

# World Models for GUI Agents

## Insights from Autonomous Driving

Tong Zhang

June 12, 2025

## What are World Models?

- A world model learns a representation of its environment and how it changes over time.
- Key functions relevant to our problem:
  - Constructing internal (often compressed) representations of the world.
  - Understanding the current state (perception, scene interpretation).
  - Predicting future scenarios or states from these representations.
  - Generating realistic and controllable simulations of the environment.

Autonomous driving research offers rich techniques for these tasks.

# Why World Models (WM)?

**Efficiency:** increase sampling efficiency (for embodied tasks such as driving)

**Time Manipulation:** “counterfactual” simulations (reset to exact states)

**Safety:** a risk-free environment to test dangerous edge cases and failures

**Environmental Mocking:** mock or emulate (hardware or external services)

## Categories of WMs for Autonomous Driving

- 1 Next Frame/Sensor Data Prediction
- 2 Implicit Scene Representation & Understanding (VAE)
- 3 Trajectory and Occupancy Prediction
- 4 Closed-loop Simulation & End-to-End World Simulation

# The Challenges in Complex Apps

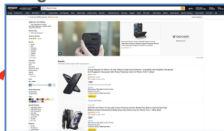
- **A core problem: an extremely large number of possible user interface (UI) scenarios or "scenes".**
- **Status in APP is much more complicated than in the physical real world.**
  - **Real-world environments can be described with a set of physical states.**

No need for combinatorial search: the state space is continuous and can be parameterized with a few variables.
- **How to fully utilize privileged data and normal data for efficient modeling and reasoning.**

# The Core Idea: Compact Representation & Reconstruction

- **Problem Statement:** How can we model a vast number of app scenes without explicitly learning each one?
- **Proposed Solution:**
  - Use a few "status" elements or a concise "description" (e.g., key UI elements, layout, user intent tokens, language descriptions) to represent the current app scene.
  - Rebuild full scene understanding, predict future scenes/states, or generate interaction sequences through:
    - **Search:** Retrieve similar, fully-realized scenes/workflows from a database based on the compact description.
    - **Decoder/Generative Model:** Use a model to reconstruct scene details or predict future states from its compact representation.
- This approach aims to reduce complexity and enhance scalability.

## Step 1: Web Random Walking



## Step 2: Synthesis



## Qwen2-VL Response

When you click on the search bar after typing "iPhone 16 case," the Amazon webpage will transition from a general homepage to a product search results page. Here's what will happen:

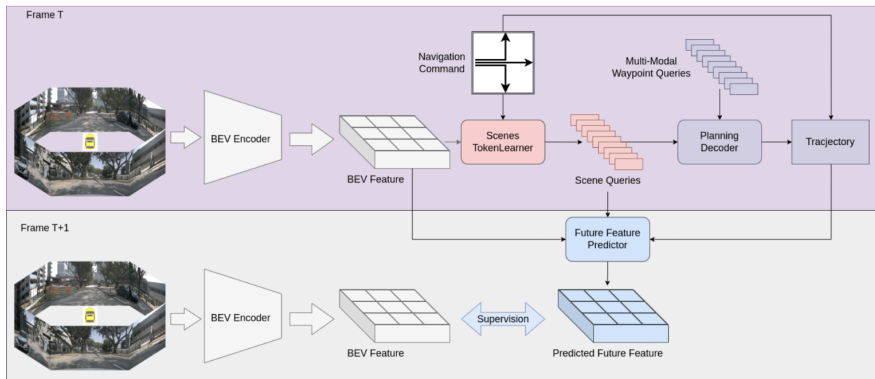
1. Search Results Display: The main content area will be populated with a list of iPhone 16 cases available for purchase. Each listing will include an image of the product, its name, price, customer ratings, and options like "Add to Cart."
2. Navigation Filters: On the left side of the page, you'll see various filters that allow you to narrow....

