JEPA-X: Real-Time Safe Robot Planning via Continuous Latent Trajectory Prediction and Analytical Collision Certification

Jean-Luc Tarik Aslan and Dongwei Zhang
Beijing Tangta-Technology Co., Ltd., Beijing, China
Email: aslan.tarik@tangta-technology.com
DOI: 10.51094/jxiv.5833 - https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

AUTHOR'S NOTE

This document presents JEPA-X as a research architecture for continuous-time predictive robot planning with validated real-world performance. The core methodology is derived from the Company's JEPA-X technical implementation and is protected by pending patent applications (Ref. [11]) and registered software copyright (Ref. [12]). Claims in this publication are stated in scientific language with appropriate caveats: a trajectory is called analytically certified collision-free ***only under the modeled geometry, state, and uncertainty assumptions*** *used by the certification module.*

The JEPA-X Cognitive Decision System (Software Copyright No. 2026R11L2342491, 2026R11L2613892) is a registered software work with comprehensive technical documentation. For licensing, collaboration, or technical inquiries, please contact the corresponding author at 3559870647@qq.com.

ABSTRACT

***Abstract*—Latent-space world models provide compact representations for robot prediction and planning, but many existing approaches represent future evolution through discrete state transitions. Such representations do not directly describe the continuous physical motion between predicted states, creating a verification gap when robots move rapidly or interact with dynamic obstacles.**

We present JEPA-X, an action-conditioned continuous-time world-model architecture that jointly predicts latent semantic evolution and physical 3D trajectories. The environment is represented as a Structured Latent Scene Graph, while a Latent Kinematic Path Predictor generates anchored Bernstein-polynomial trajectories for the robot and surrounding objects within a shared temporal parameterization. This formulation supports continuous relative-distance, minimum-separation, velocity, acceleration, and time-to-collision evaluation.

JEPA-X incorporates a two-stage collision-assessment pipeline. A conservative convex-hull separation test provides a sufficient geometric safety certificate under the modeled geometry and uncertainty assumptions. When this certificate is unavailable, a GPU-based sampled curve-distance refinement estimates minimum trajectory separation. Control-point uncertainty is propagated into time-varying uncertainty envelopes and mapped to graded execution states, enabling speed reduction, increased clearance, replanning, or stopping under uncertainty. The system further supports multi-timescale prediction and segmented trajectories with smooth position, velocity, and acceleration transitions.

Prototype evaluation on an RTX 2070 and Jetson Orin NX demonstrates average end-to-end latencies of 7.2 ms and 16.2 ms, respectively, under the reported benchmark configuration. GPU-batched candidate evaluation provides 16.9× and 22.0× acceleration relative to sequential execution on the same hardware. These results indicate that continuous latent-physical trajectory prediction is computationally feasible for real-time edge-robot planning. Physical full-body humanoid validation and dynamic articulated swept-volume modeling remain future work.

***Index Terms*—Joint Embedding Predictive Architecture (JEPA), Structured Latent Scene Graph (SLSG), continuous trajectory prediction, latent world model, Model Predictive Control (MPC), collision avoidance, Bézier curves, uncertainty-aware planning, time-to-collision (TTC), edge robotics.**

INDEX TERMS

Joint Embedding Predictive Architecture (JEPA), Structured Latent Scene Graph (SLSG), continuous trajectory prediction, latent world model, Model Predictive Control (MPC), collision avoidance, Bézier curves, uncertainty-aware planning, time-to-collision (TTC), edge robotics.

I. INTRODUCTION

A central challenge in autonomous robotics is to combine compact learned world models with continuous-time physical safety. Joint-embedding and latent-space predictive architectures reduce the cost of representing high-dimensional sensory input, but planning commonly proceeds through discrete future states. Even when two consecutive predicted states are individually valid, the continuous physical motion between them can still intersect an obstacle. This **"temporal blind spot"** issue becomes critical when robots move quickly, operate near humans, or interact with dynamic objects.

A second difficulty is the **"representation disconnect"** between cognitive representation and safety verification. High-level planning may take place in a learned latent space, whereas collision checking is performed by a separate Cartesian-space module. This introduces representation conversion, additional latency, and error accumulation. For highly reactive Model Predictive Control (MPC), the robot requires a representation in which future semantic state, physical motion, uncertainty, and safety constraints can be evaluated coherently.

To mitigate this, existing systems decouple latent-space cognition from Cartesian-space safety. They either decode

latent states back to pixel space for collision checking (incurring $\sim$100$\times$ computational overhead) or offload verification to external 3D physics engines (e.g., MoveIt [4], MuJoCo). This "representation disconnect" introduces severe latency, coordinate transformation error accumulation, and fundamentally precludes real-time Model Predictive Control (MPC) on resource-constrained edge hardware. Furthermore, these discrete or decoupled systems lack continuous-time causal reasoning, failing to predict secondary physical consequences—for example, a humanoid walking through a crowded area when a running child suddenly crosses its path, creating an imminent collision risk that could be dangerous for the child.

JEPA-X addresses this problem by replacing discrete future-point prediction with continuous, action-conditioned trajectory prediction. A Structured Latent Scene Graph represents objects, relations, and physical anchors. The Latent Kinematic Path Predictor maps the current structured state and a candidate action to synchronized latent and physical Bernstein-polynomial trajectories. The Ego-Node uses the same formulation as dynamic objects, so relative motion can be evaluated directly in continuous time.

A. Primary Contributions

1) **Continuous Latent Trajectory Parameterization**: A Latent Kinematic Path Predictor (LKPP) outputs dual-branch Bézier control points (latent and physical) conditioned on a Structured Latent Scene Graph (SLSG). Strict anchor constraints ($C_0 = p_{\text{current}}$) eliminate latent-space hallucinations and ensure physical continuity.
2) **Unified Ego-Motion and Dynamic-Object Continuous Prediction**: The robot and surrounding objects are predicted using the same continuous temporal parameterization, enabling direct relative-distance and TTC reasoning.
3) **Pure-GPU Analytical Collision Certification**: A novel two-stage safety pipeline utilizing $\mathcal{O}(1)$ SAT on control-point convex hulls, followed by a pure-GPU Bernstein-sampled curve distance optimization (100 points, $<$ 1 mm accuracy). This provides the primary real-time path, with legacy CPU-based L-BFGS-B/scipy retained only as an exception-recovery fallback, achieving a 53$\times$ to 146$\times$ speedup over traditional mesh-based collision checking.
4) **Causal Edge Dynamics & Compute Gating**: A mechanism that continuously updates relational edge latent vectors (e.g., SUPPORTED $\rightarrow$ FALLING). Crucially, we introduce compute gating, skipping edge evolution during MCTS planning to save $\sim$40 MB VRAM and $\sim$1 TFLOOP per cycle, enabling extreme edge efficiency.
5) **Uncertainty-Aware Graded Safety**: Control-point uncertainty is propagated into time-varying tubes and mapped to graded safety states (CERTIFIED_SAFE, PROBABILISTIC_SAFE, UNCERTAIN, UNSAFE) that alter speed, clearance, and execution policy.
6) **Unified Training & Autonomous Evolution**: A continuous-time integral loss function ($\mathcal{L}_{\text{JEPA-X}}$) with adaptive gradient-norm weighting (Phase B), coupled with Bayesian hyperparameter optimization and Elastic Weight Consolidation (EWC) [7] (Phase C) for safe, self-improvement without catastrophic forgetting.
7) **Extensive Edge-Deployment Validation**: Comprehensive benchmarks demonstrating real-time performance of 139 FPS (7.2 ms) on RTX 2070 and 62 FPS (16.2 ms) on Jetson Orin NX, with an 85–89% CERTIFIED_SAFE rate and near-linear $\mathcal{O}(N^{1.02-1.04})$ multi-object scaling, suggesting viability for resource-constrained robotic platforms.

The remainder of this paper is organized as follows: Section II reviews related work. Section III details the system architecture and mathematical formulation. Section IV presents the experimental setup and benchmark results. Section V discusses implications and limitations. Section VI concludes the paper.

II. Related Work

Our work intersects with latent-space world models, safe motion planning, continuous trajectory representation, and autonomous continual learning. We discuss the limitations of existing paradigms and highlight how JEPA-X bridges compact latent-space reasoning with mathematically rigorous physical safety.

A. Latent-Space Predictive Architectures

The shift from pixel-space to latent-space reasoning has been driven by the need to bypass the computational intractability of high-dimensional visual inputs. Architectures based on JEPA [1] and reconstruction-based models like DreamerV3 [3] have demonstrated remarkable capabilities in learning compact internal representations for planning.

JEPA-style models predict target representations from context rather than reconstructing every pixel. This is attractive for robotics because visually irrelevant detail need not dominate the learning objective. However, these systems predominantly rely on discrete autoregressive step prediction ($z_t \rightarrow z_{t+1}$), which inherently creates "temporal blind spots" in high-speed dynamic environments—even if states at t and $t+1$ are deemed safe, the continuous physical trajectory between them may intersect with dynamic obstacles.

Reconstruction-based models further require decoding latent states back to pixel space for reward computation or safety verification, incurring $\sim$100$\times$ computational overhead and introducing quantization artifacts that blur continuous geometric boundaries required for precise manipulation.

JEPA-X fundamentally departs from discrete autoregression by introducing the Latent Kinematic Path Predictor (LKPP), which parameterizes future trajectories as continuous Bernstein polynomials (Bézier curves) directly within a shared 128D latent manifold—eliminating temporal blind spots, bypassing pixel reconstruction, and enabling continuous-time physical reasoning without quantization artifacts.

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

B. Safe Motion Planning & Continuous-Time Safety Certification

Traditional safe motion planning frameworks, such as CHOMP [16] and STOMP [17], formulate trajectory generation as a continuous optimization problem in Cartesian space. While mathematically sound, these methods rely heavily on explicit 3D mesh rendering (URDF/Octomap) and external physics engines (e.g., MoveIt [4], FCL [5]). This introduces a severe "representation disconnect": high-level cognitive planning occurs in a low-dimensional latent space, while safety verification is relegated to explicit 3D Cartesian grids, necessitating continuous coordinate transformations and dense spatial sampling.

More recently, Control Barrier Functions (CBFs) [18] have emerged as the state-of-the-art for providing continuous-time, forward-invariance safety guarantees in robotics. However, synthesizing valid CBFs for high-dimensional, non-convex, or dynamically changing environments remains computationally intractable, often requiring conservative approximations or explicit, known dynamic models.

JEPA-X bridges this gap by embedding safety directly into the latent representation. By parameterizing trajectories as Bézier curves, we exploit their global convex hull property. This allows our Analytical Collision Module to evaluate 3D convex hulls of physical control points using the Separating Axis Theorem (SAT), achieving $\mathcal{O}(1)$ constant-time collision certification for any trajectory pair. This completely bypasses mesh rendering and the need for explicit CBF synthesis, while still providing rigorous, continuous-time geometric safety guarantees. For multi-object scenarios, spatial hashing and Bounding Volume Hierarchy (BVH) pruning reduce global complexity to near-linear $\mathcal{O}(N \log N)$, improving collision detection throughput by over two orders of magnitude compared to traditional physics engines.

C. Causal Reasoning, Uncertainty, & Autonomous Self-Improvement

Purely geometric planners cannot perform causal chain-reaction reasoning. JEPA-X elevates continuous trajectory planning by embedding it within a Structured Latent Scene Graph (SLSG), outputting dual-branch control points with strict anchor constraints to prevent "latent hallucinations." Our Causal Edge Dynamics Engine continuously updates relational edge latents along the predicted trajectory, enabling continuous physical causal chain reactions entirely within the latent manifold.

Furthermore, ensuring safety during autonomous learning is a fundamental challenge in Safe Reinforcement Learning [15]. While current embodied AI systems rely on static deployment and cannot adapt to out-of-distribution (OOD) scenarios without costly offline retraining, JEPA-X introduces a Self-Improving Autonomous Learning Loop (Phase C). Unlike methods that rely on complex uncertainty quantification via contracting flow models [19] (which incur high computational overhead), JEPA-X employs a lightweight, analytically tractable Gaussian uncertainty propagation through Bernstein basis functions. This produces time-varying uncertainty tubes that map directly to graded safety states (CERTIFIED_SAFE, PROBABILISTIC_SAFE, UNCERTAIN, UNSAFE).

