Pomona College
Department of Computer Science

Model Predictive Control in Video Game Environments

Rona Linarez

May 09 2025

Submitted as part of the senior exercise for the degree of
Bachelor of Arts in Computer Science

Professor Anthony Clark, advisor

Abstract

Many state-of-the-art AI models rely on large volumes of labeled training data; however, few large, labeled datasets exist for popular video games. This paper explores the use of Model Predictive Control (MPC) to play the retro video game F-Zero, with the goal of generating high-quality labeled gameplay data. Our system reframes the game as an optimization problem to streamline the creation of meaningful training data. By leveraging emulators to access low-level game state information, this approach offers a scalable alternative for dataset generation. This work represents an initial step toward developing tools and methodologies that support video game research and applications of AI in gaming environments.

Acknowledgments

I would like to thank Anthony Clark for his immense support and guidance during this project. I would also like to thank Joseph Osborn for encouraging me to follow my passion for computer science and video games.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A significant limitation in video game research and experimentation is the lack of high-quality labeled training data. This is largely due to the immense volume of possible game states which individually depend on user-controlled actions [Ha and Schmidhuber(2018)]. Hand labeling video game data is expensive and impractical, and alternatives like reinforcement learning only provide a limited subset of game information. The current direction of artificial intelligence research demands large volumes of data, and developing AI models capable of mastering video games remains a crucial milestone toward achieving generalized artificial intelligence [Perez-Liebana et al.(2016)]. Closing this gap of data within video game environments is thus crucial to promoting research done with games as well as providing rich data for use in AI contexts. This thesis investigates the problem of limited data sets by posing retro games as optimization problems. This work aims to explore the feasibility of applying traditional control methods to solve video games optimally, such that high quality data can be recorded without the need for human players. The aim in this paper is to create a model that can independently navigate around a racetrack in F-Zero, and to create labeled training data from this model's results.

F-Zero is a 90's racer in which the player is tasked to complete 5 laps on a track while avoiding enemy racers. The game features various obstacles and complex track designs, and offers a simple control scheme which is easy to learn but hard to master. In this work, we apply control techniques to develop an agent capable of playing the game smoothly and effectively. The resulting model is designed to facilitate straightforward generation of high-quality labeled data based on its own gameplay decisions. The motivation for this work lies in leveraging optimization-driven approaches to achieve two

Figure 1.1: Getting ready for a hairpin turn in F-Zero

goals: to explore methodologies for conducting research using video games, and to advance AI techniques that depend on the structured generation of training data. By integrating control theory and optimization into a video game environment, my project aims to provide tools and insights for researchers addressing challenges in retro gaming as well as a general audience of AI researchers. The remainder of this paper will explore past work in this field, the proposed project methodology, and specifications on how to achieve our research goal.

Chapter 2

Background and Related Work

Research focused on mastering video games autonomously is relatively new, yet the techniques employed have evolved significantly over the past decade. Each approach offers unique strengths and limitations, which have shaped the field's progress. The NeuroEvolution of Augmenting Topologies (NEAT) [da Silva Miras de Araujo and de Franca(2016)] algorithm pioneered early efforts in video game playing. NEAT employs an evolutionary approach that iteratively refines models by selecting the best-performing agents out of many, evolving their approach over many generations to produce networks optimized for specific tasks [da Silva Miras de Araujo and de Franca(2016)]. NEAT has seen applications in famous retro video games, where 2D tile-based graphics provide enough feedback to create effective models with this approach. While NEAT can be effective in many scenarios, this technique struggles with generalization. NEAT often over-fits to specific environments, even within the same iteration. This limitation severely impacts NEAT's ability to play video games autonomously for a long duration. The Evo-Man competition is an example of a video game testbed designed for AI researchers, and NEAT dominated as the preferred learning technique within the pool of successful agents. However, even the best performing genetic models struggled when facing new bosses or attack patterns in the game [Cojocaru et al.(2020)]. Additionally, the computational demands of training NEAT are high, and performance quality requires significant training time to produce effective results. As such, alternative models gained more popularity and widespread use.