Action Proposal → Description Prediction → Goal Evaluation → Execute

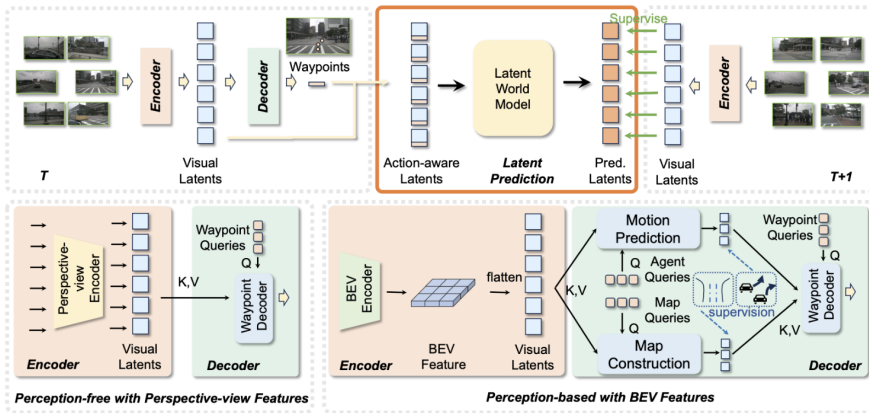
# Latent State Representations

- World models often learn a **latent (compressed) representation** of the environment.
- This captures essential information for understanding, prediction, and generation.
- **Examples from Driving:**
  - **LAW,SSR:** Learn latent dynamics models.
  - **PIWM (Dream to Drive):** Learns "intention-aware latent states."
  - **GAIA-2:** Uses a latent diffusion world model.
  - **UniFuture:** Employs Dual-Latent Sharing for joint image-depth modeling.

# Navigation-Guided Sparse Scene Representation



# Enhancing End-to-End Driving with Latent World Model



## SSR

- 1 Selects the 16 most informative BEV slots by back-propagating planning gradients.
- 2 Uses a temporal RSSM to predict future token tables up to 3 seconds ahead, enabling imagination solely in latent space.
- 3 Eliminates heavy 3D detection heads—planners directly consume sparse tokens, achieving a 10× runtime speed-up.

## Law

- 1 Dynamic predicts future ViT latents without pixel-level reconstruction.

**Task Selection is very important!**

# Decomposing Privileged and Normal Data in App World Models

## ● **Privileged Data:**

- Developer or internal logs, ground-truth UI structure, labeled widget states, backend traces, full interaction history, detailed user intent, or hidden variables available during training.
- Provides rich supervision for building compact scene/state representations and simulating rare or difficult scenarios.

## ● **Normal Data:**

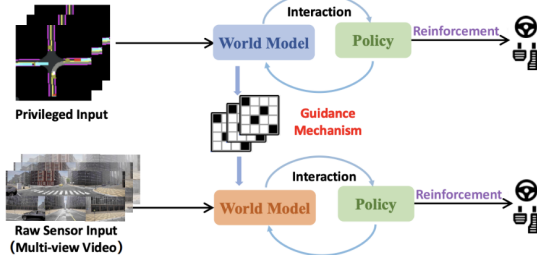
- Real user interactions, screenshots, partial logs, and sensor data available at test/inference time.
- Reflects the “observable” state, with limited or noisy information, closer to what models encounter in deployment.

## ● **Decomposition Strategy:**

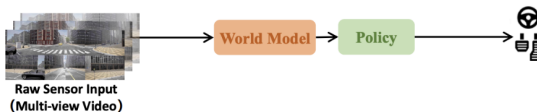
- Use privileged data to train or guide representation learning .
- Align or distill knowledge into models operating on normal data only, via feature distillation or rollout guidance.
- During inference, rely solely on normal data, but benefit from representations/models enhanced by privileged guidance.

