Multimodal World Models and Language-Based Planners for Reflective Reasoning

Tong Zhang

Introduction

My research will focus on the interplay between a **world model** and a **language-based planner** as the foundation of an intelligent agent. The central goal is to develop an architecture where a planning module, powered by natural language reasoning, operates in tandem with a learned world model that simulates the environment. This planner–world model loop forms a *reflective reasoning cycle*: the planner proposes actions or strategies in language, the world model predicts the consequences in a grounded representation of the world, and the planner then reflects on these outcomes to refine its next steps. By combining the abstract reasoning capability of language models with the predictive power of world models, the agent can "think before it acts"—iteratively imagining and evaluating its options within a safe internal sandbox.

This vision is motivated by the need for AI systems that can tackle complex, multi-step tasks in dynamic environments while maintaining interpretability and sample efficiency. Pure end-to-end approaches often struggle with long-horizon reasoning or require enormous data to learn implicit models of the world. By contrast, an explicit world model provides a compressed but faithful simulation of environmental dynamics, and a language-based planner offers flexibility in reasoning using high-level knowledge and context. The synergy between them enables **reflective reasoning**: the agent can internally generate hypothetical scenarios (via the world model) and reason about them (via the planner) before committing to real actions. This yields several benefits: the ability to handle long-horizon planning through lookahead, improved safety and goal alignment by avoiding clearly bad trajectories, and greater transparency of the decision process through intermediate reasoning steps.

In the following, I formalize the proposed architecture and describe two key application scenarios that will serve as testbench for this research. The first is an interactive image editing framework where a vision-language model acts as a high-level controller or scheduler, guiding a diffusion-based image generator in its latent space through iterative refinement steps. The second is a browser-based WebAgent environment where the agent interacts with web pages through DOM elements; here the world model simulates forward transitions of the web interface in response to actions, and the planner (a language model agent) reasons over the web page content and interaction history to decide the next action. These scenarios exemplify the broader theme of *planning–simulation loops*: in both cases, a planner and a world model engage in a cycle of proposing and evaluating actions, embodying the principle of reflective reasoning. Furthermore, I will outline how **latent visual chain-of-thought** representations, **multi-agent self-evolution** , and **natural language feedback** as a reward signal fit into this vision to enhance the agent's learning and performance.

1 Reflective Reasoning Architecture

At the core of this research vision is a unified architecture consisting of a language-based planner Π and a learned world model M that operate in a closed-loop. We can formalize the interaction as follows. The agent maintains an estimate of the environment *state* s_t at time t (which could be a real observed state or an internally simulated state), as well as a history or memory h_t summarizing past events (e.g. past actions, observations, and any internal chain-of-thought). At each step, the planner receives (s_t, h_t) and outputs an *action* $a_t = \Pi(s_t, h_t)$, expressed in a suitable language or symbolic format. The world model in turn produces a predicted next state $\hat{s}_{t+1} = M(s_t, a_t)$, simulating the effect of taking action a_t in state s_t. This prediction is fed back to the planner: the planner updates its internal history $h_{t+1} = f(h_t, a_t, \hat{s}_{t+1})$ (for some update function f that appends the action taken and the outcome) and can now analyze the imagined outcome $\hat{s}_{t+1}$. In the reflective reasoning loop, the planner can iteratively sample and score multiple action trajectories efficiently and choose the most appropriate action based on the reward.

Take a Monte-Carlo "imagination engine" as an example: from the current belief state s_t the planner samples a small set of candidate actions $\{a_t^{(i)}\}_{i=1}^{N}$, propagates each through the world model to obtain hypothetical successors $\hat{s}_{t+1}^{(i)} = M(s_t, a_t^{(i)})$, and scores the resulting futures with a task utility $U(\hat{s}_{t+1}^{(i)})$. Any branch whose predicted reward is sub-threshold or violates safety constraints is rejected; if all branches fail, the planner resamples or lengthens the look-ahead horizon until an acceptable trajectory emerges. The real action issued to the environment is

$$a_t^{\star} = \arg \max_{i \in \{1,\dots,N\}} U\big(\hat{s}_{t+1}^{(i)}\big),$$

making the choice explicitly traceable to a ranked set of imagined outcomes. After execution the agent observes the true next state s_{t+1}, logs $(\hat{s}_{t+1}^{\star}, s_{t+1})$ to refine M via the prediction error $\|s_{t+1} - \hat{s}_{t+1}^{\star}\|$, and appends $(a_t^{\star}, s_{t+1})$ to the history h_{t+1}.