To address the highly nondeterministic action spaces of video games, research shifted towards deep learning methods. Deep neural networks (DNNs) are capable of creating more complex agents, though they require

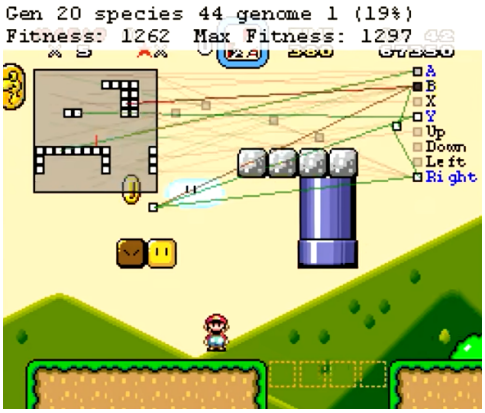

Figure 2.1: An example of NEAT being used to play Super Mario World

computationally intensive training. Despite this, DNNs still share difficulties with generalization, and struggle to account for environments that were never seen before [Gupta et al.(2021)]. Some methods leverage Reinforcement learning as a complementary approach to mitigate these limitations; reinforcement learning allows an agent to learn policies adapted to dynamic environments with learned experience. Other machine learning techniques like variational auto-encoders (VAEs) further advanced model and state estimation, demonstrating their utility in generating high-quality labeled data [Ha and Schmidhuber(2018)]. An essential component of advancing neural network research leverages reinforcement learning to accomplish better generalization; Ha and Schmidhuber demonstrate a recurrent world model helps in simulating environments efficiently for policy evolution, and collect labeled training data from reinforcement learning agents to train visual encoders. Building the necessary training data in this way serves as the inspiration for my paper.

Building on this foundation, Yunlong Lu's work [Lu and Li(2024)] modeled Mahjong, a game with a lot of randomness and incomplete information which affect the optimal player strategy. This work quantitatively compared the efficacy of various AI models, finding trade offs between performance and training time. Expanding on this work, Diego et al. [Perez-Liebana et al.(2016)] extended their analysis from traditional games to video games, highlighting the shift from structured environments to the more complex and chaotic state spaces characteristic of video games. Creating a generalizable agent capable of learning how to play all types of games is an open problem in this field, and multi-agent approaches serve as key steps towards reaching

this goal.

The intersection of reinforcement learning and multi-agent systems has also gained traction, as explored in works by Jack Serrino [Serrino et al.(2019)]. These approaches emphasize interdisciplinary methods, combining evolutionary algorithms, psychology, and sociology to improve agent reasoning in scenarios with sparse rewards and cooperative challenges. Tongtong et al.'s seminal work [Yu et al.(2024)] demonstrates a deep learning algorithm paired with reinforcement learning to teach agents to play complex games like StarCraft, offering a large step up in terms of game complexity (most video game work in AI tests on games made before the 2000's). Their model, M2RL, learns policies in game environments with up to 16 different agents playing simultaneously. With the goal of learning complex game objectives when reward is sparse and the goal of various agent's are uncertain, M2Rl offered a step forward in building a human-like game playing agent.

The focus on my paper lies between the middle of these two achievements, focusing on a 1990's racing game with several racers competing with the player. Techniques such as race-line optimization [Vesel(2015)] are applied to the real-world analogue to this game in autonomous racing. Vesel's work optimizes racing trajectories using a simplified model of a vehicle, serving as important groundwork in the construction of my model. Autonomous racing synthesizes many cutting edge control techniques in the aim of creating competition viable self-driving vehicles [Tătulea-Codrean et al.(2020)], These frameworks provide key insights into achieving effective and efficient agents. My paper builds on these foundations, aiming to integrate MPC [Williams et al.(2016)] to create a versatile agent capable of producing meaningful, labeled data of game play. The biggest challenge to bridge control theory and video games will be fulfilling the model requirements in a simulated space with limited sensing elements. Autonomous vehicles have access to many position sensing equipment that allows for constant checks to the vehicle's position error to be made. Creating effective approximations for these in our game poses a significant challenge, especially given the limited information provided to the player during gameplay. While this limitation does pose a tangible restriction on the accuracy of our model, we want to show in this work that the results produced by applying control theory are comparable to human game play.