# Raw2Drive: Dual-Stream Model-based RL

## a) Train Phase

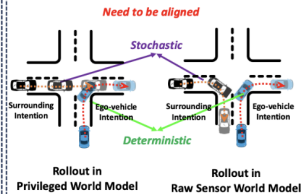


## b) Inference Phase

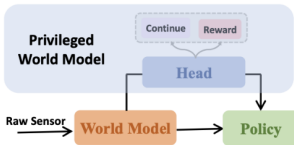


## c) Guidance Mechanism

### > Rollout Guidance



### > Head Guidance



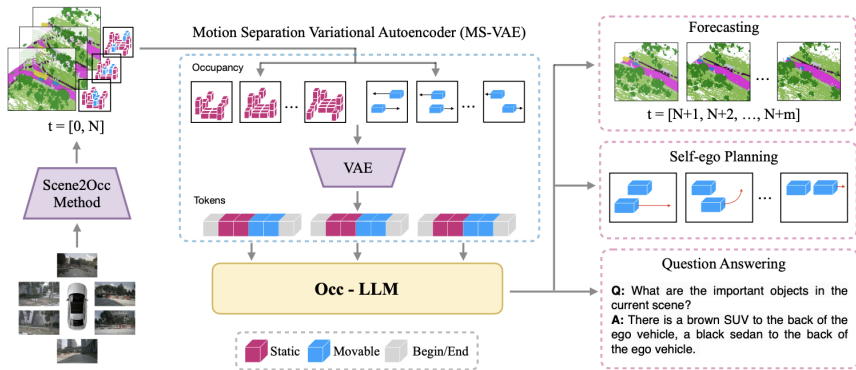
- 1 **Rollout Guidance:** KL divergence and feature distillation are used to align the latent states of the raw and privileged world models at each rollout step, ensuring that the state distribution of the raw model does not drift over time.
- 2 **Head Guidance:** All training rewards and supervision signals for the policy are derived from the privileged world model, preventing raw sensor noise from degrading learning and transferring high-quality planning competence.
- 3 Demonstrates that dual-stream RL outperforms both pixel-only and privileged-only baselines.

## Backend Logic Prediction + UI Prediction

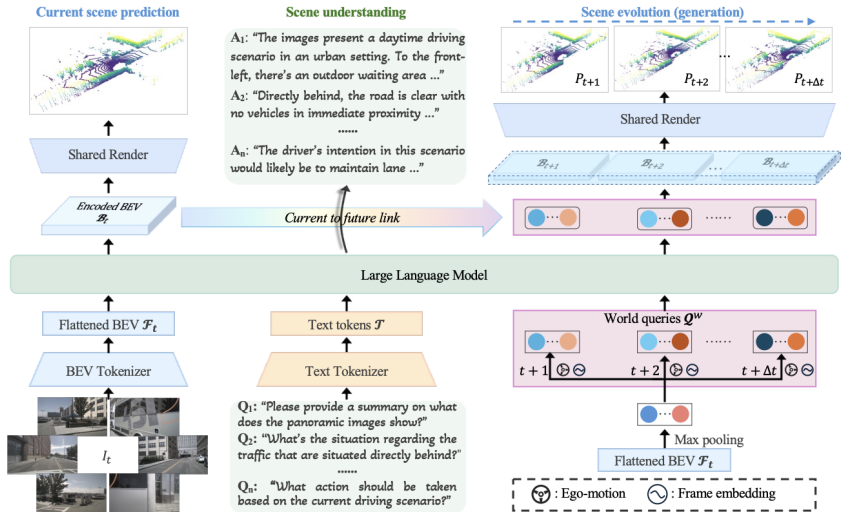
# LLM driven Autonomous Driving Agents

- Recent advances in large language models (LLMs) and multimodal architectures have enabled unified modeling of driving perception, reasoning, prediction, and planning.
- A new paradigm: **representing the environment as compact, structured tokens**—either from occupancy grids or BEV features—that can be processed by LLMs for a wide range of tasks.
- **Key Questions:**
  - How to encode rich spatial-temporal information into a sequence of tokens for LLMs?
  - How to support both understanding (reasoning, QA, description) and generation (forecasting, planning) in a single model?
- Recent works explore **occupancy-based** (Occ-LLM) and **BEV/vision-based** (HERMES) unified world models, bridging perception and cognition in autonomous driving.