Monte Carlo rollouts are only one instantiation of this reflective mechanism; beam search, gradient-guided optimization, or tree-expansion RL can be substituted without altering the loop structure. Whichever generator is used, the agent benefits from internal counterfactual reasoning—evaluating futures before acting—which yields high sample efficiency, principled constraint handling, and interpretable decisions, since every action stems from an explicit comparison among simulated trajectories. The same architecture therefore scales naturally from continuous vision tasks to discrete web interaction, and will serve as the foundation for a robust, general-purpose reasoning system in the coming stages of this project.

An important aspect of the architecture is the representation of states and the chain-of-thought. The world model typically operates over *latent representations* z_t of the state (such as the hidden vector encoding an image or a compact description of a web page) rather than directly on raw observations. These latent states serve as a compact "imagination space" for M. Likewise, the planner's chain-of-thought can be represented either explicitly in language/vision or implicitly in latent form. Explicit chain-of-thought (where the planner articulates each reasoning step in words) improves transparency, but it can be slow and sometimes limiting. In contrast, a **latent chain-of-thought** allows the planner to reason using internal neural representations that do not have to be verbalized, which can be more efficient and closer to how humans might mentally simulate outcomes without describing every detail in words. In this research, I will explore both modes: using language when interpretability is prior, and latent internal reasoning when speed or complexity demands it. The integration of these will be key to building an agent that is both understandable and effective.

2 Application Scenario 1: Vision-Language Guided Image Editing

The first application scenario is an interactive image editing framework in which a vision-language model and a generative diffusion model work in concert as planner and world model, respectively. The task here is: given a source image and a high-level edit specification (for example, "remove the tree from the backyard and make the sky sunset orange"), the agent should produce an edited image fulfilling the request. Achieving this through a single pass of a diffusion model is difficult when the edit involves multiple sub-tasks or precise changes. Instead, the agent will perform a sequence of editing steps, each one bringing the image closer to the desired outcome. This sequential approach is orchestrated by a **vision-language planner** (the scheduler), which uses iterative reasoning to decide on each editing action, while the **world model** is embodied by the diffusion model operating in its latent image space.

Concretely, let z_t denote the latent representation of the image at step t (with z_0 being the latent of the original input image, or an initial noise if generating from scratch). The diffusion model provides a function for applying a controlled edit to the latent: we can denote by $z_{t+1} = D(z_t, a_t)$ the result of diffusing the image z_t for a certain number of steps or in a certain manner guided by an edit command a_t. Here a_t is a high-level action or instruction, which could be represented in language (e.g. "paint the sky orange") or as some structured editing directive. The role of the planner Π_{VL} is to generate these actions a_t based on the current image and the overall goal. Because Π_{VL} is a vision-language model, it can understand both visual input and text: it encodes the current latent z_t (or a rendering of it) to assess the image's content, and it encodes the textual goal description (the requested edit). From these, it reasons about what modification is needed next.

For instance, early in the sequence it might output a_1: "remove the tree in the backyard," focusing on the first part of the task. After the diffusion model executes this and produces a new image z_2 with the tree gone, the planner observes z_2 and might issue a_2: "adjust the sky color to a warm sunset orange." Through this dialogue, complex edits are broken down into a series of simpler steps, each addressed in turn.

A key characteristic of this framework is that the reasoning happens over *latent representations, not raw pixels*. The diffusion model's latent space is a compressed, learned representation where semantic changes (like removing an object or changing color tone) can be effected more directly and smoothly than in pixel space. Working in the latent space allows the agent to maintain a **latent visual chain-of-thought**: the intermediate latents $z_1, z_2, \ldots, z_n$ constitute a sequence of imagined images that progressively incorporate the intended changes. This is analogous to a chain-of-thought in language, except that here the "thoughts" are implicit in visual features. For example, z_1 represents the intermediate concept of "image without the tree," and z_2 represents "image without the tree and with an orange sky." The planner might not explicitly narrate these concepts in words, but it is reflected in the latent trajectory. By not requiring explicit verbalization at each step, the planner can leverage powerful latent representations to reason about visual content directly, avoiding the inefficiency of describing every detail in text.