Chapter 3

System Overview

3.1 Methodology

We implement Model Predictive Path Integral control to optimize game play in F-Zero, modifying the application of the algorithm to fit in a video game context. The control algorithm requires many state variables in order to accurately predict a system's behavior, which necessitated hardware-level access to the video game's memory. We conduct the development of this model within a Super Nintendo emulator in order to retrieve this key state information. This chapter describes the development of the agent in detail. The methodology is divided into three key sections: model prerequisites, emulation, and system development.

3.1.1 Model Prerequisites

To apply MPC in F-Zero, a kinematic model of our vehicle is required, along with a state estimator. However, players do not normally have access to vehicle data while playing; the game keeps track of key data in the console's RAM, such as speed, velocity and player position. We can access these values by reading the RAM values in real time to build visual approximations of track positions. Confirming the location of these values in RAM is a prerequisite step towards constructing an accurate model of the game.

Bicycle Model

We will be using the bicycle model, a simplified representation of vehicle kinematics often used for vehicle control [Soudbakhsh and Eskandarian(2012)]. This model allows us to simplify F-Zero's movement physics, while provid-

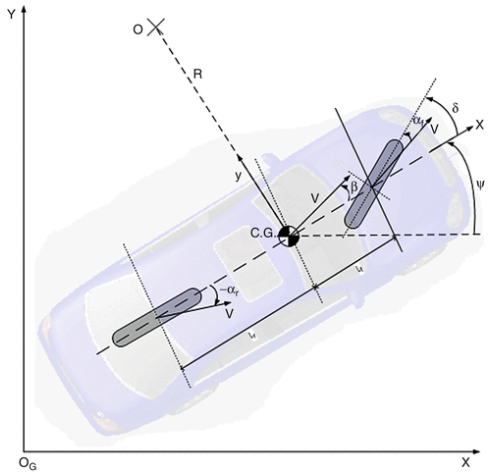

Figure 3.1: A diagram of the bicycle model's various state variables

ing the necessary movement variables required for Model Predictive Path Integral Control. The bicycle model approximates a four-wheeled vehicle as a two-wheeled system with a front and rear axle, which sufficiently maintains accuracy for most use cases. Essential parameters are preserved, such as the position, orientation, velocity, and steering angle of the vehicle, as well as physical characteristics like wheelbase length and maximum steering angle; F-Zero is not a racing simulator, hence using a simplified model like this is more than sufficient for our research goal. For our work, the bicycle model will describe the in-game vehicle dynamics, providing a mathematical framework for trajectory optimization and control. To construct the model, we require information such as the vehicle's speed, acceleration and deceleration, and yaw, which will be extracted or approximated from the game environment, which are not normally accessible to the player. As such, we need to leverage game emulation to retrieve these values.

We will be using the Bizhawk emulator to simulate the Super Nintendo console. Bizhawk is typically used for creating tool-assisted playthroughs of games, and provides special tools that give the user access to high levels of control [2]. Video game emulation mimics game console hardware, allowing us to play a legitimate copy of F-Zero via software. The Bizhawk emulator offers many tools that allow us to access and search through memory while the game is running, as well as allowing us to increment the game frame by frame (akin to stepping through a program via a debugger). To

[2]The Bizhawk emulator can be found at https://tasvideos.org/Bizhawk

extract key game state information without approximating from on-screen visuals, we leverage Bizhawk's built in support of scripting via Lua. The Lua programming language is designed for embedded programming, and is already integrated within the Bizhawk emulator to execute user-generated code while a game is running without modifying the original game code. This functionality allows us to observe and access valuable game data, as well as build code on top of the game without disrupting the game's execution.