# Occ-LLM: Occupancy-Based Large Language Model



# HERMES: A Unified Self-Driving World Model



## Occ-LLM

- 1 **Motion Separation VQ-VAE (MS-VAE):** Separates dynamic and static occupancy elements using two distinct VQ codebooks.
- 2 **Patchified Latent Representation:** Converts occupancy grids into patch-based latents as input tokens for the LLM.
- 3 **Occupancy-to-LLM Integration:** Directly feeds occupancy latents and action tokens into an LLM (Llama-2 7B), enabling unified forecasting, planning, and QA tasks.

## HERMES

- 1 **BEV Tokenizer:** Compresses multi-view camera features into compact BEV tokens and inserts BEV-pooled "world queries" into the LLM sequence.

# Conclusion

- Compact latent representations and unified frameworks.
- Decomposing privileged and normal data
- Leverage prior of large language models (LLMs).

## High-Fidelity Image Editing (vivo Work)

- The core is a well-defined **energy minimization framework** that balances three objectives:

### Energy Function:

$$E(\mathbf{H}) = \underbrace{\|\mathbf{H} - \mathbf{H}_0\|^2}_{\text{Fidelity}} + \alpha \underbrace{\sum_k \sum_{i \in \mathcal{C}_k} (H_i - \mu_k)^2}_{\text{Intra-Component}} + \beta \underbrace{\sum_{(i,j) \in \mathcal{B}} (H_i - H_j)^2}_{\text{Boundary}} \quad (1)$$

Optimized via Jacobi iteration with rapid convergence

# Next Step: VLM with Image-driven Self-Reflection

**Core Idea:** Model a self-reflective reasoning loop inspired by human Aha moments ( DeepSeek-R1).

## Detailed Workflow:

- 1 **Thinking:** The Vision-Language Model (VLM) does panning and generate a candidate editing prompt for testing.
- 2 **Thinking with feedback from Flux:** The Flux model generates an edited image based on the candidate prompt and append to the thinking process.
- 3 **Iterative Reasoning (Aha-moment):** The VLM, now visually informed, updates or refines the next reasoning step, achieving insights triggered by visual feedback.

**Research Hypothesis:** Explicit integration of generated visual feedback into VLM reasoning loops will significantly enhance multi-round image editing quality and semantic precision.

- **Multi-Agent CoT:** Agents co-train and share intermediate thoughts; CoT reasoning emerges via modular weight updates.
- **Natural Language Feedback in RL:** Intrinsic reward from language feedback + semantic multi-goal grounding + active self-query.
- **Latent-Space Image Reasoning:** Joint training of planner and world model in image embedding space; simulate edit–reflect–replan loop without decoding.

# Core Principle: Imagine First, Act Later

- **World Model (WM)** = a learned simulator of environment dynamics.
- **Planning loop:** *simulate* → *evaluate* → *choose* → *execute*.
- **Benefits**
  - 1 **Sample-efficiency** — one real step can fund thousands of model roll-outs.
  - 2 **Safety** — probe risky behaviours in silico before field deployment.
  - 3 **Long-horizon reasoning** — bridge sparse or delayed rewards.
  - 4 **Interpretability** — imagined trajectories explain decisions.

# Unified Six-Step Workflow



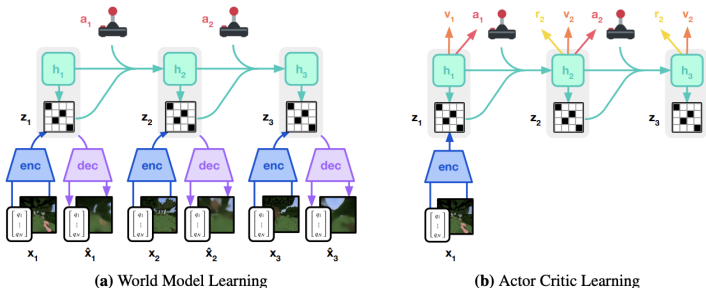
## A. WM Architectures

- **Latent Dynamics** — DreamerV3, TD-MPC 2
- **Pixel / Video Generation** — Vista, Drive-WM
- **Diffusion-based** — DIAMOND, DrivingDiffusion
- **Transformer / LLM** — UniZero, 3D-VLA
- **Hybrid (Symbolic + Neural)** — WorldCoder