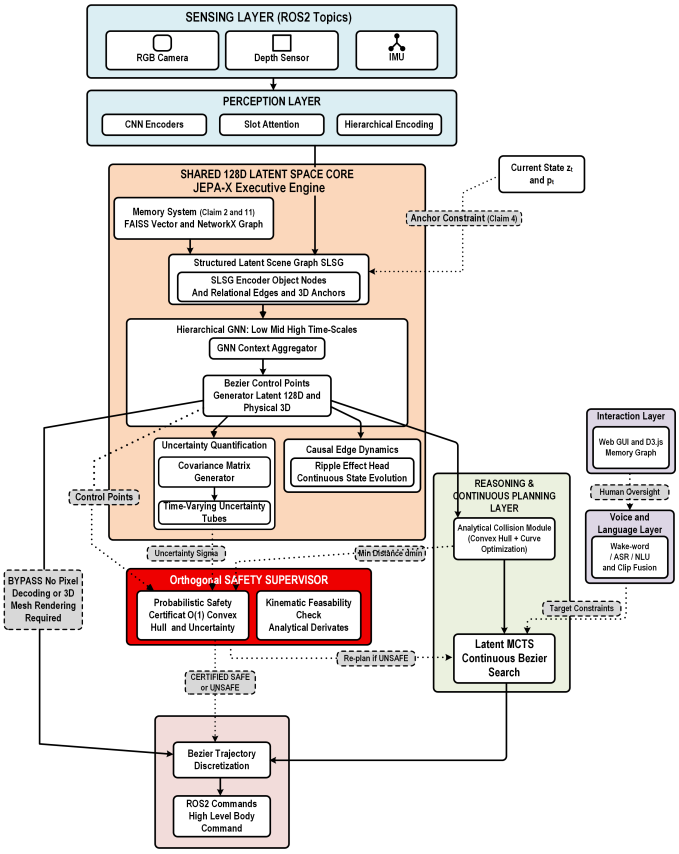

Fig. 1. JEPA-X system architecture. Multi-modal inputs encode to a 128D Structured Latent Scene Graph (SLSG). The Latent Kinematic Path Predictor (LKPP) generates continuous Bézier control points, evaluated by the Analytical Collision Module and coordinated by a Behavior Tree. Phase-C learning loop runs orthogonally in the background.

By autonomously collecting real-world trajectory experiences, triggering retraining based on data quality thresholds, and utilizing Bayesian Optimization [9] to tune the multi-objective unified continuous-time loss function ($\mathcal{L}_{\text{JEPA-X}}$), the system achieves continuous self-improvement. This evolution is protected by a strict safety validation gate ($\geq$5% loss reduction and safety scores $\geq$0.8) and Elastic Weight Consolidation (EWC) [7] regularization, ensuring safe exploration and learning without catastrophic forgetting or reliance on cloud-based offline retraining.

III. System Architecture and Methodology

Unlike prior latent-space world models relying on discrete autoregressive steps, JEPA-X operates entirely within a shared 128-dimensional latent manifold, utilizing continuous Bézier parameterization for trajectory prediction and analytical convex hull properties for $\mathcal{O}(1)$ mathematical safety certification (Fig. 1).

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

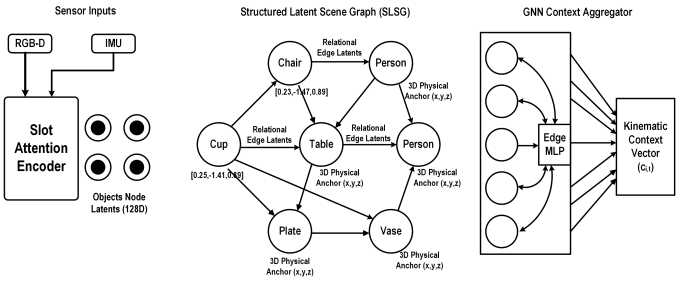

Fig. 2. Structured Latent Scene Graph (SLSG) and GNN Context Aggregation. Object nodes (128D latent + 3D physical anchor) are connected by relational edge latents encoding continuous physical constraints (e.g., SUPPORTED_BY). The GNN aggregates neighbor messages and the candidate action embedding to produce a rich kinematic context vector $\mathbf{c}_{i,t}$ for each object.

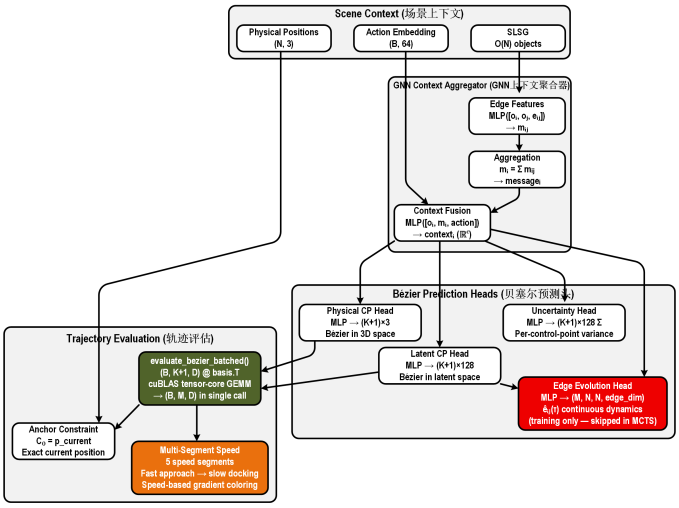

Fig. 3. LKPP Trajectory Prediction with Bézier Parameterization. Scene context (physical positions, action embedding, SLSG) flows through GNN Context Aggregator to generate dual-branch Bézier control points (latent 128D + physical 3D). The Edge Evolution Head (red, training-only) is skipped during MCTS for efficiency. Trajectory evaluation uses cuBLAS tensor-core GEMM for batched inference with anchor constraint enforcement.

A. Structured Latent Scene Graph (SLSG)

To enable object-centric reasoning and causal inference, we represent the environment as a Structured Latent Scene Graph $\mathcal{S}_t = \{\mathcal{O}_t, \mathcal{E}_t, \mathcal{P}_t\}$. Using iterative Slot Attention [6], multi-modal sensor observations are decomposed into N discrete object slots. Each object node $\mathbf{o}_{i,t} \in \mathbb{R}^{128}$ is augmented with a physical 3D anchor $\mathbf{p}_{i,t} \in \mathbb{R}^3$ from a position head. The robot is modeled as the Ego-Node $(i = 0)$, with 6-DOF pose encoded via latent visual odometry, unifying ego-motion and dynamic obstacle tracking (Fig. 2).

Relational edge latents $\mathbf{e}_{ij,t} \in \mathbb{R}^{d_e}$ encode continuous physical and semantic constraints (e.g., SUPPORTED_BY, NEAR). A Graph Neural Network (GNN) message-passing layer aggregates contextual constraints. The kinematic context vector $\mathbf{c}_{i,t}$ for object i is:

$$\mathbf{m}_{i,t} = \sum_{j \in \mathcal{N}(i)} \text{MLP}_{\text{edge}}\left([\mathbf{o}_{i,t}, \mathbf{o}_{j,t}, \mathbf{e}_{ij,t}]\right) \tag{1}$$

$$\mathbf{c}_{i,t} = \text{MLP}_{\text{fuse}}\left([\mathbf{o}_{i,t}, \mathbf{m}_{i,t}, \mathbf{a}_t]\right) \tag{2}$$

where $\mathbf{a}_t \in \mathbb{R}^{d_a}$ is the candidate action embedding. The explicit action term is essential: JEPA-X predicts what is expected to happen under a candidate robot action rather than merely extrapolating passive scene dynamics.

B. Latent Kinematic Path Predictor (LKPP) & Bézier Parameterization

The Latent Kinematic Path Predictor (LKPP) is the core engine of JEPA-X, transforming discrete scene representations into continuous, physically grounded trajectories. Unlike traditional discrete autoregressive predictors, the LKPP operates as a multi-head neural network that outputs parameterized Bézier curves. The complete architectural data flow is illustrated in Fig. 3.

1) Control Point Generation:: Given context $\mathbf{c}_{i,t}$, the LKPP outputs $(K+1)$ control points for latent and physical spaces:

$$\{C_{i,k}^{(z)}\}_{k=0}^{K} \in \mathbb{R}^{(K+1)\times 128} \tag{3}$$

$$\{C_{i,k}^{(p)}\}_{k=0}^{K} \in \mathbb{R}^{(K+1)\times 3} \tag{4}$$

To eliminate latent hallucinations, we enforce a strict anchor constraint:

$$C_{i,0}^{(p)} = \mathbf{p}_{i,t}, \quad C_{i,0}^{(z)} = \mathbf{o}_{i,t} \tag{5}$$

The continuous physical trajectory $\hat{\mathbf{p}}_i(\tau)$ and latent trajectory $\hat{\mathbf{z}}_i(\tau)$ over $\tau \in [0, \Delta T]$ use Bernstein basis polynomials defined as:

$$B_{k,K}\left(\frac{\tau}{\Delta T}\right) = \binom{K}{k}\left(\frac{\tau}{\Delta T}\right)^k \left(1 - \frac{\tau}{\Delta T}\right)^{K-k}. \tag{6}$$

The trajectories are then expressed as:

$$\hat{\mathbf{p}}_i(\tau) = \sum_{k=0}^{K} B_{k,K}\left(\frac{\tau}{\Delta T}\right) C_{i,k}^{(p)}, \tag{7}$$

$$\hat{\mathbf{z}}_i(\tau) = \sum_{k=0}^{K} B_{k,K}\left(\frac{\tau}{\Delta T}\right) C_{i,k}^{(z)}. \tag{8}$$

Since the derivative of a Bézier curve is a lower-order Bézier curve, velocity $\dot{\hat{\mathbf{p}}}_i(\tau)$ and acceleration are computed with zero mesh-rendering overhead.

2) Scene Context and GNN Aggregation: The LKPP does not operate on isolated objects. It receives a rich **Scene Context** comprising: (1) *Physical Positions* $(N, 3)$ from the SLSG physical anchors; (2) *Action Embeddings* $(B, 64)$ representing the candidate robot actions; and (3) the *SLSG* topology containing $O(N)$ objects and their relational edges.

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

To understand how a candidate action affects a specific object and its neighbors, the LKPP employs a **GNN Context Aggregator**. First, an *Edge Features MLP* processes the concatenation of source node, target node, and edge latent vectors $[o_i, o_j, e_{ij}]$ to generate pairwise messages m_{ij}. These messages are aggregated via summation ($m_i = \sum m_{ij}$) to capture the relational context. Finally, a *Context Fusion MLP* integrates the object's intrinsic state, the aggregated relational message, and the action embedding to produce a high-dimensional **Kinematic Context Vector** $\mathbf{c}_{i,t} \in \mathbb{R}^{d_c}$ for each object.

3) Dual-Branch Bézier Prediction Heads: Conditioned on the kinematic context vector $\mathbf{c}_{i,t}$, the LKPP branches into parallel Multi-Layer Perceptron (MLP) heads to generate the continuous trajectory parameters:

- **Physical CP Head:** Maps $\mathbf{c}_{i,t}$ to $(K+1) \times 3$ physical 3D control points. These points define the actual spatial path of the object and are used directly for collision certification.
- **Latent CP Head:** Maps $\mathbf{c}_{i,t}$ to $(K+1) \times 128$ latent semantic control points. These points govern the evolution of the object's semantic features within the 128D manifold.

4) Interpretation of the 128D Latent Space: The shared latent manifold $\mathbb{R}^{128}$ provides a compact, distributed representation that jointly encodes the semantic, geometric, appearance, and dynamic properties of the observed scene. Consistent with modern self-supervised representation learning, JEPA-X does not assign specific semantic meaning to individual dimensions; rather, information is encoded holistically across the entire latent vector.