Emulation

11	7E:0B20	2 bytes	Machine speed.
12	7E:0B30	2 bytes	Machine X velocity, in pixels. Signed.
13	7E:0B40	2 bytes	Machine X velocity, in subpixels.
14	7E:0B50	2 bytes	Machine Y velocity, in pixels. Signed.
15	7E:0B60	2 bytes	Machine Y velocity, in subpixels.
16	7E:0B70	2 bytes	Machine X coordinate. Wraps around automatically at #$1FFF.
17	7E:0B80	2 bytes	Machine X sub-coordinate.
18	7E:0B90	2 bytes	Machine Y coordinate. Wraps around automatically at #$0FFF.
19	7E:0BA0	2 bytes	Machine Y sub-coordinate.
20	7E:0BB0	2 bytes	Machine height gain. Signed.
21	7E:0BC0	2 bytes	Machine height.
22	7E:0BD0	2 bytes	Machine angle facing. Wraps around automatically at #$BFFF.
23	7E:0BE0	2 bytes	Machine velocity direction. Wraps around automatically at #$BFFF.
24	7E:0BF0	2 bytes	Rebound direction of last wall hit. Actually unused.

Figure 3.2: A RAM map for the game physics available on a F-Zero wiki

The use of an emulator is necessary to find the necessary state information needed for our control algorithm. However, the player is provided with minimal information during normal game play, which is not sufficient to predict optimal movement. With Lua scripts, we gather game state information (e.g., vehicle position, velocity, and steering angle) without needing direct access to the game's source code. Using Bizhawk's RAM searching tool, we can locate specific bytes and addresses in working RAM, and store them as Lua variables for our model to use later. We do not need to locate the byte-addresses for all of the values on our own: with the assistance of user created RAM maps documenting the game's code and variable locations in memory, we are able to retrieve all of the necessary information for our model. These maps are essential for the completion of this research, because we'd otherwise need to search for these values through RAM ourselves, or generate estimates for the required values. Having the location of variables in RAM is one of the biggest hurdles for conducting research on retro video games, though some games do provide players with direct access to positional and telemetry data out of the box.

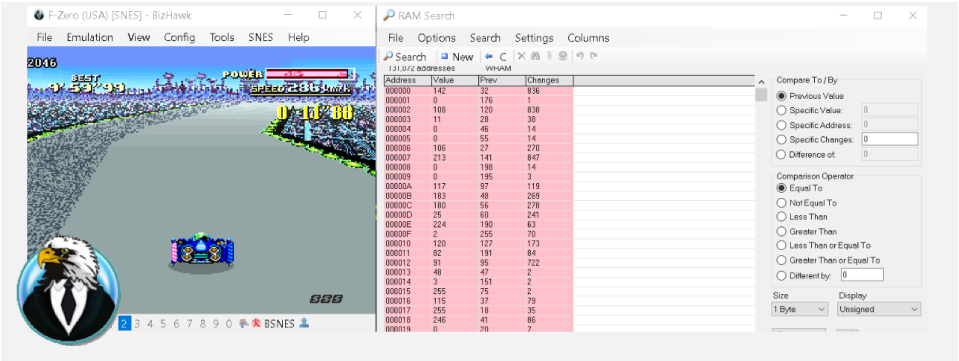

Figure 3.3: The Bizhawk emulator running zero while performing a RAM search of the game's current memory

3.2 System Development

In order to satisfy the constraint of not modifying the original game code, we separate the development of our controller from the emulator. This approach makes the application of the control method simpler by allowing us to use any implementation of MPC that we prefer, but comes at the downside at requiring us to create a communication link between the client (the game) and the the controller.