## B. Planning Methods

- **Policy Optimization in Latent Space** — Dreamer, HarmonyDream
- **Model-Predictive Control (MPC)** — TD-MPC 2, DINO-WM
- **Trajectory Sampling & Ranking** — Sparse Imagination, NWM
- **Visual Reward Evaluation** — Drive-WM, Vista
- **Logical Search over Code** — WorldCoder

# Paradigm 1 Latent Policy Learning (Dreamer Family)



## Latent Imagination-Based Policy Learning

- 1 **Roll-out:** RSSM generates latent trajectory  $\langle z_{t:t+H}, \hat{r}_{t:t+H} \rangle$ .
- 2 **Evaluate:** Critic estimates return  $\hat{V}(z_t)$ .
- 3 **Optimize:** Actor receives gradients from imagined returns.

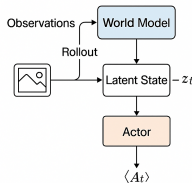
*Pros:* high throughput, offline friendly.

*Cons:* slower adaptation at test time; compounding model error.

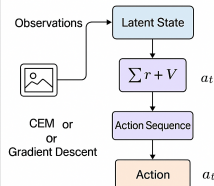
# Paradigm 2: Latent MPC (TD-MPC 2)

- 1 Encode observation  $\rightarrow z_t$ .
- 2 Initialise action sequence  $\mathbf{a}_{t:t+H-1}$ .
- 3 Roll-out in WM; objective: maximize 
$$\sum_{h=0}^{H-1} \hat{r}_{t+h} + V(z_{t+H}).$$
- 4 CEM or gradient descent refines  $\mathbf{a}_{t:t+H-1}$ .
- 5 Execute first action, re-plan every step.

Policy Optimization in Latent Space



Model-Predictive Control (MPC)



*Dreamer vs MPC*

*Pros:* responsive, fault-tolerant, no explicit actor needed.

*Cons:* heavier compute; tight latency budgets.

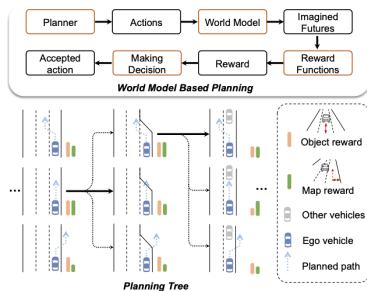
# Paradigm 3: Diffusion Simulation + Reward Ranking

- Candidate trajectories  $\rightarrow$  diffusion world model imagines multi-camera futures.
- Vision-based reward model scores each imagined video:
  - 3D object detection
  - HD map prediction
  - Optionally, GPT-4V evaluation
- Choose action with highest predicted reward:

$$a^* = \arg \max_{a_i} \text{Reward}(a_i)$$

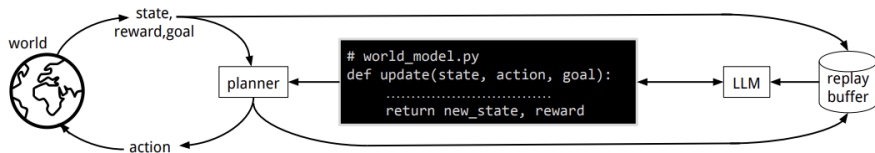
*Pros:* human-interpretable futures; flexible reward design.

*Cons:* diffusion is compute-intensive  $\rightarrow$  think in the early stage?.



*Drive-WM: Planning via diffusion-predicted futures and image-based rewards.*

# Paradigm 4 — Code-World Models (WorldCoder)



**Premise**: the world model *is* a Python script; an LLM rewrites `step(state, action)` whenever observations disagree.

**Loop**: *Synthesize*  $\rightarrow$  *Plan*  $\rightarrow$  *Repair*. Patch code with  $(s, a, s')$ ; run it to search an optimistic action sequence; execute the first unfalsified action, refine if wrong.

**Pros**: interpretable rules, symbolic roll-outs are fast, easy task transfer by editing code.

**Cons**: poorly scales to high-dimensional or stochastic worlds; logical search may explode combinatorially.