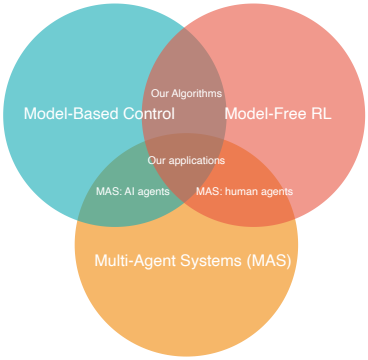
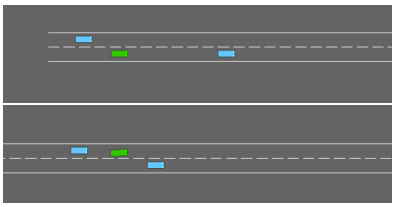
RESEARCH STATEMENT

In my doctoral research at University of Illinois Chicago, I have applied *model-based Reinforcement Learning* (RL), *switched control theory*, *stochastic game theory*, and *composite optimization* to develop algorithms that leverage model information to accelerate model-free training (Fig. 1a), achieving greater sample efficiency. The algorithms designed require fewer interactions with the environment to achieve the same level of optimality (Fig. 1b). Additionally, we extended the algorithm to multi-agent systems involving participants, where humans are treated as black-box agents (model-free) while other agents follow a model-based approach, enabling effective collaboration between the AI and human agents without the need to control or model human behavior (Fig. 1c).



(a) Model-based control and model-free reinforcement learning

(b) Filter and Filter w/o Ensemble are our algorithms



(c) Green: AI agents; Blue: Human agents. Initial position. The AI agent successfully changed the lane.

Figure 1: Summary of Research

Beyond these researches, which I correspond to, I am also interested in LLMs in decision-making. Together with researchers at Stony Brook University, we developed a hybrid system for end-to-end autonomous driving (Dong et al., 2024), integrating Large Language Models (LLMs) as high-level instructors to enhance the generalization capabilities of local end-to-end models. Our approach mitigates the latency impact of LLMs and is the first to apply LLMs to real-world, closed-loop, end-to-end autonomous driving (Fig. 2).

I am now exploring ways to generalize these methods beyond toy experiments to help data scientists, automotive engineers, journalists, and other end-users more easily train AI agents for deployment in real-world environments, such as autonomous driving, industrial robotics, AI for science, and healthcare.

Beyond research in academia, I am a co-founder of LIII NETWORK, an open-source company that developed GNU TEXMACS/Mogan, a WYSIWYG TEX-like editor enabling users to type equations 10 times faster—and Goldfish Scheme, a Scheme interpreter with a Python-like standard library. Goldfish Scheme is developed using literate programming, a methodology introduced by Knuth in 1984, which we believe will gain renewed relevance in the era of LLMs.

1 COMBINING MODEL-BASED CONTROL AND MODEL-FREE LEARNING

Many real-world nonlinear systems are challenging to model accurately. While linear approximation models simplify this process, they introduce model bias that can lead to highly suboptimal policies if used exclusively in model-based control. On the other hand, model-free methods require substantial data to learn an optimal policy, which can be impractical for physical systems. To address this issue, we developed several hybrid algorithms based on composite optimization and Dyna-style learning, which integrates model-based control using approximation models with model-free methods, requiring less data to achieve the same level of optimality as purely model-free approaches. **Gradient Compensation** (**Yansong Li** and Han, 2022) (GC) algorithm is a switched algorithm developed based on composite optimization, which integrates model-based control using linear approximation models with model-free methods We build a switching rule LSF that switches between the model-free gradient