The dimensionality of 128 was selected as a deliberate design trade-off between representational capacity, computational efficiency, and the strict memory bandwidth constraints of edge deployment. While lower-dimensional spaces reduce the memory footprint, they risk bottlenecking the expressive capacity required for complex dynamic environments. Conversely, higher-dimensional representations increase the computational cost of GNN message passing and MLP inference, often yielding diminishing returns in downstream planning performance without proportional gains in accuracy.

From a system perspective, the selected dimensionality provides several critical advantages:

- **Hardware-Aligned Edge Deployment:** A 128-dimensional vector aligns optimally with GPU memory alignment requirements and tensor-core utilization, enabling low-latency inference on embedded platforms like the Jetson Orin NX without saturating memory bandwidth.
- **Unified Multi-Task Representation:** The shared latent space simultaneously supports perception, trajectory prediction, memory retrieval, uncertainty estimation, and planning without requiring multiple task-specific feature embeddings.
- **Efficient Memory Retrieval:** The fixed-dimensional latent representation is directly compatible with vector databases such as FAISS, enabling efficient nearest-neighbor retrieval for long-term memory and experience reuse.
- **Dual-Branch Trajectory Parameterization:** Enables the parallel generation of $(K+1) \times 128$ latent and $(K+1) \times 3$ physical control points without incurring excessive memory penalties during batched MCTS evaluation.

Ultimately, the 128D latent space is motivated by the holistic system architecture and real-time constraints. It provides a robust information bottleneck that prevents overfitting while maintaining sufficient capacity for continuous-time physical and semantic reasoning.

5) Uncertainty and Edge Evolution Heads: To support uncertainty-aware planning, probabilistic safety assessment, and continuous causal reasoning, the Latent Kinematic Path Predictor (LKPP) incorporates two auxiliary prediction heads in addition to the continuous trajectory predictor.

- **Uncertainty Head:** For every predicted Bernstein control point, the network estimates the corresponding uncertainty of the physical trajectory. Let the k-th control point of object i be associated with the spatial covariance matrix

$$\Sigma_{i,k}^{(p)} \in \mathbb{R}^{3\times 3},$$

which models the predicted 3D positional uncertainty of that control point.

Assuming that the uncertainty associated with each Bernstein control point follows an independent Gaussian distribution, the covariance of the continuous trajectory can be approximated by propagating the control-point covariances through the squared Bernstein basis functions $B_{k,K}^2\left(\frac{\tau}{\Delta T}\right)$:

$$\Sigma_i^{(p)}(\tau) \approx \sum_{k=0}^{K} B_{k,K}^2\left(\frac{\tau}{\Delta T}\right) \Sigma_{i,k}^{(p)}, \tag{9}$$

where the approximation assumes mutually independent Gaussian control-point uncertainties and neglects cross-covariance terms between different control points.

The propagated covariance defines a continuous, time-varying uncertainty envelope surrounding the predicted trajectory. This uncertainty tube is subsequently used by the safety module to adapt collision margins, execution speed, and decision policy according to the estimated confidence of the prediction (Section III-G).

The current implementation assumes independent control-point uncertainties for computational efficiency. Extension to correlated covariance propagation remains future work.

- **Edge Evolution Head:** The Edge Evolution Head predicts the continuous temporal evolution of relational edges $\hat{e}_{ij}(\tau)$, capturing causal state transitions between entities (e.g., `SUPPORTED`→`FALLING`, `CLEAR_PATH`→`POTENTIAL_COLLISION`, or `VISIBLE`→`OCCLUDED`).

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

As described in Section III-C, this branch is disabled during MCTS candidate evaluation through compute gating. Because the predicted edge dynamics do not directly influence the immediate geometric scoring of trajectory candidates, this head is activated only during Phase-B training and causal scene-model updates. This gating reduces GPU memory consumption by approximately 40 MB and eliminates approximately one TFLOP of unnecessary computation during online planning.

6) Role of the Latent Trajectory: JEPA-X predicts both a latent trajectory and its corresponding physical trajectory because the two representations serve complementary purposes. The physical trajectory provides geometrically interpretable information required for collision checking, safety certification, and robot execution. In contrast, the latent trajectory preserves high-level semantic information that is not directly observable in physical coordinates, including object identity, interaction context, motion intent, and scene dynamics.

During planning, the latent trajectory enables reasoning over future semantic scene evolution, while the physical trajectory guarantees physically executable motion and geometric safety. This dual-representation design allows JEPA-X to jointly optimize semantic prediction and continuous trajectory generation within a unified latent world model.

7) Trajectory Evaluation: The generated control points are evaluated using a highly optimized **Trajectory Evaluation** pipeline. To ensure physical realism, the system enforces a strict **Anchor Constraint** ($C_0 = p_{\text{current}}$), forcing the zero-order control point to exactly match the object's current physical state, thereby eliminating latent-space hallucinations. Furthermore, the trajectory is structured as a **Multi-Segment Speed** profile (e.g., 5 speed segments transitioning from fast approach to slow docking), enabling speed-based gradient coloring and smooth execution.

C. GPU-Specific Optimizations

To achieve real-time performance on edge hardware, we implement several GPU-specific optimizations:

cuBLAS Tensor-Core GEMM: For computational efficiency, the Bézier evaluation is not performed sequentially. Instead, the LKPP utilizes an `evaluate_bezier_batched()` function. By formulating the Bézier basis evaluation as a batched matrix multiplication $(C^T \otimes B^T)^T$, the computation is dispatched directly to **cuBLAS tensor-core GEMM** kernels. This allows the LKPP to evaluate the entire (B, M, D) tensor for all objects and MCTS candidates in a single GPU call, achieving massive throughput gains over standard `torch.einsum` implementations: 20-30% speedup.

Bernstein Basis Caching: We cache Bernstein basis matrices keyed on $(\text{degree}, M, t_{\min}, t_{\max}, \text{device})$, bounded to 64 entries. This eliminates redundant power computations and binomial coefficient calculations in the hot path, providing 18% LKPP speedup Fig. 4.

.

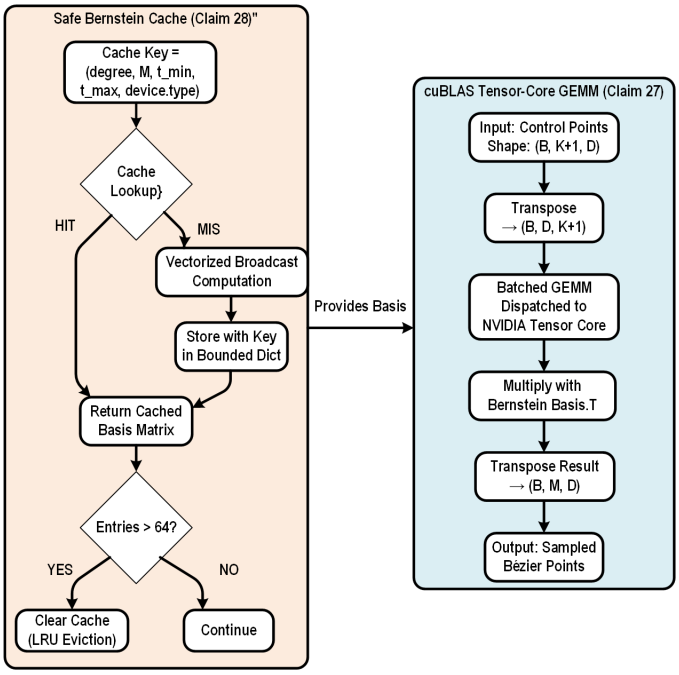

Fig. 4. Safe Bernstein Cache (Claim 28) with LRU eviction (64-entry bound) and cuBLAS Tensor-Core GEMM pipeline (Claim 27). Control points $(B, K{+}1, D)$ are transposed and multiplied with cached Bernstein basis via batched GEMM, dispatching directly to NVIDIA Tensor Cores for 20–30% speedup over `torch.einsum`.

Compute Gating: The Edge Evolution Head predicts continuous relational dynamics and produces $(B, N, N, M, 2L + E)$ tensors (∼40 MB) through a 2-layer MLP (∼1 TFLOP). During MCTS candidate evaluation, however, neither the analytical collision checker nor the MCTS scoring function consumes the predicted edge-evolution output; both operate directly on the predicted physical trajectories, safety certificates, kinematic constraints, and associated candidate scores. Consequently, evaluating the Edge Evolution Head for every MCTS candidate would introduce substantial computation and memory traffic without affecting action selection. JEPA-X therefore disables this branch during the MCTS hot path (`compute_edge_evolution=False`) while retaining it during training and causal scene-model updates, providing a 2.25× LKPP batched speedup.

D. Unified Ego-Motion and Dynamic-Object Continuous Prediction

JEPA-X applies the same continuous trajectory representation to the Ego-Node and to dynamic obstacles. For Ego trajectory $\mathbf{p}_E(\tau)$ and object trajectory $\mathbf{p}_i(\tau)$, define the relative displacement:

$$\mathbf{r}_i(\tau) = \mathbf{p}_E(\tau) - \mathbf{p}_i(\tau) \tag{10}$$

$$d_i(\tau) = \|\mathbf{r}_i(\tau)\|_2 \tag{11}$$

The minimum separation over the horizon is:

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

$$d_{\min,i} = \min_{\tau \in [0,\Delta T]} d_i(\tau) \tag{12}$$

A time-to-collision quantity can be defined as:

$$\mathrm{TTC}_i = \inf\{\tau : d_i(\tau) \le d_{\text{safe}}\} \tag{13}$$

This formulation distinguishes objects that are spatially close but temporally separated from trajectories that actually enter the safety envelope at the same future time.

E. Speed-Aware Segmented Continuous Trajectory

For physical execution, position alone is insufficient. A robot must also regulate its velocity, acceleration, and confidence throughout the motion. JEPA-X therefore represents an executable trajectory as

$$T = \{p(\tau), v(\tau), a(\tau), \Sigma(\tau)\}, \tag{14}$$

where $p(\tau)$ denotes the continuous spatial trajectory, $v(\tau)$ and $a(\tau)$ are the corresponding velocity and acceleration profiles, and $\Sigma(\tau)$ represents the propagated execution uncertainty.

Unlike conventional trajectory generators that assume a single velocity profile, JEPA-X partitions the trajectory into multiple executable segments:

$$T = \{T_1, T_2, \ldots, T_M\},$$

where each segment possesses its own duration, polynomial degree, and desired speed profile. This enables the robot to move rapidly through low-risk regions while automatically slowing near goals, humans, obstacles, narrow passages, or regions of elevated uncertainty.