Throughout the iterative editing process, the agent relies on **visual feedback** to improve the outcome progressively. After each action, the planner evaluates the new image (via its vision encoder) to see how the visual scene has changed relative to the goal. This feedback loop allows the planner to correct course if needed. For instance, if after applying an edit the sky's color is still not vibrant enough, the planner can issue another action to further increase the saturation. Conversely, if an edit overshoots (imagine the sky turned too red), the planner can dial it back in a subsequent step. The loop continues until the planner determines the image matches the desired specification or a maximum number of steps is reached. At that point, the final image is decoded

from the latent and presented as the output.

This scenario demonstrates the power of the planner–world model architecture in a creative domain. The diffusion model D serves as a learned world model for images: it encapsulates knowledge of how visual content can be altered realistically (for example, how removing a tree would affect the surrounding pixels, or how the lighting should adjust when the sky color changes). The planner Π_{VL} brings high-level reasoning, decomposition of tasks, and understanding of the user's request. Importantly, because the planner is language-based, it can potentially incorporate textual guidance or constraints as well. Imagine a user additionally saying, "don't affect the house in the foreground"; the planner can integrate that instruction into its chain-of-thought, perhaps by focusing edits on certain regions or masking out the house. By working in tandem, the two components achieve something neither could do alone: the planner ensures the right edits are done in the right order, and the world model ensures each edit is executed in a photorealistic and coherent way.

3 Application Scenario 2: WebAgent with Multimodal World Model

The second application scenario involves an autonomous agent that interacts with websites through a browser interface, aiming to perform user-specified tasks (such as booking a ticket, finding information, or automating form submissions). In this context, the environment is the web page (consisting of a Document Object Model, or DOM, along with the visual rendering of that DOM), and actions are things like clicking buttons, typing into fields, scrolling, or navigating links. Building a competent WebAgent is challenging due to the enormous diversity of websites and the need for combining language understanding (to read and interpret page content) with decision making (to choose correct actions among many) and long-horizon planning (to complete multi-step procedures). Here, the proposed planner–world model architecture provides a clear pathway to a solution: the language-based planner interprets and reasons about the content and the task, while a world model of the web environment simulates the outcomes of possible user interface actions.

Formally, we can describe the state s_t at time t as a combination of current DOM data and any relevant visual or textual information extracted from the page, along with the agent's internal context (e.g., what subtasks have been completed so far). A simple representation of s_t could be a textual description of the page (including visible text, the hierarchical structure of elements and their properties such as 'button' or 'textbox'), augmented by a record of past actions and observations of the agent (this forms the history h_t). The planner in this scenario, Π_{web}, would likely be built on a large language model that has been adapted to the web interaction domain. Given (s_t, h_t), the planner could produce an output like: *Thought:* 'I need to log in first to proceed. There is a 'login' button on the top right.' *Action:* `click(#login-button)`. This format illustrates that the planner can have an internal reasoning (the Thought) leading to an actual executable action (the Action). While the thought is not strictly necessary to execute the action, exposing it in language helps with transparency and debugging; the agent is essentially explaining to itself what it's doing and why.

Now enters the world model M_{web}. Its job is to predict what will happen if a certain action is taken on the current page. For example, given the state s_t describing a page and an action $a_t =$ `click(#login-button)`, the world model might predict that the next state $\hat{s}_{t+1}$ will contain a login form (username and password fields) or that a new page will load with certain content. We can imagine M_{web} is implemented by a combination of learned models and rule-based simulations: it could have a neural network that, given the DOM and an action, outputs a predicted new DOM (or a description of changes), possibly using a library of known web interface templates. Alternatively, it might search a knowledge base of previously seen web navigation sequences to find what usually

happens when clicking a "login" button on similar sites (perhaps it usually leads to a page with "username" and "password"). In any case, $\hat{s}_{t+1}$ provides the planner with a lookahead: the planner can examine $\hat{s}_{t+1}$ to see if it aligns with the goal (in this example, if the goal is to make a purchase, then seeing a login form might be a necessary step, so it is expected and good). If the predicted outcome was something undesirable or irrelevant (say the click opens an ad popup or leads to a wrong page), the planner could reconsider and try a different action instead.

In practice, the reflective loop might work as follows at a given step. The planner Π_{web} considers all or a subset of possible actions (clicks or inputs it could do on the current page). For each candidate action a, it queries the world model M_{web} to simulate the result, getting $\hat{s}^a_{t+1}$. It then evaluates these predicted states for how promising they are with respect to the task. If one action clearly leads toward the goal, the planner selects it. If multiple seem plausible, the planner might use its internal knowledge (the language model might infer which sequence of steps is generally effective for the given task) or even engage in a brief internal debate (comparing options) to decide.