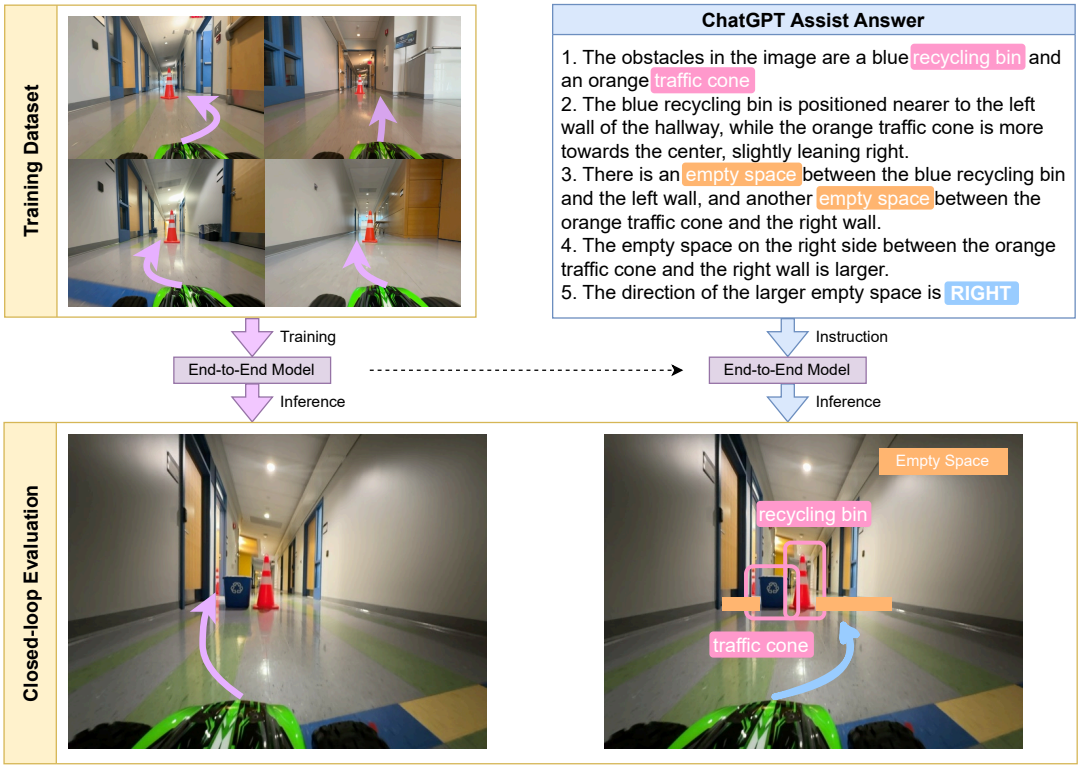


Figure 2: LLMs enhances the generalization of autonomous driving.

and the gradient computed from the linear estimated model. The gradient computed from the linear estimated model is updated whenever a model-free gradient is called.

Societal and potentially commercial impact Algorithms that combine model-based control with model-free learning methods are well-suited for application in complex, real-world environments. First, these algorithms require fewer interactions with the environment during training, making deployment more cost-effective and safer. Second, the model-based components offer interpretability, allowing for a clearer understanding of the AI agent's behavior and easier modification as needed. In the following sections, we list two applications for our algorithms in simulation environments.

Application: Continuous control on legged robots We developed a *Dyna-style learning with data filter* algorithm (**Yansong Li** et al., 2024), an extension of Dyna-style learning, which uses data generated from an estimated model to accelerate model-free training. Our modification introduces a filter that excludes data points likely to hinder rather than enhance training speed due to biases in the model's estimations. The algorithm was tested in the MuJoCo simulation environment, a platform for legged robot simulations. Our method demonstrates state-of-the-art (SOTA) performance in the model-based RL community, achieving comparable scores with fewer interactions (Fig. 1b).

Application: Multi-Agent Systems with Humans In a multi-agent system involving human agents, it can be challenging or impractical to model human behaviors using utility or transition models. One way to address this limitation is to treat human agents as black boxes, observing their actions without assuming a specific model—essentially treating them as model-free components within the system. Conversely, AI agents, which we aim to control, can be modeled explicitly and treated as model-based components within the multi-agent system. Based on this intuition, we developed an algorithm (**Yansong Li** and Han, 2023) for Stackelberg games in normal form that can effectively train an AI agent to collaborate with the follower (human agent) without modeling him.

The work is then extended to stochastic games with tabular setting (**Yansong Li** and Han, 2024).

2 LARGE LANGUAGE MODELS FOR DECISION MAKING

Algorithms that combine model-based and model-free methods form a type of hybrid system. We believe that hybrid systems are also essential for applying LLMs in decision-making. Our rationale is that while LLMs are not well-suited for direct control-level decision-making, they can support decision-making by providing higher-level instructions.

As shown in Fig. 2, we developed a hybrid system comprising LLMs for end-to-end autonomous driving. Unlike previous approaches, ours is the first to apply LLMs for closed-loop real-world driving. Previous work could not achieve closed-loop functionality in real-world environments due to the high latency of LLMs in giving a direct control signal. We address this challenge by decoupling the LLM model from direct action controls and creating an architecture that integrates multimodal LLMs with end-to-end autonomous driving. We evaluate the effectiveness of the proposed method in a closed-loop, real-world setting. The experiments show that LLMs enhance the generalization of the learning car, enabling it to drive in environments not encountered in the training set (Dong et al., 2024).

3 LARGE LANGUAGE MODELS FOR LITERATE PROGRAMMING

GNU TEXMACS/Mogan GNU TEXMACS is a WYSIWYG TEX-like editor that can help the user type equations 10 times faster than LATEX. However, GNU TEXMACS has several issues, which led us to create a fork called Mogan Research, aimed at addressing these problems and enhancing the user experience. Table 1 provides a comparison.

