Mechanism:** Attention heads act as **Wilson Lines** (Parallel Transport operators), comparing queries and keys only after transporting them to a common reference frame (Gauge Invariance).
- **Metric-Temperature:** The softmax temperature is position-dependent: $\tau(z) \propto 1/\lambda(z)$. High-curvature regions (large metric λ) force low temperature (sharp attention), while flat regions allow high temperature (broad exploration).
- **Geodesic Correction:** Christoffel symbols are encoded via linear and quadratic terms in the Query projection.

B. Inputs & Outputs (Integration Step):

- **Input:**
  - Current State z_t (Query position).
  - Action u_t (Latent Force/Momentum).
  - Memory Context (Keys/Values from past trajectory).
- **Output:**
  - Next State z_{t+1} (Integrated position after Kick-Drift-Kick).

C. Training Losses:

1. **Geodesic Distance Loss:** $\mathcal{L}_{\text{geo}} \approx (z_{\text{pred}} - z_{\text{true}})^T G(z_t)(z_{\text{pred}} - z_{\text{true}})$. Minimizes local Riemannian distance. Using the diagonal approximation (Section 5), this becomes a **Weighted MSE**: $\sum_i G_{ii}(z^{(i)}_{\text{pred}} - z^{(i)}_{\text{true}})^2$. High-risk dimensions (large G_{ii}) are penalized more heavily.

2. **Thermodynamic Consistency:** $\mathcal{L}_{\text{NLL}} = -\log p(z_{t+1}|z_t, u_t)$. Ensures the model captures the stochastic thermal noise term ($\sqrt{2T_c}dW$) correctly.

D. Structural Inductive Bias (Why it acts as an Integrator): We do not simply train a generic MLP to output z_{t+1}. Instead, we bake the **Boris-BAOAB** integration scheme directly into the attention mechanism, ensuring the model cannot violate the symplectic structure:

1. **Metric as Temperature:** The attention temperature $\tau(z) \propto \sqrt{d_k}/\lambda(z)$ forces the update step size to scale inversely with curvature (conformal factor). High-curvature regions (large metric) automatically induce small, cautious steps (sharp attention).
2. **Geodesic Query Terms:** We explicitly feed geometric terms $(z, z \otimes z)$ into the Query projection $Q(z)$. This forces the attention scores to learn the **Christoffel Symbols** Γ^k_{ij} needed to correct for manifold curvature, rather than making up arbitrary dynamics.
3. **Operator Splitting:** We use **multiple attention heads** to implement the split operators of the BAOAB scheme:
   - *Kick Head (B):* Updates momentum using force (Gradient + Curl).
   - *Drift Head (A):* Updates position using momentum (Geodesic flow).
   - *Thermostat Head (O):* Applies friction and noise (OU process).
   - *Result:* The network is forced to learn a decomposable, reversible integrator rather than a "black box" transition.

5. Efficient Metric Computation (Adam-Style)

Computing the full Riemannian metric G_{ij} ($D \times D$ tensor) is expensive ($O(D^3)$ inversion). We use the same engineering tricks as the **Adam Optimizer** to approximate it efficiently ($O(D)$).

A. The "Adam" Isomorphism:

- **Adam Optimizer:** Maintains a diagonal approximation of the Hessian (via squared gradients) to precondition updates.
  - $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Second Moment Estimate).
  - Preconditioner: $P = \text{diag}(1/\sqrt{v_t})$.
- **Fragile Agent:** Maintains a diagonal approximation of the Metric Tensor (via Risk Tensor) to curve the space.
  - $\text{Metric}_t = \beta \text{Metric}_{t-1} + (1 - \beta)\text{Risk}_t$ (Metric Evolution).
  - Geometry: $G_{ij} \approx \text{diag}(\text{Metric}_t)$.

B. Implementation Details:

1. **Diagonal Approximation:** We assume G_{ij} is diagonal (independent curvature per dimension). This reduces storage from $O(D^2)$ to $O(D)$ and inversion to element-wise division.
2. **Risk as Squared Gradient:** The Risk Tensor T_{ij} is dominated by the gradient of the potential: $T_{ij} \approx \partial_i \Phi \partial_j \Phi$. Its diagonal is simply $(\nabla \Phi)^2$.
3. **Low-Rank Updates (EMA):** We do not solve the Einstein Field Equations at every step. Instead, we update the metric using an **Exponential Moving Average (EMA)** of the Risk Tensor.
   - *Update Rule:* `metric_diag.lerp_(risk_diag, 1 - momentum)`
   - *Interpretation:* The geometry "flows" slowly towards the high-risk regions, smoothing out transient noise just like Adam smooths gradient variance.

Result: We get Riemannian Manifold Hamiltonian Monte Carlo (RMHMC) benefits for the cost of standard SGD+Momentum.