Once an action is chosen, the agent executes it in the real browser environment, obtaining the actual next state s_{t+1}. The actual state may differ from the prediction $\hat{s}_{t+1}$ if M_{web} is not perfectly accurate; such discrepancies will be valuable signals to improve M_{web} over time (the model can be updated or refined using this new real transition), and to continue the loop with s_{t+1} in place of s_t.

This web interaction scenario benefits greatly from the interplay of language reasoning and world modeling. The planner Π_{web} can harness its language understanding to read web pages (essentially performing comprehension of text on the fly) and to reason about user intentions or abstract instructions (e.g., if the user says "find me the cheapest flight," the agent knows it should use a flight search form and apply a price filter). The world model, grounded in the DOM and interface dynamics, provides a check on the planner's ideas: it ensures that the plan is viable in the real world of the web page by simulating it. Without M_{web}, a language model planner might hallucinate actions or outcomes that are not actually possible (for instance, expecting a button that doesn't exist or assuming a login will succeed without proper credentials). With M_{web} in the loop, such mistakes can be caught early: if the planner proposes to click a non-existent element, the model would fail to produce a coherent $\hat{s}_{t+1}$, cueing the planner to reconsider.

Moreover, the use of simulation enables long-horizon tasks to be broken down: the agent can plan a multi-step strategy (login, then navigate to product page, then add to cart, then checkout) by anticipating each stage with the world model's help. At any point, if the predicted path seems to veer off (e.g. after adding to cart, the predicted outcome doesn't show a checkout option due to being logged out), the agent can revise the plan.

In summary, the WebAgent scenario illustrates a language-based planner using reflective reasoning in a discrete, symbolic environment (the web UI). The planner's chain-of-thought is often explicit here (the agent can literally generate a step-by-step textual reasoning). The world model's role is crucially grounded: it deals with the actual structures and possible transitions of web pages, providing a form of **environmental grounding** to the otherwise free-form reasoning of the language model. This ensures the agent remains connected to reality and helps avoid the pitfalls of purely hallucinated plans. By the end of this project, I expect to have a WebAgent capable of handling non-trivial web tasks robustly, using far fewer real interactions thanks to extensive mental simulation via its world model.

4 Emergence of Specialized Roles through Co-Evolution of Planners

We consider a team of n planning agents that are initially identical in architecture and objectives, each operating over a shared world model. During training these agents *co-evolve* distinct roles (for example, explorer, critic, and primary planner) without manual role assignment, allowing the system to exploit complementary strengths. All agents access the same learned dynamics model M, yet their decision narratives diverge because each undergoes an independent, reflection-driven learning process. Random perturbations in early updates become amplified: one agent may gravitate toward exploratory behaviour, probing uncertain strategies; another becomes conservative, refining the most promising trajectory; a third adopts a critic stance, scrutinising plan coherence. This differentiation is not imposed but arises from the agents' coupled optimisation, which balances utility maximisation with inter-agent alignment.

Each agent maintains a language-based policy $\pi_i(\theta_i)$ that outputs a narrative update given its current narrative and simulated feedback from M. At every iteration an agent proposes a candidate trajectory, queries M for predicted outcomes, appends those outcomes to its narrative, and revises in a reflective loop. Small biases in these loops lead to divergent strategies that reinforce themselves through reward. Let $U(\tau)$ be the task utility of a joint trajectory τ and let $C(N_i, N_j)$ measure inconsistency between narratives N_i and N_j. Agent i optimises

$$J_i(\theta_1, \ldots, \theta_n) = \mathbb{E}_{\tau \sim (\pi_1, \ldots, \pi_n)}\big[U(\tau)\big] - \lambda \sum_{j \neq i} \mathbb{E}\big[C(N_i, N_j)\big],$$

where $\lambda > 0$ trades off global reward against harmful divergence. The associated policy gradient

$$\nabla_{\theta_i} J_i = \mathbb{E}_{\tau}\Big[\nabla_{\theta_i} \log \pi_i(\tau)\, U(\tau)\Big] - \lambda \sum_{j \neq i} \nabla_{\theta_i} \mathbb{E}\big[C(N_i, N_j)\big]$$

shows how each agent simultaneously pursues utility and preserves team coherence. Disagreement among agents furnishes intrinsic exploration signals, while utility divergence lets each agent specialise where its influence on U is greatest; reflection loops propagate insights so that specialisation remains beneficial to the whole.