	GNU TEXMACS	**Mogan Research**
Release Cycle	yearly	monthly
Performance	slow	fast
Qt Framework	mainly in Qt 4 (some in Qt 5)	mainly in Qt 6 (some in Qt 5)
Scheme Engine	GNU Guile 1.8.x	S7 Scheme
Build Tool	GNU Autotools	xmake
Lead by	Joris van der Hoeven (Mathematician)	LIII NETWORK

Table 1: Comparison between GNU TEXMACS and Mogan Research

Goldfish Scheme and literate programming As shown in Table 1, the Scheme engine for TEXMACS is GNU Guile 1.8.x, which is no longer maintained by the GNU community. Consequently, the TEXMACS community decided to maintain their own Scheme interpreter. In contrast, Mogan uses the S7 Scheme as its foundation. However, the S7 Scheme also has limitations, such as incomplete support for R7RS and Unicode. To address these issues, we developed a new Scheme interpreter called *Goldfish Scheme*[1], based on the S7 Scheme. Goldfish Scheme is developed using a literacy programming method, first introduced by Knuth in 1984. Literate programming is a method where project code and its descriptions are written together in a single document. Code is presented in interconnected chunks, forming a narrative thread. A utility program then assembles these chunks into executable code, creating a structure that reads as a monologue and, at times, a dialogue. Mogan gives the utility program as a plugin. The literate programming document for Goldfish is available in `https://github.com/LiiiLabs/goldfish/releases/tag/v17.10.9` (Shen et al., 2024).

[1]Goldfish Scheme is named with a playful reference to the idea that goldfish have a 7-second (7s) memory, the inverse of S7. In the future, we hope LLMs can help create new functions for Goldfish Scheme within those same 7s.

Literate programming in the era of LLMs We believe that literate programming will be revived in the era of LLMs. Our conjecture is that:

> **Projects developed using literate programming are inherently more understandable for LLMs than projects containing even the most detailed comments.**

We solidify this conjecture with an example, which, due to its technical details, is provided at the end of this statement.

FUTURE RESEARCH DIRECTIONS

How can we deploy learning-based decision-making policies in real-world real-time? Deploying learning-based decision-making policies in real-world real-time faces the following challenges:

- *Data requirements*: Training learning-based policies requires a large amount of data from interactions with the real world.
- *Real-time performance*: Many learning-based policies involve complex computations that may not meet the time constraints required for real-world decision-making.
- *Adaptability to environmental changes*: Many learning-based models struggle to generalize to new situations that were not part of the training data.

Algorithms that combine model-based control with model-free learning address issues related to data requirements, while the LLM-assisted driving framework (Dong et al., 2024) we developed tackles challenges in real-time performance and adaptability to environmental changes. Both are hybrid systems. However, beyond the two problems mentioned above, several other challenges, such as *exploration vs. exploitation balance* and *safety and stability*, have not been addressed in my Ph.D. research. I will continue collaborating with researchers in robotics, control theory, and game theory to further explore these topics and integrate them with my previous research, aiming to develop a more robust AI agent adaptable to real-world applications.

Can LLMs help for decision-making in the real world? We believe that LLMs can assist in decision-making rather than making decisions directly for several reasons:

- *High Latency:* LLMs require substantial computational power for inference, resulting in latency that can slow down real-time decision-making processes.
- *Limited Context Awareness:* LLMs often lack full awareness of real-time context or situational changes, which can lead to suboptimal decisions if used directly in dynamic environments.
- *Reliability Concerns:* Directly relying on LLMs for decision-making can introduce unpredictable behavior, as they may produce varied responses based on slight changes in input, making them less reliable for critical decisions.

Guided by this belief, enabling LLMs to assist in decision-making requires the design of hybrid systems. At a high level, algorithms that combine model-based and model-free methods are also hybrid systems, allowing ideas from my previous research to be applied. I will collaborate with other researchers to explore this topic further.