3.2.1 Control Method

MPC controller's are designed to navigate highly nondeterministic environment by predicting future states based on current observations. [Rawlings et al.(2017)] We use MPPI control to complete this task, due to its ability to sample future states with a simplified model of the environment. The MPPI controller, based off a paper implementation in python takes these values and returns an optimal instruction in the form of steering angle and acceleration [1], which can be mapped onto the game via emulation tools. This control method requires the current game state information from our Kinematics model, as well as in-game estimations of the surrounding track in order to properly adjust and steer the vehicle onto the pre-determined optimum path. In order to create this path, we sample the vehicle's position as it goes around a track in real time. We created a sampling program in Lua which

[1]The GitHub repository for the MPPI controller can be found at `https://github.com/MizuhoAOKI/python_simple_mppi`

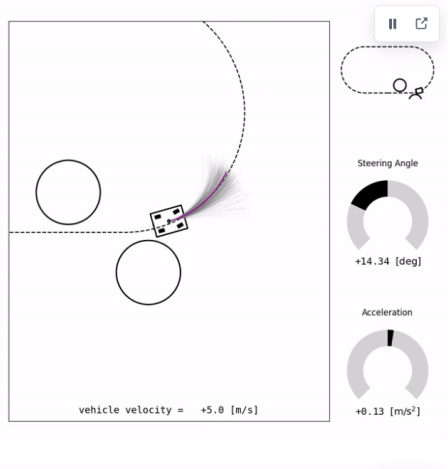

Figure 3.4: A demo of the MPPI controller running in Python

records the X and Y values of the vehicle as it steps through every in-game frame, and stores the results in a `.csv` file. This format was specified by our controller, and allowed us easy access to the optimal line.

3.2.2 Communication between client and the controller

In order to send the recorded values from our game into the controller, and send the optimal input decision back to the emulator, we set up a python server to act as a bridge between the client and the controller.

The flow is as follows: The client runs the game, and records the current state. The emulator then pauses the game execution, which effectively stops time within the game itself, and sends these values to the server via a POST request, A GET request is made for the next decision to make in-game, which will be answered by the controller. The server makes a call to the MPPI controller, which processes the data and returns an input decision. This result answers the GET request, and is processed by the client before repeating once again.

3.2.3 Creating Input Heuristics

When playing F-Zero normally, the player can left or right to move their vehicle, and press the accelerator to move their vehicle forward. These values are binary and not analog, meaning to achieve precise movement and

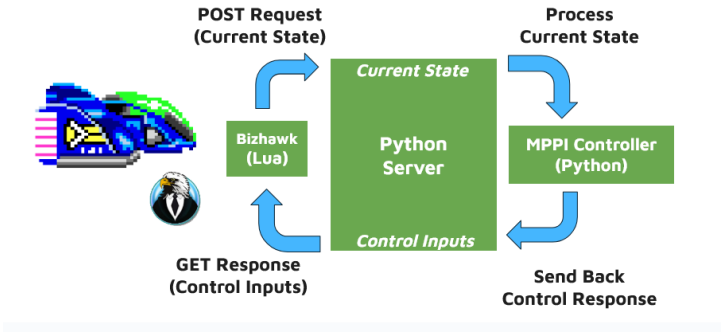

Figure 3.5: An overview of the communication system between the game and the controller

in-between states, a player must tap and release the respective buttons until the desired position is acquired. Notably, the MPPI controller provides exact steering angle and acceleration values for the client to utilize. This provides a significant hurdle in the interaction between the client and the controller, which must be corrected through approximations of the given control response. Our model utilizes sharp thresholds that trigger the activation or release of buttons for acceleration. However, in order to provide smooth and continuous turning, Pulse Width Modulation (PWM) can be employed to simulate more precise steering angles.

Pulse Width Modulation is a technique that controls the amount of power delivered to a device by rapidly switching the signal on and off at a fixed frequency. The ratio of the "on" time to the total cycle time determines the output level. For example, a higher frequency cycle results in more power or input being delivered, while a lower duty cycle reduces it. In the context of video game control, PWM can be used to simulate analog-like behavior on digital inputs by adjusting the duration and frequency of button presses.

The benefit of this technique is that it we can create more accurate translations of the model's values in the game, allowing for smoother and more natural movement that closely mimics human input. This is particularly useful for fine-grained control tasks like steering, where binary on/off signals are insufficient for producing realistic gameplay behavior.