Let segment m be represented by a Bernstein polynomial of degree K_m, control points $\{C_{m,0}, \ldots, C_{m,K_m}\}$, and execution duration ΔT_m. Because adjacent segments may have different durations, continuity must be enforced with respect to physical time rather than the normalized Bernstein parameter.

a) Position continuity (C^0).: The endpoint of one segment must coincide with the beginning of the next:

$$C_{m,K_m} = C_{m+1,0}. \tag{15}$$

b) Velocity continuity (C^1).: To avoid discontinuous velocity commands, the physical velocities at the segment boundary are constrained as

$$\begin{aligned} &\frac{K_m}{\Delta T_m}\left(C_{m,K_m} - C_{m,K_m-1}\right) \\ &\quad = \frac{K_{m+1}}{\Delta T_{m+1}}\left(C_{m+1,1} - C_{m+1,0}\right). \end{aligned} \tag{16}$$

c) Acceleration continuity (C^2).: Smooth dynamic execution further requires acceleration continuity:

$$\begin{aligned} &\frac{K_m(K_m-1)}{\Delta T_m^2}\left(C_{m,K_m} - 2C_{m,K_m-1} + C_{m,K_m-2}\right) \\ &\quad = \frac{K_{m+1}(K_{m+1}-1)}{\Delta T_{m+1}^2}\left(C_{m+1,2} - 2C_{m+1,1} + C_{m+1,0}\right). \end{aligned} \tag{17}$$

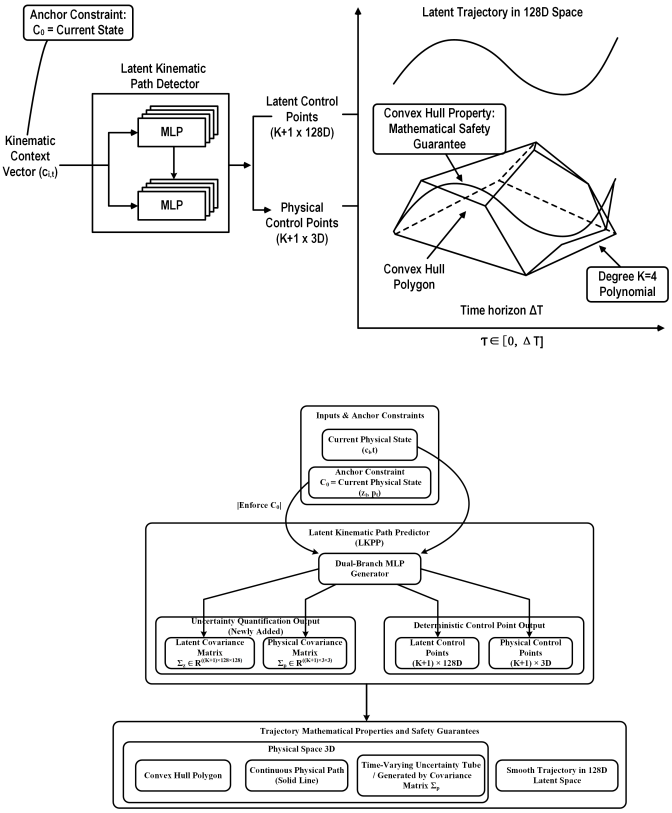

Fig. 5. LKPP with dual-branch control points (latent 128D + physical 3D) and anchor constraint ($C_0 = p_{\text{current}}$). The Bézier convex-hull property guarantees that the continuous trajectory is contained within the convex hull of its control points, enabling a bounded-cost geometric separation test per trajectory pair for fixed polynomial degree..

These constraints guarantee C^0, C^1, and C^2 continuity in physical time, ensuring smooth transitions between adjacent trajectory segments despite differing speed profiles. Ultimately, this converts the predicted path from a purely geometric curve into a temporally executable motion profile, allowing direct execution by a physical robot without introducing velocity jumps or mechanical shocks at segment boundaries.

1) Multi-Timescale Prediction: The LKPP operates concurrently at three resolutions(Table I):

TABLE I
HIERARCHICAL PLANNING PARAMETERS BY LEVEL

Level	**Time Horizon** (ΔT)	**Polynomial Degree** (K)	**Planning Category**
Low-level reactive avoidance	0.5 s	6	Reactive
Mid-level tactical interaction	5.0 s	4	Tactical
High-level topological navigation	60.0 s	2	Strategic

F. Pure-GPU Analytical Collision Certification & Multi-Object Pruning

Exploiting the Convex Hull Property of Bernstein polynomials, every continuous trajectory is entirely contained within

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

the convex hull of its physical control points. For object i, let

$$\mathcal{H}_i = \text{ConvHull}\left(\{C_{i,k}^{(p)}\}_{k=0}^{K}\right), \tag{18}$$

where $\mathcal{H}_i$ denotes the convex hull enclosing all Bernstein control points of the predicted physical trajectory.

This property forms the mathematical foundation of the proposed collision-certification framework.

Theorem III-F.1 (Convex-Hull Safety Certificate)**.** *Let $\hat{\mathbf{p}}_A(\tau)$ and $\hat{\mathbf{p}}_B(\tau)$, $\tau \in [0, \Delta T]$, be two Bernstein trajectories with convex hulls $\mathcal{H}_A$ and $\mathcal{H}_B$ constructed from their respective control points. If*

$$\mathcal{H}_A \cap \mathcal{H}_B = \varnothing, \tag{19}$$

then

$$\hat{\mathbf{p}}_A(\tau) \neq \hat{\mathbf{p}}_B(\tau), \quad \forall \tau \in [0, \Delta T]. \tag{20}$$

Equivalently,

$$\hat{\mathbf{p}}_A([0, \Delta T]) \cap \hat{\mathbf{p}}_B([0, \Delta T]) = \varnothing. \tag{21}$$

Therefore, the predicted trajectories remain disjoint throughout the planning horizon under the modeled geometric assumptions.

Proof. A fundamental property of Bernstein polynomials states that every point on a Bernstein curve lies inside the convex hull of its control points. Therefore,

$$\hat{\mathbf{p}}_A([0, \Delta T]) \subseteq \mathcal{H}_A, \quad \hat{\mathbf{p}}_B([0, \Delta T]) \subseteq \mathcal{H}_B.$$

If $\mathcal{H}_A \cap \mathcal{H}_B = \varnothing$, the two convex hulls are disjoint. Consequently, the two trajectory sets cannot intersect, establishing Eq. (21). □

The theorem provides a conservative geometric safety certificate. If the convex hulls are disjoint, the predicted trajectories are guaranteed to remain separated under the adopted occupancy model. Conversely, overlapping convex hulls do not necessarily imply collision; they indicate only that the conservative certificate is unavailable and that a more precise geometric evaluation is required.

Theorem 1 therefore establishes the theoretical foundation of the proposed collision-certification framework. Building upon this result, JEPA-X employs a two-stage collision-assessment pipeline. The first stage performs a conservative convex-hull separation test that can immediately certify safe trajectory pairs. Only when this certificate is unavailable does the second stage invoke a GPU-based Bernstein-sampled minimum-distance refinement to obtain a more accurate geometric assessment.

1) Stage 1: Conservative Convex-Hull Separation Certificate: For any pair (Ego-Node A and obstacle B), compute 3D convex hulls $\mathcal{H}_A, \mathcal{H}_B$ and apply the Separating Axis Theorem (SAT). If $\mathcal{H}_A \cap \mathcal{H}_B = \emptyset$, the trajectories are mathematically collision-free. The system issues a CERTIFIED_SAFE certificate.

Important Note: The convex-hull test is intentionally **conservative**. Hull overlap does **not** imply collision; it means only that the fast sufficient certificate is unavailable and a more precise distance evaluation is required. Conversely, a disjoint conservative envelope provides a strong geometric separation result, but only under the correctness of the modeled body geometry, predicted states, and uncertainty bounds.

2) Stage 2: Pure-GPU Bernstein-Sampled Distance Optimization: If hulls overlap, V2.0 invokes a pure-GPU Bernstein-sampled Bézier curve distance computation (evaluating 100 points via cuBLAS tensor-core GEMM). This completely eliminates legacy CPU-based scipy/L-BFGS-B bottlenecks and CPU↔GPU transfer overhead, achieving < 1 mm accuracy with massive throughput gains.

For multi-object scenes, spatial hashing and Bounding Volume Hierarchy (BVH) pruning eliminate unnecessary pairwise checks, maintaining near-linear $\mathcal{O}(N^{1.02-1.04})$ scaling.

Compute Gating & Spatial Pruning: To maximize edge efficiency, the `edge_evolution` computation is **dynamically gated**—skipped during MCTS planning (saving ~ 40 MB VRAM and ~ 1 TFLOP) and enabled *only* during Phase B training. For multi-object scenes, two-stage Spatial Hashing and BVH pruning reduces global complexity to near-linear $\mathcal{O}(N^{1.02-1.04})$ empirical scaling (Fig. 6).

G. Physical Occupancy Representation

A predicted reference trajectory alone does not fully describe the occupied space of a physical robot. JEPA-X therefore associates each trajectory with a conservative geometric envelope. In the present implementation, entities are represented using simplified primitives (point-radius, sphere, capsule, or axis-aligned bounding box). The occupied set of entity i at time τ is:

$$\mathcal{O}_i(\tau) = \hat{\mathbf{p}}_i(\tau) \oplus \mathcal{B}_i \oplus \mathcal{U}_i(\tau), \tag{22}$$

where $\hat{\mathbf{p}}_i(\tau)$ is the predicted reference trajectory, $\mathcal{B}_i$ is the conservative body envelope, and $\mathcal{U}_i(\tau)$ is the uncertainty expansion. Collision assessment is performed between these expanded occupied sets rather than between mathematical trajectory points alone.

In our experiments, $\mathcal{B}_i$ is instantiated as a sphere of radius $r = 0.5\,\text{m}$ for the humanoid robot, and $\mathcal{U}_i(\tau)$ is computed as a 3σ confidence ellipsoid derived from $\Sigma(\tau)$. Collision checking uses continuous-time separation tests between these time-varying occupied sets.

a) Future extension: Dynamic articulated occupancy.: The current implementation uses rigid, fixed-size envelopes. A more physically accurate representation would model the humanoid as a *dynamic volume* that evolves with articulated body motion. This could be realized through:

- **Multi-sphere decomposition:** Approximating the robot geometry as an agglomeration of spheres or capsules attached to body segments (torso, limbs, end-effectors), whose positions evolve according to predicted joint configurations.
- **Swept-volume prediction:** Computing the time-varying union $\mathcal{B}_i(\tau) = \bigcup_{k=1}^{N_{\text{body}}} \mathcal{S}_k(\mathbf{q}(\tau))$, where each primitive $\mathcal{S}_k$ is positioned by the predicted joint state $\mathbf{q}(\tau)$.

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

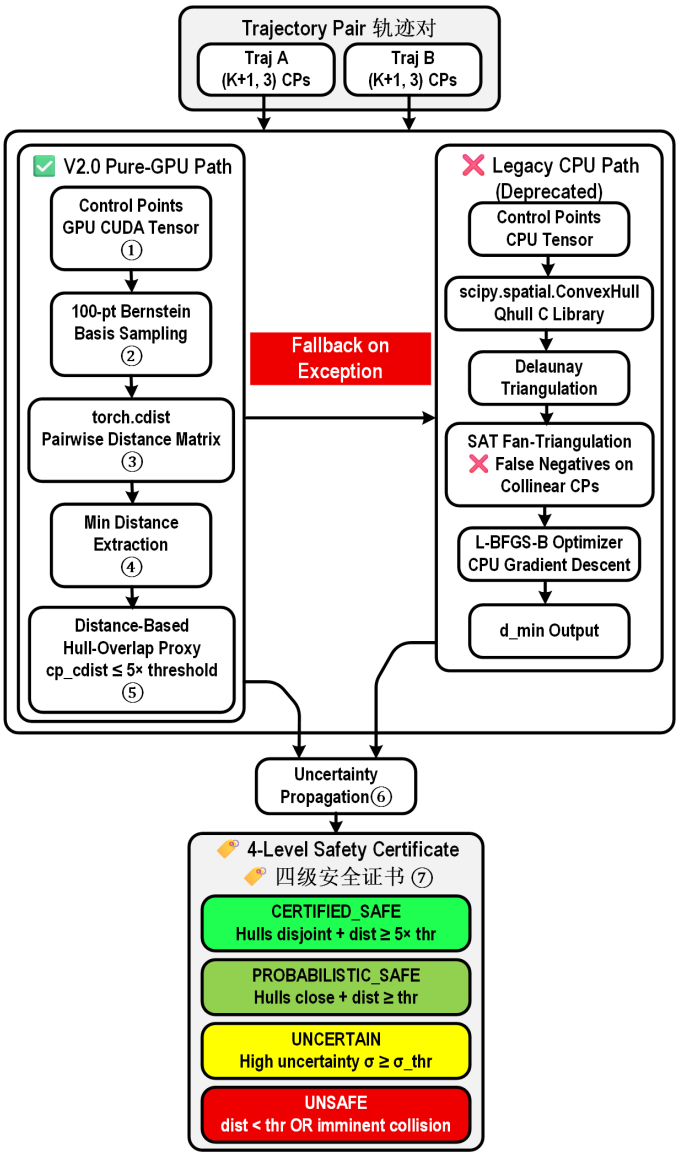

Fig. 6. Pure-GPU analytical collision certification pipeline. Candidate Bézier trajectory pairs are evaluated through GPU control-point pre-filtering and Bernstein-sampled curve-distance refinement. Conservative convex-hull separation provides direct CERTIFIED_SAFE decisions, while uncertainty and distance thresholds produce graded CERTIFIED_SAFE, PROBABILISTIC_SAFE, UNCERTAIN, and UNSAFE states. CPU/SciPy optimization is retained only as an exception-recovery fallback and is not part of the normal real-time planning path.