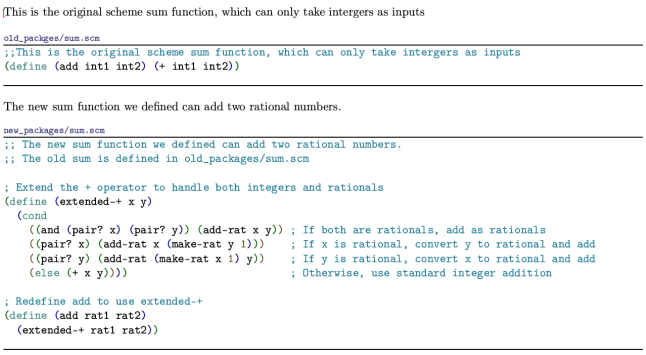
Renaissance of Literate Programming in the Era of LLMs As discussed in Sec. 3, we believe that Literate Programming will be revived in the era of LLMs. To illustrate, we present an example contrasting code with detailed comments and code written in the style of literate programming. As discussed in Sec. J of Knuth (1984), literate programming enables a flexible structure where code is organized as a "web" of interconnected ideas rather than following a strict top-down or bottom-up sequence. Fig. 3a illustrates this concept with two files connected by ideas. The file `old_packages/sum.scm` contains the original sum function in Scheme, which only accepts integer inputs. The file `new_packages/sum.scm` extends this sum function to accept rational numbers. These two files are linked by ideas rather than function calls or class inheritance. Fig. 3a demonstrates how LLMs can directly recognize these connections, as literate programming allows these chunks to be organized this way. In

contrast, Fig. 3b shows the same project with detailed comments. Even with such comments, LLMs must read and interpret the paths mentioned within, locate the referenced file, and then analyze it to grasp the connection.

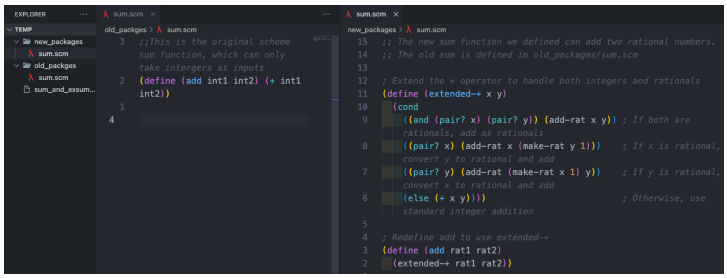
This falls into a research area called "prompt engineering". The motivation behind developing Goldfish Scheme for literate programming and prompt engineering experiments includes:

- *Simplicity for Inference*: The Scheme's minimalistic design limits the options for inference, allowing LLMs to provide more precise and straightforward answers.
- *Python Compatibility for Knowledge Transfer*: LLMs are generally more familiar with Python due to extensive training in Python code. Since the Goldfish Scheme includes several features from Python's standard library, it enables smoother knowledge transfer from Python to the Goldfish Scheme, making it more accessible for LLMs than other Scheme variants.

However, literate programming faces several challenges, such as difficulties with collaboration, version control, and establishing clear writing standards, which are harder to define than coding standards. Together with my colleagues at LIII NETWORK, we will continue exploring the potential of literate programming and LLMs to address these challenges.



(a) Two idea-connected files edited in Mogan using literate programming; LLMs can directly read the connection.



(b) Connection between these two files are commented in `new_packages/sum.scm`. Even with such comments, LLMs must read and interpret the paths mentioned within, locate the referenced file, and then analyze it to grasp the connection.

Figure 3: Literate programming enables code to be organized as a "web" of interconnected ideas.

REFERENCES

Dong, Z., Zhu, Y., **Yansong Li**, Mahon, K., and Sun, Y. (2024). Generalizing end-to-end autonomous driving in real-world environments using zero-shot LLMs. In *8th Annual Conference on Robot Learning*.

Knuth, D. E. (1984). Literate programming. *Comput. J.*, 27(2):97–111.

Shen, D., Liu, N., **Yansong Li**, Wang, D., He, L., and Zhou, Y. (2024). *Goldfish Scheme: A Scheme Interpreter with Python-Like Standard Library*. LIII NETWORK.

Yansong Li, Dong, Z., Luo, E., Wu, Y., Wu, S., and Han, S. (2024). When to trust your data: Enhancing dyna-style model-based reinforcement learning with data filter.

Yansong Li and Han, S. (2022). Accelerating model-free policy optimization using model-based gradient: A composite optimization perspective. In Firoozi, R., Mehr, N., Yel, E., Antonova, R., Bohg, J., Schwager, M., and Kochenderfer, M. J., editors, *Learning for Dynamics and Control Conference, L4DC 2022, 23-24 June 2022, Stanford University, Stanford, CA, USA*, volume 168 of *Proceedings of Machine Learning Research*, pages 304–315. PMLR.

Yansong Li and Han, S. (2023). Solving strongly convex and smooth stackelberg games without modeling the follower. In *American Control Conference, ACC 2023, San Diego, CA, USA, May 31 - June 2, 2023*, pages 2332–2337. IEEE.

Yansong Li and Han, S. (2024). Efficient collaboration with unknown agents: Ignoring similar agents without checking similarity. In Dastani, M., Sichman, J. S., Alechina, N., and Dignum, V., editors, *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024*, pages 2363–2365. International Foundation for Autonomous Agents and Multiagent Systems / ACM.