Chapter 4

Results

We produced a model capable of playing F-Zero independently. However, this model does not navigate the course effectively, and is prone to many driving mistakes. One of the biggest reasons for this is the rough translation between the game environment and the controller's simulated environment. This can be improved by adjusting either the model parameters or the emulator's input translation heuristics.

Fortunately, creating a data set using this model is straightforward and fast. In a similar way that we used a sampler to acquire the positional values of an in-game path, we are able to extract the player inputs aligned to positional coordinates. This data can easily be saved in the `.csv` format, which is satisfactory for our research goals, Despite our model being highly game specific, the approach we use to sample inputs is flexible and can easily be ported to other games on the same system.

While video games provide rich tests beds for data creation and collection, the question as to what the best way to harvest this data remains highly specific to each and every game. Through this work, it's clear that more tools and resources are needed to streamline the process of acquiring essential data from games. Even with this area covered, the question as to what form the data should take is difficult to answer; positional coordinates aligned to player inputs could serve as a good starting point, but also depends on the game and the clear conditions within a certain game. Racing games are a more straightforward in this regard, since the goal does not move while the player makes their way towards it. More work needs to be done to discover what format an input-aligned dataset should take to best suit specific game genres, as well as other applications of control methods

to solve common video game obstacles.

An ongoing debate in the generative AI space involves the cost and resource heavy training needed to feed these models. Finding new and cheaper ways to create training data is a crucial problem that requires further discussion. MPC is less resource intensive than GAI methods, due to the fact that no prior training is required. However, MPC does require a significant more amount of fine tuning and testing to achieve results. New research is currently exploring the use of MPC with applied RL, which could provide the best of both worlds in terms of efficiency and generalization.

There are many ways to get labeled video game data, and this paper serves to offer a new method that does not require high computation power or test subjects. We can thus efficiently and systematically generate training data while avoiding the ethical pitfalls associated with human data collection, such as privacy violations or the exploitation of test subjects.

This approach underscores the importance of exploring alternative, low-cost methods that align with broader efforts to reduce the environmental and economic impact of AI research. Moreover, MPC deviates from the current standard of applying GAI methods to solving games, and offers a new framing for game playing not currently in use.

As video games become more integrated into AI development for broader applications, including robotics, education, and simulation training, ensuring ethical and sustainable practices in data generation will remain a critical concern. This paper aims to contribute by demonstrating a responsible approach to solving the data vacuum affecting the creation of more problem-solving AI agents that can beat video games. [Valevski et al.(2024)]

This paper provides a novel framework for leveraging control theory in video games to address the need for high-quality game data sets, bridging the gap between classical optimization techniques and modern AI approaches for video game research.

Chapter 5

Future Work

The data produced by this model can be leveraged to train a variety of machine learning algorithms. The combination of position coordinates, speed, yaw, and button inputs offers a clean and structured framework for model development. In particular, models that benefit from structured datasets during training, such as warm-start reinforcement learning, are well-suited to this kind of data. Neural network-based approaches would also benefit, especially given how easily large volumes of data can be collected using this method.

A valuable extension of this work would be a tool that allows users to customize which variables and button inputs are recorded in the resulting `.csv` file. In the future, I aim to develop such a tool, leveraging existing resources to streamline the setup of state representations for popular video games. This tool would allow researchers to define which in-game variables and control inputs are sampled and exported, enabling more targeted and efficient data collection. By making the process of dataset generation more accessible and customization, this tool could greatly increase the volume and variety of datasets produced from gameplay. Such an approach would not only accelerate experimentation in machine learning applications using video game environments, but also contribute toward standardizing how gameplay data is collected and formatted. This standardization is crucial for benchmarking algorithms and enabling reproducibility across different research projects.

This work demonstrates how emulated video game environments can serve as rich, controllable platforms for generating training data for machine learning. By capturing a game's internal state and aligning it with corresponding button inputs, we provide a practical bridge between simulation-

based learning and real-world applications. Looking forward, expanding the tools and infrastructure for gameplay data collection will be essential for enabling more reproducible, scalable, and insightful research. As video games continue to evolve as testbeds for artificial intelligence, this work represents an early but meaningful step toward making these environments more accessible for data-driven experimentation.