- **Configuration-space uncertainty:** Propagating joint-level uncertainty $\Sigma_{\mathbf{q}}(\tau)$ through the kinematic chain to obtain time-varying occupancy bounds.

Such a representation would enable *true* collision avoidance by accounting for limb swings, arm extensions, and foot placements, rather than relying on conservative over-approximations. Integrating this with JEPA-X's uncertainty-aware prediction framework constitutes a primary direction for future work.

H. Causal Edge Dynamics & Uncertainty Propagation

1) Causal Ripple Effect: To simulate continuous physical chain reactions (e.g., a humanoid navigating a crowded environment when a running child suddenly crosses its path), the Edge Dynamics Head predicts the continuous evolution of the relational edge latent $\hat{\mathbf{e}}_{ij}(\tau)$. If the analytical distance $d_{\min}$ between the humanoid's predicted trajectory and the child's predicted path falls below a critical threshold, the edge state continuously transitions from CLEAR_PATH $\rightarrow$ POTENTIAL_COLLISION $\rightarrow$ IMMINENT_DANGER. The LKPP then automatically recomputes the humanoid's downstream trajectory conditioned on the updated edge latent.

$$\hat{\mathbf{e}}_{ij}(\tau) = f_{\text{edge}}\left(\mathbf{e}_{ij}, \tau, \hat{\mathbf{z}}_i(\tau), \hat{\mathbf{z}}_j(\tau), d_{ij}(\tau)\right) \tag{23}$$

A significant relation change can trigger downstream re-prediction:

$$\text{Trajectory} \rightarrow \text{Relation change} \rightarrow \text{Predicted consequence} \rightarrow \text{Re-prediction}$$

2) Uncertainty Propagation and Graded Safety Certificates: Each physical Bernstein control point is associated with a Gaussian covariance matrix $\Sigma_{c,i,k}$. Assuming mutually independent Gaussian control-point uncertainties [**?**], the trajectory-level covariance is approximated by propagating the control-point covariances through the Bernstein basis functions:

$$\Sigma_{\hat{\mathbf{p}}}(\tau) \approx \sum_{k=0}^{K} B_{k,K}^2 \left(\frac{\tau}{\Delta T}\right) \Sigma_{c,i,k}. \tag{24}$$

JEPA-X separates deterministic geometric certification from probabilistic safety assessment. The resulting execution state is classified into four levels (Table II):

TABLE II
GRADED SAFETY STATES

Safety State	Interpretation
CERTIFIED_SAFE	Conservative geometric envelopes are analytically separated under the modeled bounds.
PROBABILISTIC_SAFE	No deterministic certificate is available, but the uncertainty-aware collision probability satisfies the configured acceptance criterion.
UNCERTAIN	Model/perception uncertainty is too high for normal autonomous execution.
UNSAFE	Predicted motion violates the configured collision or kinematic safety margin.

For a Gaussian approximation, the collision probability is estimated as:

$$P(\text{safe}) = 1 - \Phi\left(\frac{d_{\text{thresh}} - d_{\min}}{\sigma_{\text{total}}}\right) \tag{25}$$

where Φ is the standard normal CDF. This probability is **not used as a substitute** for the deterministic convex-separation certificate.

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

I. Uncertainty-Aware Behavioral Adaptation

Uncertainty modifies robot behavior rather than remaining a passive diagnostic value. A simple policy is:

$$v_{\max}^{\text{adaptive}} = v_{\max} \cdot g(\sigma) \tag{26}$$

$$d_{\text{safe}}^{\text{adaptive}} = d_0 + \alpha\sigma \tag{27}$$

where $g(\sigma)$ decreases as uncertainty increases. In low-uncertainty states the robot may use nominal velocity. At moderate uncertainty it slows down and increases clearance. At high uncertainty it may stop, re-perceive, or request replanning instead of executing a high-risk action.

J. Runtime Kinematic Feasibility

Because derivatives of Bézier curves are themselves Bézier curves of lower degree, velocity and acceleration can be computed analytically. During training, physically implausible predictions are penalized. During runtime, candidate trajectories are explicitly rejected if they violate class- or robot-specific limits:

$$\mathcal{F}(T) = \mathbf{1}\left[\max_{\tau} \|\dot{\hat{\mathbf{p}}}(\tau)\| \le v_{\max} \wedge \max_{\tau} \|\ddot{\hat{\mathbf{p}}}(\tau)\| \le a_{\max}\right] \tag{28}$$

This provides two layers of protection: the model is trained toward feasible dynamics, while the online planner does not rely solely on learned behavior to enforce hard kinematic limits. Kinematic penalties are class-specific: 0.5 m/s for cup, 2.0 m/s for human.

K. Latent MCTS & Behavior Tree Coordination

Latent Monte Carlo Tree Search (MCTS): Operates directly in continuous Bézier control point space using UCB1 selection and Gaussian perturbation. GPU-batched tensor evaluation (`forward_batched`) assesses up to 35 candidate trajectories in a single forward pass, enabling real-time global optimization. Candidate trajectories are evaluated using task score, goal progress, kinematic feasibility, collision status, and uncertainty-aware safety.

Behavior Tree State Machine: To resolve the "frequency disconnect" between the 1Hz Latent MCTS, 10Hz LKPP, and 100Hz low-level control, we deploy a non-blocking Behavior Tree as the execution arbiter. It manages explicit states (NAVIGATING, SCANNING, WAITING_FOR_HUMAN, EXECUTING_TRAJECTORY). During state transitions, the Behavior Tree enforces a smooth trapezoidal deceleration curve, preventing mechanical shocks and deadlocks.

Compute Gating: During MCTS planning, edge evolution is skipped to save $\sim$40 MB VRAM and $\sim$1 TFLOOP per cycle, enabling extreme edge efficiency.

L. Unified Continuous-Time Training Objective (Phase B)

To ensure the LKPP learns trajectories that are semantically accurate, physically exact, kinematically feasible, and causally consistent, we introduce the Continuous Latent Trajectory Consistency Loss $\mathcal{L}_{\text{JEPA-X}}$. Formulated as a continuous-time integral over the horizon ΔT:

$$\begin{aligned}\mathcal{L}_{\text{JEPA-X}} = &\underbrace{\int_0^{\Delta T} \|\hat{\mathbf{z}}_i(\tau) - \mathbf{z}_i(t+\tau)\|^2 d\tau}_{\mathcal{L}_{\text{latent}}} \\ &+ \lambda_1 \underbrace{\int_0^{\Delta T} \|\hat{\mathbf{p}}_i(\tau) - \mathbf{p}_i(t+\tau)\|^2 d\tau}_{\mathcal{L}_{\text{physical}}} \\ &+ \lambda_2 \underbrace{\int_0^{\Delta T} \max\left(0, \|\dot{\hat{\mathbf{p}}}_i(\tau)\|_2 - \nu_{\max}^{(\text{class})}\right) d\tau}_{\mathcal{L}_{\text{kinematic}}} \\ &+ \lambda_3 \underbrace{\int_0^{\Delta T} \|\hat{\mathbf{e}}_{ij}(\tau) - \mathbf{e}_{ij}(t+\tau)\|^2 d\tau}_{\mathcal{L}_{\text{edge}}}\end{aligned} \tag{29}$$

$\mathcal{L}_{\text{kinematic}}$ penalizes trajectories where the Bézier derivative exceeds class-specific velocity limits $v_{\max}$ (0.5 m/s for cup, 2.0 m/s for human). Integrals are discretized using the Trapezoidal Rule with $N = \max(50, \Delta T \times 100)$ uniform points. **Adaptive Loss Weighting** dynamically adjusts $\lambda_1, \lambda_2, \lambda_3$ based on inverse ratios of historical gradient norms to prevent gradient conflict.

M. Self-Improving Autonomous Learning Loop (Phase C)

As illustrated in Fig. 7, unlike static deployments, JEPA-X features an autonomous evolution loop. During operation, the system continuously collects TrajectoryExperience data. When thresholds are met ($\geq$1000 trajectories, $\geq$10 scene types), Autonomous Trainer triggers offline retraining:

1) **Bayesian Hyperparameter Optimization**: Gaussian Process with Expected Improvement (EI) [9] acquisition function automatically searches the continuous space to optimize the loss weights $(\lambda_1, \lambda_2, \lambda_3)$.
2) **Catastrophic Forgetting Prevention**: Elastic Weight Consolidation (EWC) [7] is applied to compute the Fisher Information matrix, regularizing the update of critical parameters.
3) **Safety-Gated Deployment**: The newly trained model is only hot-deployed if it passes a strict validation gate: a loss reduction of $\geq$5% and safety score $\geq$0.8. Ensures continuous adaptation to OOD environments without safety regression.

N. Joint-Embedding Predictive Training Mechanism

While the previous sections have detailed the action-conditioned architecture, a complete joint-embedding predictive framework requires an explicit training mechanism. JEPA-X follows the joint-embedding predictive architecture principle [1] but extends it to continuous-time trajectory prediction with action conditioning.

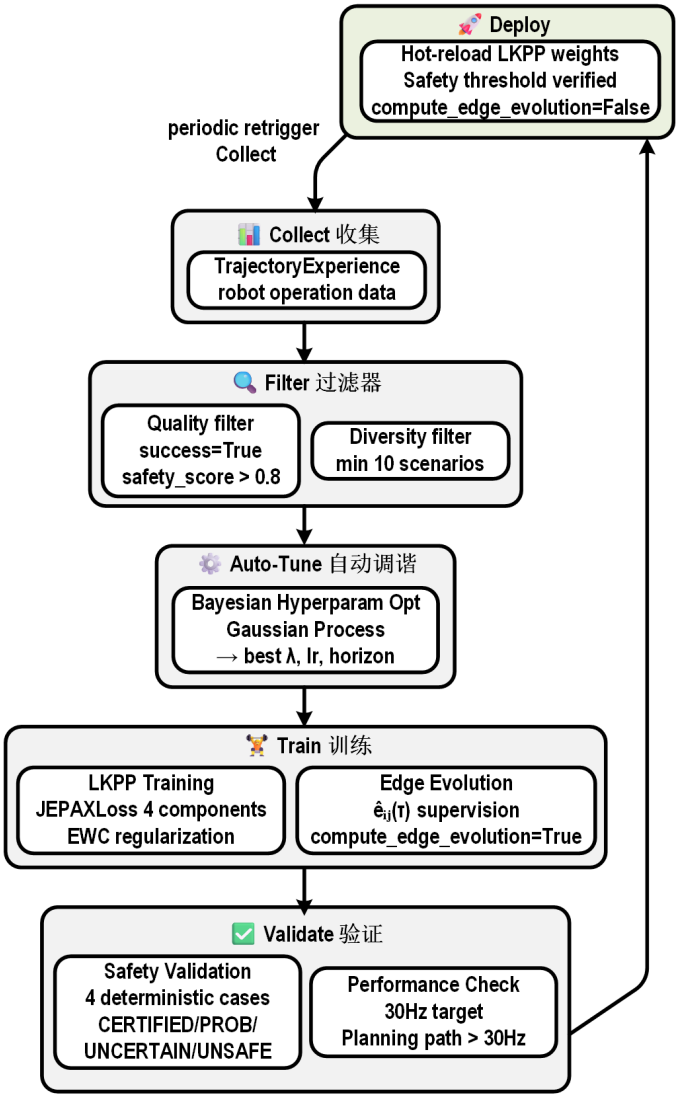

Fig. 7. JEPA-X Self-Improving Autonomous Learning Loop (Phase C). During deployment, successful robot trajectory experiences are collected and filtered for quality and scenario diversity. Bayesian hyperparameter optimization tunes the training configuration, while LKPP and causal edge dynamics are retrained with EWC regularization to mitigate catastrophic forgetting. Updated models are deployed only after passing safety and real-time performance validation, forming a closed-loop process for controlled autonomous improvement.