An *"Aha" moment* occurs when one agent's narrative undergoes a sudden, large update $\Delta N_i^{(t)} = \|N_i^{(t)} - N_i^{(t-1)}\|$ exceeding a threshold, typically after integrating knowledge revealed by peers through the shared model M. Such events punctuate training as qualitative leaps in collective understanding: the explorer uncovers a high-value trajectory, the critic validates it, and the planner reorients around the new insight. Because these moments emerge from self-supervised reflection rather than external instruction, they demonstrate genuine role-based cognitive evolution within the team.

In sum, a multi-agent planner need not receive explicit role assignments. Through self-supervised disagreement, utility-driven differentiation, and reflection-mediated alignment, distinct roles arise naturally and stably. The resulting division of cognitive labour enables more robust and creative problem solving, with "Aha" moments marking key inflection points in the agents' shared learning journey.

Semantic Self-Alignment During RL Training

Dialogue between multiple large models at inference time is very popular these days: agents can critique each other's proposals and converge on polished answers with minimal engineering. Yet

this conversational fluency masks a deeper weakness in the *training* loop. During reinforcement-learning fine-tuning, the loss is usually nothing more than a weighted sum of a few rule-based rewards. Such a single scalar collapses the rich semantic space of feedback into one dimension, starving the optimiser of the nuance needed to refine higher-order behaviour. The result is an agent that can talk about subtle criteria at inference time but never *feels* those criteria in its gradient—a persuasive speaker trained on coarse applause.

The paradox is sharper when we recall that a pretrained LLM or VLM already contains a vast portfolio of expert skills. Prompt engineering can selectively "light up" those capabilities, proving that the knowledge is present, yet the training objective rarely encourages *communication across domains*. A technique activated while drafting legal text remains dormant when the model writes code; an insight mastered in image composition fails to surface during document layout. The internal knowledge graph resembles a city whose districts share no bridges.

Achieving genuine self-alignment—allowing ideas to migrate and recombine—therefore demands more than larger datasets or models. Classical empiricism reminds us that imagination is the recombination of abstractions already stored in memory; generalisation emerges not from conjuring the unprecedented but from arranging the known in novel constellations. If we want an LLM or VLM to move beyond likelihood-biased echoes of its corpus, we need a post-training regimen that rewards the *synthesis* of concepts drawn from different shelves of its library. Otherwise the model will excel only where pretraining handed it a ready-made template, falling back to safe paraphrase whenever it meets tasks that straddle domains.

In short, the bottleneck is not expressive capacity but the poverty of the optimisation signal. Without a learning objective that recognises and values cross-domain synthesis, latent expert skills remain siloed. The real challenge is to craft objectives rich enough to pressure the model into weaving those skills together—objectives whose gradients carry semantics, not just success/failure bits—so that the agent's imagination becomes an active recombinator rather than a sampler of isolated fragments.

Conclusion

The research vision outlined above aims to create AI agents with a powerful combination of imagination and reasoning. By tightly coupling a grounded world model with a language-based planner, we obtain a system that can simulate its environment and reflect on those simulations to make informed decisions—a truly *reflective reasoning* agent. The two application scenarios, in image editing and web interaction, demonstrate how this general principle can be applied to very different domains, one involving continuous visual data and the other involving discrete symbolic interfaces. In both cases, the agent's ability to iteratively reason (in latent or explicit form) and to foresee outcomes is key to handling complex, multi-step tasks that are beyond the reach of reactive or single-shot methods.

Over the next five years, my work will develop the theoretical foundations of this planner–world model loop, as well as practical algorithms and models to realize it. I will investigate how latent chain-of-thought representations can be learned and utilized, how to maintain coherence between the planner and world model as they co-evolve, and how multi-agent and feedback-driven training regimes can continuously enhance the agent's capabilities. Success will be measured by the creation of systems that are not only proficient in their domains but also offer insights into their decision-making process (via intermediate states or explanations) and can adapt/improve over time with minimal human intervention.

Ultimately, this research will contribute to the broader goal of building AI that can reason

and act in open-ended environments safely and effectively. The combination of a simulator (world model) and a reasoner (language planner) mirrors a common-sense approach to problem solving:

> Imagination is the beginning of creation. You imagine what you desire, you will what you imagine, and at last you create what you will. — George Bernard Shaw