Bibliography

[Cojocaru et al.(2020)] Gabriel-Codrin Cojocaru, Sergiu-Andrei Dinu, and Eugen Croitoru. 2020. Increasing the Upper Bound for the EvoMan Game Competition. In *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. 231–237. https://doi.org/10.1109/SYNASC51798.2020.00045

[da Silva Miras de Araujo and de Franca(2016)] Karine da Silva Miras de Araujo and Fabrício Olivetti de Franca. 2016. Evolving a generalized strategy for an action-platformer video game framework. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. 1303–1310. https://doi.org/10.1109/CEC.2016.7743938

[Gupta et al.(2021)] Dhawal Gupta, Gabor Mihucz, Matthew Schlegel, James Kostas, Philip S. Thomas, and Martha White. 2021. Structural Credit Assignment in Neural Networks using Reinforcement Learning. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 30257–30270. https://proceedings.neurips.cc/paper_files/paper/2021/file/fe1f9c70bdf347497e1a01b6c486bdb9-Paper.pdf

[Ha and Schmidhuber(2018)] David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf

[Lu and Li(2024)] Yunlong Lu and Wenxin Li. 2024. Mahjong AI Competition: Exploring AI Application in Complex Real-World Games. In

Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, Kate Larson (Ed.). International Joint Conferences on Artificial Intelligence Organization, 8733–8736. https://doi.org/10.24963/ijcai.2024/1020 Demo Track.

[Perez-Liebana et al.(2016)] Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, and Simon Lucas. 2016. General Video Game AI: Competition, Challenges and Opportunities. *Proceedings of the AAAI Conference on Artificial Intelligence* 30, 1 (Mar. 2016). https://doi.org/10.1609/aaai.v30i1.9869

[Rawlings et al.(2017)] J.B. Rawlings, D.Q. Mayne, and M. Diehl. 2017. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing. https://books.google.com/books?id=MrJctAEACAAJ

[Serrino et al.(2019)] Jack Serrino, Max Kleiman-Weiner, David C Parkes, and Josh Tenenbaum. 2019. Finding Friend and Foe in Multi-Agent Games. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, and E. Foxand R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/912d2b1c7b2826caf99687388d2e8f7c-Paper.pdf

[Soudbakhsh and Eskandarian(2012)] Damoon Soudbakhsh and Azim Eskandarian. 2012. *Vehicle Lateral and Steering Control*. Springer London, London, 209–232. https://doi.org/10.1007/978-0-85729-085-4_10

[Tătulea-Codrean et al.(2020)] Alexandra Tătulea-Codrean, Tommaso Mariani, and Sebastian Engell. 2020. Design and Simulation of a Machine-learning and Model Predictive Control Approach to Autonomous Race Driving for the F1/10 Platform. *IFAC-PapersOnLine* 53, 2 (2020), 6031–6036. https://doi.org/10.1016/j.ifacol.2020.12.1669 21st IFAC World Congress.

[Valevski et al.(2024)] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. 2024. Diffusion Models Are Real-Time Game Engines. arXiv:2408.14837 [cs.LG] https://arxiv.org/abs/2408.14837

[Vesel(2015)] Ricky Vesel. 2015. Racing line optimization @ race optimal. *SIGEVOlution* 7, 2–3 (Aug. 2015), 12–20. https://doi.org/10.1145/2815474.2815476

[Williams et al.(2016)] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. 2016. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 1433–1440. https://doi.org/10.1109/ICRA.2016.7487277

[Yu et al.(2024)] Tongtong Yu, Chenghua He, and Qiyue Yin. 2024. M2RL: A Multi-player Multi-agent Reinforcement Learning Framework for Complex Games. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, Kate Larson (Ed.). International Joint Conferences on Artificial Intelligence Organization, 8847–8850. https://doi.org/10.24963/ijcai.2024/1046 Demo Track.