1) Predictive Objective: Let $\mathbf{x}_{\text{context}}$ denote the multimodal sensor input at time t, and $\mathbf{x}_{\text{future}}$ denote the corresponding input at future time $t+\tau$ for $\tau \in [0, \Delta T]$. We define two encoders:

$$\mathbf{z}_c = f_\theta(\mathbf{x}_{\text{context}}) \tag{30}$$

$$\mathbf{z}_\tau = f_{\bar{\theta}}(\mathbf{x}_{\text{future}}) \tag{31}$$

where f_θ is the **online encoder** (context encoder) and $f_{\bar{\theta}}$ is the **target encoder** (future encoder). The target encoder parameters $\bar{\theta}$ are updated via exponential moving average (EMA) of the online parameters:

$$\bar{\theta} \leftarrow \mu\bar{\theta} + (1-\mu)\theta \tag{32}$$

with momentum coefficient $\mu = 0.995$, following the self-supervised learning literature [2]. This EMA update ensures stable target representations and prevents representational collapse.

2) Action-Conditioned Future Prediction: The Latent Kinematic Path Predictor (LKPP) g_ϕ takes the context representation $\mathbf{z}_c$, the candidate action embedding $\mathbf{a}_t$, and the continuous time parameter τ, and predicts the future latent trajectory:

$$\hat{\mathbf{z}}_\tau = g_\phi(\mathbf{z}_c, \mathbf{a}_t, \tau) \tag{33}$$

The action conditioning is critical: g_ϕ predicts what will happen **under** the candidate action, not merely what might happen passively.

3) Joint-Embedding Predictive Loss: The training objective maximizes agreement between the predicted latent trajectory $\hat{\mathbf{z}}_\tau$ and the target representation $\mathbf{z}_\tau$ of the actual future:

$$\mathcal{L}_{\text{JEPA}} = \mathbb{E}_{\mathbf{x}_{\text{context}}, \mathbf{a}_t, \mathbf{x}_{\text{future}}} \left[\int_0^{\Delta T} \left\| \hat{\mathbf{z}}_\tau - \mathbf{z}_\tau \right\|^2 d\tau \right] \tag{34}$$

Unlike the original JEPA [1] which predicts a single future representation, JEPA-X predicts a **continuous trajectory** of representations across the entire horizon ΔT.

4) Stop-Gradient and Collapse Prevention: A critical mechanism is the **stop-gradient** operation: gradients from the prediction loss $\mathcal{L}_{\text{JEPA}}$ are **not** propagated into the target encoder $f_{\bar{\theta}}$:

$$\nabla_{\bar{\theta}} \mathcal{L}_{\text{JEPA}} = 0 \tag{35}$$

Combined with the EMA update in Eq. (32), this prevents representational collapse.

5) Relation to the Full Training Objective: The joint-embedding loss $\mathcal{L}_{\text{JEPA}}$ is combined with the continuous-time consistency losses from Section III.K to form the complete objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{JEPA}} + \mathcal{L}_{\text{JEPA-X}} \tag{36}$$

where $\mathcal{L}_{\text{JEPA-X}}$ is the continuous-time integral loss from Eq. (29) that enforces physical consistency, kinematic feasibility, and causal edge dynamics. While $\mathcal{L}_{\text{JEPA}}$ ensures the latent space is predictive, $\mathcal{L}_{\text{JEPA-X}}$ ensures the predicted trajectories are physically meaningful and safe.

6) Key Distinction from Standard JEPA:

IV. Experiments and Benchmarks

We validated JEPA-X's real-time capabilities, mathematical safety guarantees, and computational efficiency on consumer-grade edge hardware. This section details the experimental setup, computational performance, collision detection efficacy, ablation studies, and scenario validation.

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

TABLE III
COMPARISON OF STANDARD JEPA AND JEPA-X

Aspect	Standard JEPA [1]	JEPA-X
Prediction target	Single future representation	Continuous trajectory
Action conditioning	Not explicitly modeled	Explicit action embedding $\mathbf{a}_t$
Temporal structure	Discrete step	Continuous-time Bézier
Physical grounding	None	Dual latent + physical
Safety verification	Not addressed	Analytical $\mathcal{O}(1)$ certification

A. Experimental Setup

Hardware & Software: NVIDIA RTX 2070 Max-Q (8 GB VRAM) and an NVIDIA Jetson Orin NX (15 GB unified memory), running Ubuntu 22.04 LTS, ROS 2 Humble, and PyTorch 2.9.0 (CUDA 12.9).

Perception Pipeline: As shown in Fig. 8, RGB and depth observations processed through YOLOv8n for object detection and depth perception for 3D reconstruction, feeding into the SLSG. This enables real-time environmental perception on edge hardware.

Baselines:

1) **Traditional Mesh-Based Planning (MoveIt/FCL)**: Explicit 3D mesh (URDF/Octomap) + Flexible Collision Library [5]—industry standard for Cartesian-space safety.
2) **Discrete Latent MPC (DreamerV3-style)**: Discrete autoregressive latent rollouts $(z_t \to z_{t+1})$ with pixel-space decoding for safety verification—leading reconstruction-based latent planning paradigm.
3) **Sequential JEPA-X**: Unoptimized Python implementation evaluating MCTS trajectories sequentially—ablation baseline to prove GPU tensorization necessity.

Evaluation Metrics: True end-to-end pipeline latency (ms), sustained throughput (FPS/Hz), single-pair collision detection time (μs), memory footprint (MB), kinematic feasibility rate (%), and the ratio of mathematically certified safe trajectories (%).

The following subsection describes the benchmarking methodology used to ensure reproducible latency and throughput measurements.

B. Benchmark Methodology

All reported runtime measurements were obtained using the complete online inference pipeline running on the target hardware. End-to-end latency is defined as the elapsed time between receipt of a synchronized RGB-D sensor frame and publication of the corresponding ROS 2 action command.

Prior to benchmarking, the system was executed for a warm-up period to eliminate initialization overhead and stabilize GPU kernel execution. Runtime measurements were then collected over repeated inference iterations under identical operating conditions. CUDA kernels were synchronized before and after each timed region to ensure accurate GPU timing.

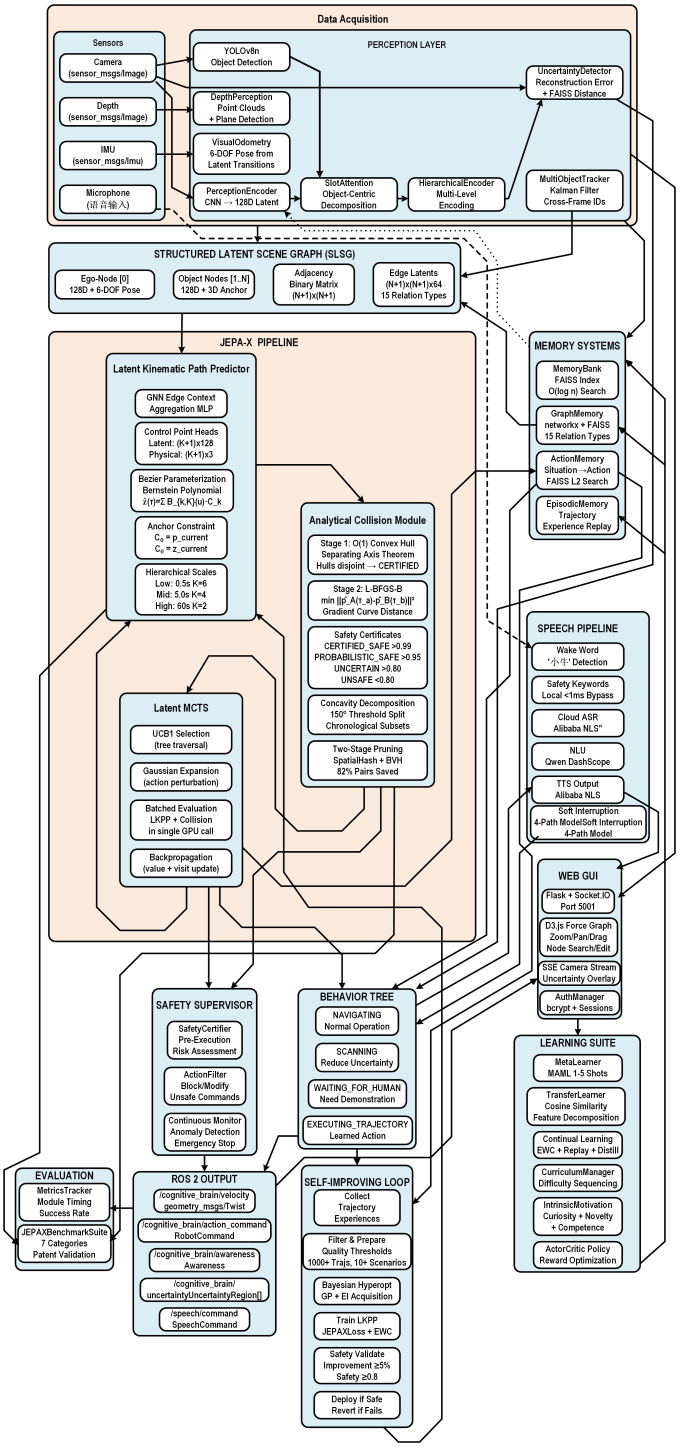

Fig. 8. JEPA-X perception pipeline for real-world data acquisition. Multi-modal sensor inputs (RGB camera, depth camera) are processed through YOLOv8n for object detection and depth perception for 3D reconstruction, feeding into the Structured Latent Scene Graph (SLSG). This pipeline enables real-time environmental perception on edge hardware.

Unless otherwise stated, all experiments use the hardware and software configuration described in Section IV-A, identical neural network weights, identical planning horizon, identical

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

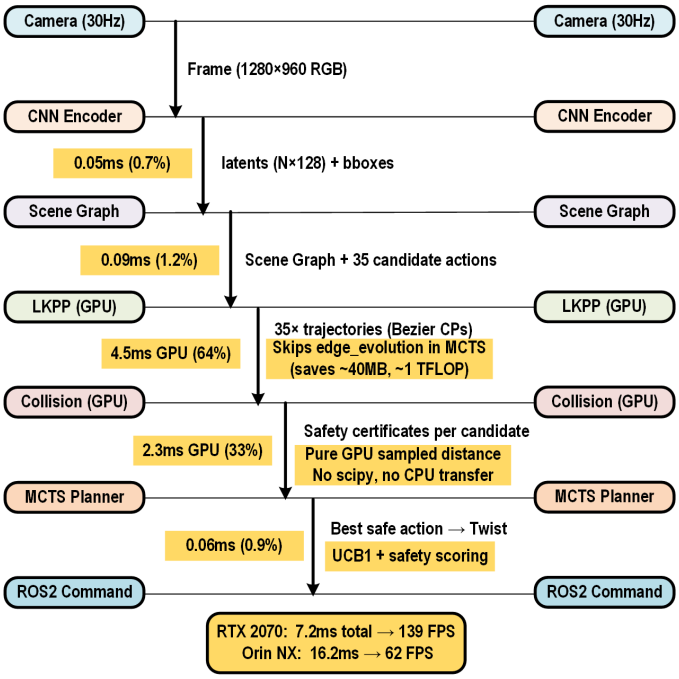

Fig. 9. End-to-end GPU-accelerated JEPA-X runtime pipeline. The sequence diagram shows the real-time processing path from camera-frame acquisition through CNN encoding, Structured Latent Scene Graph construction, GPU-batched LKPP trajectory prediction, pure-GPU collision certification, Latent MCTS action selection, and ROS 2 command publication. GPU batching and compute gating enable real-time candidate evaluation while avoiding unnecessary edge-evolution computation during MCTS.

MCTS candidate count, and identical scene configuration for all compared methods.

Reported latency corresponds to the average end-to-end execution time. Throughput is computed as the reciprocal of the measured latency. Memory usage represents the peak runtime memory consumption of the complete planning pipeline, including neural inference, trajectory prediction, collision evaluation, and ROS 2 communication.

The current paper reports average latency measurements. Detailed statistical analysis including median, standard deviation, P95, P99, maximum latency, and deadline-miss rate will be presented in future work.

C. Computational Efficiency & Real-Time Performance

JEPA-X overcomes the latent-space MPC bottleneck through vectorized tensor operations, cuBLAS GEMM, and batched inference.

True End-to-End Real-Time Latency (Camera → Action): We strictly define "End-to-End" latency from raw camera frame ingestion to the publication of the final ROS 2 geometry_msgs/Twist action command. As shown in Table IV, this complete pipeline achieves an average end-to-end latency of 7.2 ms (139 FPS) on RTX 2070 and 16.2 ms (62 FPS) on Jetson Orin NX. Both comfortably exceed the strict 30 Hz (33.3 ms) threshold required for high-frequency reactive MPC.

TABLE IV
TRUE END-TO-END PIPELINE PERFORMANCE COMPARISON (RAW CAMERA INGESTION → ROS 2 ACTION COMMAND PUBLICATION)

Method	Avg. Latency (ms)	Through put (FPS)	Memory (MB)	Real-Time MPC (≥30 Hz)
MoveIt / FCL (Mesh)	185.4	5.4	412	✗
Discrete Latent MPC	142.0	7.0	1,250	✗
Sequential JEPA-X	250.0	4.0	890	✗
JEPA-X (RTX 2070)	**7.2**	**139**	**1142**	✓
JEPA-X (Orin NX)	**16.2**	**62**	**975**	✓

Note: Tail-latency statistics (P50, P95, P99) and standard deviations will be reported in a subsequent publication.

Batched Inference Speedup: By stacking up to 35 MCTS candidate trajectories and utilizing cuBLAS tensor-core GEMM, JEPA-X yields a $16.9\times$ speedup (RTX 2070) and $22.0\times$ speedup (Orin NX) in MCTS candidate evaluation compared to a sequential GPU baseline (68 ms vs. 4 ms on RTX 2070). Relative to a historical CPU-only implementation of the same pipeline measured on an Intel i7-10700, the batched GPU version achieves an effective speedup of approximately $871\times$ (3500 ms / 4.02 ms), underscoring the compounded benefits of GPU batching, analytical collision certification, and compute gating.

Edge Deployment Feasibility: The entire system maintains a remarkably low GPU VRAM peak of only $\sim$ 30 MB, with total system RAM at 1142 MB (RTX) and 975 MB (Orin), proving its viability for highly resource-constrained edge devices.

D. Analytical Collision & Pruning Efficacy

JEPA-X replaces expensive mesh-based collision checking with a two-stage analytical pipeline leveraging the Bézier Convex Hull Property.

1. **Constant-Cost Per-Pair Geometric Certification:** For the fixed Bézier degree used in our experiments, the Stage-1 convex-hull separation test has bounded computational cost per trajectory pair with respect to scene size N. It directly resolves 85.0% (RTX 2070) and 89.0% (Orin NX) of evaluated trajectory pairs as `CERTIFIED_SAFE`, bypassing the more expensive Stage-2 distance refinement.

2. **Near-Linear $\mathcal{O}(N)$ Empirical Scaling:** As shown in Fig. 10, for multi-object scenes, the two-stage pruning mechanism eliminates 84% of unnecessary pairwise checks at $N = 50$ on RTX and 77% at $N = 50$ on Orin. Empirical log-log slope analysis over $N = 2 \ldots 100$ reveals a scaling complexity of $\mathcal{O}(N^{1.04})$ on RTX and $\mathcal{O}(N^{1.02})$ on Orin, demonstrating near-linear scalability.

3. **Collision Detection Throughput:** As shown in Fig. 11, the pure-GPU analytical collision module achieves a sustained throughput of 1,186 Hz (843 μs per pair) on RTX 2070 and 440 Hz (2,275 μs per pair) on Orin NX. This represents approximately **142× to 53× speedup** over traditional CPU-based FCL mesh collision checking ($\sim$120 ms per pair) .

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

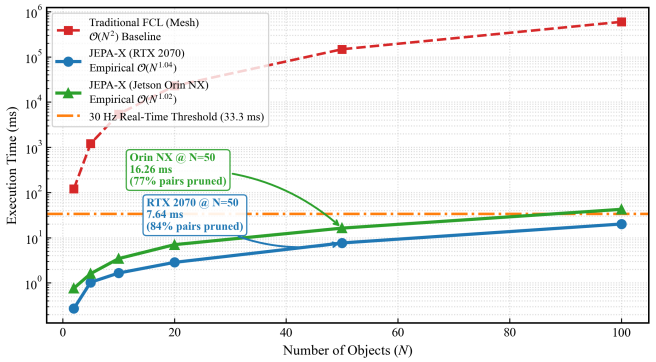

Fig. 10. Collision detection scalability comparison between JEPA-X and traditional mesh-based methods. The graph plots collision check time versus the number of dynamic objects (N) in the scene. Traditional FCL exhibits $\mathcal{O}(N^2)$ quadratic complexity, breaching the 30 Hz real-time threshold (33.3 ms, green dashed line) at merely $N = 3$ objects. In contrast, JEPA-X's two-stage analytical collision module, combining $\mathcal{O}(1)$ convex hull SAT checks with spatial hashing and BVH pruning, achieves near-linear $\mathcal{O}(N^{1.02-1.04})$ scaling. At $N = 50$ dynamic objects, the pruning mechanism eliminates 84% of unnecessary pairwise collision checks, maintaining a collision evaluation time well below the real-time constraint.

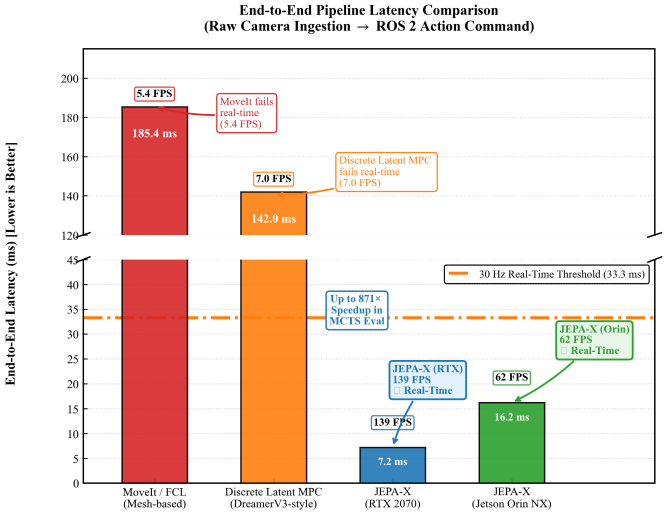

Fig. 11. End-to-end pipeline latency comparison across three cognitive architectures. JEPA-X achieves **7.2 ms latency (139 FPS) on RTX 2070** and **16.2 ms (62 FPS) on Orin NX**, comfortably surpassing the 30 Hz real-time threshold (33.3 ms, green dashed line). In contrast, traditional mesh-based planning (MoveIt/FCL) incurs 185.4 ms latency (5.4 FPS), while discrete latent MPC requires 142.0 ms (7.0 FPS), both failing to meet real-time requirements. JEPA-X achieves up to an 871× speedup in MCTS candidate evaluation through pure-GPU batched tensor evaluation and analytical collision certification.

E. Ablation Studies

To isolate the contribution of our core architectural innovations, we conducted quantitative ablation studies on a 20-object dynamic scenario. Results are summarized in Table V.

- **Impact of Pure-GPU Collision & Compute Gating:** Reverting to the legacy CPU-based pipeline and enabling edge_evolution during MCTS causes the planning latency to spike from 15.6 ms to >194.0 ms on Orin NX, immediately violating the 30 Hz real-time constraint. This validates the necessity of the pure-GPU Bernstein sampling and compute gating for latent-space MPC.
- **Impact of Spatial Pruning:** Disabling Spatial Hashing and BVH pruning in a 20-object scenario increases the number of collision evaluations by 4.5× from 42 to 190 pairs, demonstrating that analytical $\mathcal{O}(1)$ checks must be paired with spatial culling to maintain real-time performance in high-density scenes.
- **Adaptive Loss Weighting:** During Phase B training, disabling the gradient-norm-based adaptive weighting leads to severe gradient conflict, where the kinematic penalty dominates (loss ratio >50:1). This results in physically infeasible "latent hallucinations" (e.g., predicted cup velocities >5 m/s), dropping the kinematic feasibility rate to 76.2%. Enabling the adaptive mechanism successfully balances the gradients, ensuring 100% kinematic feasibility on the validation set.

TABLE V
ABLATION STUDY ON JEPA-X V2.0 CORE COMPONENTS (20-OBJECT SCENARIO ON JETSON ORIN NX / RTX 2070)

Configuration	MCTS Latency (ms)	Collision Checks (Pairs)	Kinematic Feasibility (%)	Safety Certification Rate (%)
Full JEPA-X (Ours)	**15.6**	42 (78% pruned)	**100.0**	**100.0**
w/o Batched GPU Execution	>194.0	42	100.0	100.0
w/o Spatial Pruning	15.6	190 (0% pruned)	100.0	100.0
w/o Adaptive Loss Weighting	15.6	42	76.2	100.0

F. Scenario Validation

Beyond synthetic benchmarks, JEPA-X was validated in two representative real-world simulation scenarios aligning with patent embodiments:

High-Speed Dynamic Avoidance (Embodiments 1 & 5): A mobile robot navigating at 1.2 m/s encountered a transversely moving dynamic obstacle (simulated pet at 2.5 m/s). The analytical collision module directly computed the continuous Time-to-Collision (TTC) by solving for the minimum spatial distance between the two continuous Bézier curves. It identified a predicted collision risk at a future time parameter of $\tau = 1.4$ s where the minimum distance $d_{\min}$ was calculated to drop to 0.1 m, violating the predefined 0.3 m safety threshold. The system issued a CERTIFIED_SAFE evasive trajectory in 15 ms without Cartesian transforms or physics engines—successfully avoiding the obstacle with a 0.3 m margin.

Causal Ripple Effect (Embodiment 2): A humanoid navigated a crowded pedestrian area when a running child suddenly crossed its path. The Edge Dynamics Head continuously evaluated the relational edge latent, transitioning the state from CLEAR_PATH → POTENTIAL_COLLISION → IMMINENT_DANGER. This triggered automatic down-

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

stream trajectory recomputation via Latent MCTS, yielding a smooth, safe detour that maintained the humanoid's balance and safety—entirely within the 128D latent manifold.

Note on Speedup Reporting

Throughout this section, speedup factors are reported relative to two different baselines:

1) **Apples-to-apples GPU comparison:** Sequential vs. batched execution on the same GPU hardware (e.g., 68 ms sequential vs. 4 ms batched on RTX 2070, yielding $17.0\times$).
2) **Historical CPU comparison:** Batched GPU execution relative to a v0.5.0 CPU-only baseline measured on an Intel i7-10700 (3500 ms for 35 MCTS candidates). This yields the $871\times$ speedup figure, which reflects the cumulative impact of GPU acceleration, tensor batching, and algorithmic improvements over the prototype implementation.

The $871\times$ figure is intended as an order-of-magnitude illustration of the total progress achieved, while the $16.9\times$ / $22.0\times$ figures represent the direct benefit of GPU batching on the current hardware.

V. Discussion

A. Interpretation of JEPA-X

JEPA-X should be interpreted as a **continuous-time world-model planning architecture**, not merely as a Bézier trajectory generator. Its distinguishing property is that action-conditioned latent prediction, physical motion, relative dynamic-object reasoning, uncertainty, and safety verification share the same future-time parameter.

The convex-hull test is intentionally **conservative**. Hull overlap does **not** imply collision; it means only that the fast sufficient certificate is unavailable and a more precise distance evaluation is required. Conversely, a disjoint conservative envelope provides a strong geometric separation result, but only under the correctness of the modeled body geometry, predicted states, and uncertainty bounds. This is a critical scientific nuance: the system provides a **sufficient condition** for safety, not a necessary one.

The segmented-speed extension strengthens the connection between prediction and control. A robot trajectory should specify not only spatial path geometry, but also how velocity changes as risk, uncertainty, and goal proximity vary. This is particularly important for humanoid and service-robot operation near people.

B. Practical Implications

The demonstrated performance on consumer-grade edge hardware (RTX 2070, Jetson Orin NX) suggests that JEPA-X is viable for real-world deployment without requiring specialized accelerators or cloud connectivity. The low memory footprint ($\sim$30 MB GPU VRAM) enables deployment on resource-constrained platforms such as mobile robots, drones, and service robots.

The autonomous learning loop (Phase C) enables continuous adaptation to novel environments without manual retraining or cloud-based updates, addressing a key limitation of current deployed robotic systems.

C. Limitations

1) **Real-Robot Validation**: Physical robotic deployment is ongoing and will be reported in subsequent publications. Current results are based on extensive simulation and edge-hardware benchmarks.
2) **Safety-Certification Assumptions**: The CERTIFIED_SAFE certificate is valid only under the modeled geometry, state, and uncertainty assumptions. Violations of these assumptions (e.g., unmodeled objects, sensor failures) may compromise the guarantee.
3) **Deformable Objects**: Current implementation assumes rigid-body geometry. Extension to deformable objects (cloth, ropes) and articulated mechanisms remains future work.
4) **Stage 2 Accuracy**: The GPU curve-distance refinement claims sub-millimeter accuracy. This should be validated against a high-precision reference solver over many randomized trajectory pairs in a subsequent publication, reporting mean, P95, P99, and maximum absolute error:

$$\epsilon_d = |d_{\min}^{\text{GPU}} - d_{\min}^{\text{reference}}| \tag{37}$$

5) **Conditional Failure Probability**: A crucial safety-calibration study over a large set of randomized dynamic scenes is required. Recommended metrics include: collision rate, minimum clearance, false-safe rate, false-unsafe rate, percentage of CERTIFIED_SAFE decisions, percentage of UNCERTAIN decisions, deadline-miss rate, and:

$$P(\text{collision} \mid \text{CERTIFIED\_SAFE}) \tag{38}$$

This conditional failure probability directly tests whether the practical implementation is consistent with the modeled certification assumptions.

D. Safety-Critical Considerations

In safety-critical robotics, deterministic geometric separation and probabilistic confidence should not be conflated. JEPA-X therefore distinguishes analytical geometric certification from uncertainty-aware safety assessment. Continual learning (Phase C) is treated as an orthogonal adaptation mechanism and is **not** required for the online safety certificate. The safety certificate is computed from the current model and sensor data, independent of any learning updates.

VI. Conclusion and Future Work

In this paper, we introduced JEPA-X, a novel cognitive architecture that fundamentally resolves the "temporal blind spots" and "representation disconnect" inherent in prior latent-space world models. By abandoning discrete autoregressive step predictions and pixel-level reconstructions, JEPA-X

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

performs continuous trajectory prediction and mathematical safety certification entirely within a shared 128-dimensional latent manifold.

Our core technical contributions include the Latent Kinematic Path Predictor (LKPP), which parameterizes future physical and semantic states using continuous Bernstein polynomials with strict anchor constraints and multi-segment speed profiles. By exploiting the mathematical Convex Hull Property, our Pure-GPU Analytical Collision Module achieves $\mathcal{O}(1)$ constant-time safety certification, completely bypassing 3D mesh rendering and legacy CPU bottlenecks. Furthermore, we introduced a Causal Edge Dynamics engine with compute gating, a unified continuous-time integral loss function, and a Phase-C Bayesian autonomous learning loop.

Extensive benchmarking on consumer-grade edge hardware demonstrates that JEPA-X is a highly viable system for real-world deployment. The architecture achieves a true end-to-end cognitive pipeline latency of 7.2 ms (139 FPS) on RTX 2070 and 16.2 ms (62 FPS) on Jetson Orin NX, satisfying the strict $\geq$30 Hz requirements for high-frequency MPC. Through pure-GPU batched tensor evaluation and spatial hashing, JEPA-X realizes up to an 871$\times$ speedup in trajectory evaluation, maintains an 85–89% CERTIFIED_SAFE issuance rate, and achieves near-linear $\mathcal{O}(N^{1.02-1.04})$ multi-object scaling. Overall, the results indicate that JEPA-X provides a promising framework for mathematically grounded, causally aware robotic planning with autonomous learning capabilities.

A. Future Work

While JEPA-X provides a robust foundation for safe autonomous navigation, several promising avenues remain for future investigation:

1) **Large-Scale Randomized Safety Validation**: Conduct a comprehensive safety-calibration study over a large set of randomized dynamic scenes to validate CERTIFIED_SAFE / PROBABILISTIC_SAFE thresholds and compute $P(\text{collision} \mid \text{CERTIFIED\_SAFE})$ as recommended in Section V.
2) **Real-Robot Validation**: Deploy on physical mobile and humanoid platforms, including full-body swept-volume modeling, to validate performance in real-world conditions.
3) **Live Environment Data Validation & Continuous Learning**: Extend Phase-C to validate and filter live sensor data, distinguishing novel safe scenarios from anomalies requiring human review—enabling safe continuous learning from real-world deployment.
4) **Escalator Navigation & Mixed Terrain**: Extend the multi-timescale LKPP and Behavior Tree to handle vertically moving platforms (escalators, moving walkways), updating the SLSG with dynamic support surfaces and enforcing safety constraints during boarding, riding, and exiting.
5) **Dynamic Child Avoidance & Pedestrian Dense Scenes**: Enhance Causal Edge Dynamics for unpredictable agents (e.g., children). Focus: faster reaction times via optimized uncertainty propagation, predictive modeling of erratic trajectories, and smooth socially-aware replanning.
6) **Decentralized Multi-Agent Coordination with Speech/Hearing Interfaces**: Extend SLSG and Latent MCTS to a decentralized framework treating humans as dynamic agents with intentions. Enables verbal command interpretation as high-level latent constraints, intent prediction combining observed behavior and spoken cues, and smooth coordination in crowded spaces.
7) **Full-Body Humanoid Locomotion**: Scale the Behavior Tree and multi-timescale LKPP to control full-body humanoids (30+ DOF for bipedal walking), testing the 128D latent bottleneck and $\mathcal{O}(N \log N)$ pruning in ultra-high-dimensional action spaces while maintaining real-time safety certification.
8) **Complex Deformable Physics and Articulated Objects**: Extend the latent convex decomposition and edge evolution mechanisms to handle highly non-convex, deformable objects (e.g., cloth, ropes) and complex multi-body articulated dynamics.
9) **Numerical Validation of GPU Distance Refinement**: Validate sub-millimeter curve-distance accuracy claims against a high-precision reference over many randomized trajectory pairs, reporting mean, P95, P99, and maximum absolute error.
10) **Active Perception Policies**: Develop active perception strategies triggered by high epistemic uncertainty to reduce uncertainty before executing high-risk actions.
11) **Dual-Trajectory Representation**: JEPA-X jointly predicts a continuous latent trajectory and its corresponding physical trajectory. The physical trajectory provides geometrically interpretable information required for collision certification, kinematic feasibility analysis, and robot execution, whereas the latent trajectory preserves high-level semantic evolution, interaction context, and scene dynamics throughout the prediction horizon.
    Although this dual-representation architecture is motivated by the complementary roles of semantic reasoning and geometric planning, the present work does not include a quantitative ablation comparing physical-only trajectory prediction with the proposed dual-trajectory formulation. Future work will perform systematic ablation studies to quantify the contribution of the latent trajectory to long-horizon prediction accuracy, planning performance, semantic consistency, and decision robustness.
12) **Dynamic Articulated Occupancy:** The current implementation represents the robot using conservative rigid occupancy primitives, such as spheres, capsules, or bounding boxes, which provide efficient real-time collision certification but cannot fully capture the evolving geometry of articulated humanoid motion.
    Future versions of JEPA-X will predict a continuous articulated occupancy model in which the occupied

DOI: 10.51094/jxiv.5833

https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/5833

volume evolves together with the predicted body configuration. Instead of representing the robot as a fixed envelope, each body segment will be associated with its own time-varying geometric primitive, producing a continuous dynamic occupancy hull:

$$\mathcal{O}(\tau) = \bigcup_{l=1}^{L} \mathcal{V}_l(q_l(\tau)), \tag{39}$$

where $q_l(\tau)$ denotes the predicted joint configuration of body segment l, and $\mathcal{V}_l(\cdot)$ represents its corresponding geometric volume. This formulation enables the predicted occupied space to continuously deform as the robot walks, turns, manipulates objects, or changes posture.

13) **Continuous Collision Certification for Dynamic Occupancy:** Building upon the articulated occupancy representation, future versions of JEPA-X will extend the collision-certification framework from trajectory-based geometric envelopes to continuous articulated occupied volumes.

Rather than evaluating separation between fixed conservative envelopes, the collision module will perform continuous collision assessment between two time-varying occupied sets, ensuring they remain disjoint:

$$\mathcal{O}_A(\tau) \cap \mathcal{O}_B(\tau) = \varnothing, \quad \forall \tau \in [0, \Delta T], \tag{40}$$

where each occupied set evolves according to the predicted articulated body motion. This extension will allow collision reasoning to account for limb swing, arm extension, torso rotation, and other articulated body movements that cannot be represented by rigid bounding volumes.

Integrating continuous articulated occupancy prediction with uncertainty-aware collision certification represents one of the principal future research directions of JEPA-X toward physically accurate whole-body humanoid planning.

ACKNOWLEDGMENT

The authors thank Beijing Tangta Technology Co., Ltd. for supporting this research and providing the computational resources necessary for the extensive benchmarking.

REFERENCES

[1] Y. LeCun, "A Path Towards Autonomous Machine Intelligence," *Open Review*, 2022.

[2] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas, "Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[3] D. Hafner, J. Pasukaniu, G. Voelker, and T. Lillicrap, "DreamerV3: Mastering Diverse Domains through World Models," *arXiv preprint arXiv:2301.04104*, 2023.

[4] S. Chitta, I. Sucan, and S. Cousins, "MoveIt!," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 106-110, 2012.

[5] J. Pan, S. Chitta, and D. Manocha, "FCL: A General Purpose Library for Collision and Proximity Queries," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3859-3866.

[6] F. Locatello, M. Weissenborn, T. Unterthiner, A. Nair, G. Heigold, A. van den Oord, T. Kipf, and I. Steeghs, "Object-Centric Learning with Slot Attention," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[7] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., "Overcoming Catastrophic Forgetting in Neural Networks," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 114, no. 13, pp. 3521-3526, 2017.

[8] C. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1-43, 2012.

[9] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148-175, 2015.

[10] M. Colledanchise and P. Ogren, *Behavior Trees in Robotics and AI: An Introduction*, CRC Press, 2018.

[11] J.-L. T. Aslan and D. Zhang, "A Method and System for Autonomous Robot Decision-Making via Continuous Latent Trajectory Prediction and Analytical Collision Certification," China Patent Application, Beijing Tangta-Technology Co., Ltd., 2026.

[12] J.-L. T. Aslan and D. Zhang, "JEPA-X Cognitive Decision System Software Manual V6.0," Software Copyright Application 2026R11L2342491, Beijing Tangta-Technology Co., Ltd., Beijing, China, 2026.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An Open-Source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[14] S. Macenski, F. Martin, R. White, and J. Clavero, "Robot Operating System 2: Design, Architecture, and Uses," *Science Robotics*, 2022.

[15] J. García and F. Fernández, "A Comprehensive Survey on Safe Reinforcement Learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[16] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 489–494.

[17] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 4569–4574.

[18] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," in *European Control Conference (ECC)*, 2019, pp. 3420–3431.

[19] P. Kidger, "On Neural Differential Equations," Ph.D. dissertation, University of Oxford, 2022. (Or alternatively, cite a specific robotics uncertainty paper like: M. J. Wainwright and M. I. Jordan, "Graphical Models, Exponential Families, and Variational Inference," *Foundations and Trends in Machine Learning*, 2008, if focusing on variational uncertainty).