ETH zürich

Institute for Dynamic Systems and Control

Philippe Brigger

MPC-Imitation Learning for a Labyrinth Game

Semester Project

Institute for Dynamic Systems and Control
ETH Zurich

Supervision

Prof. Dr. Raffaello D'Andrea
Thomas Bi

December 2023

IDSC-RD-TB-01

IDSC

Abstract

This semester project aims to achieve autonomous guidance of a ball through a BRIO labyrinth using a neural network. The neural network is trained through imitation learning, specifically by emulating a model predictive control (MPC) expert. The project combines an online and offline imitation learning pipeline to a mixture-of-experts model.

In the online pipeline, state trajectories are gathered by the neural network in real-time on the physical testbench. These trajectories are then forwarded to the MPC expert, who calculates optimal control inputs. Subsequently, the neural network is trained using the acquired state and control input pairings. This incremental learning process enables the neural network to progressively master the task of guiding the ball through the labyrinth. Conversely, the offline pipeline utilizes an augmented offlined dataset from a successful MPC run.

From fifteen trial runs the neural network completed 64.4% of the labyrinth, while the expert solved 61.5% on average. The expert had a success rate of 20.0% and took 57.8 s on average for a successful run, while the neural network completed the labyrinth 13.3% of the time in 59.0 s. The neural network took 1 ms with a standard deviation of 0.1 ms to calculate a control input whereas the expert took 46 ms with a standard deviation of 32 ms. Hence, the advantage of employing imitation learning lies in its ability to enhance labyrinth control at a significantly higher bandwidth compared to the expert, while maintaining a comparable success rate.

Keywords: Imitation Learning, Labyrinth Game.

Contents

Nomenclature

GPU - Graphics Processing Unit
IDSC - Institute for Dynamic Systems and Control, ETH Zurich
LQR - Linear quadratic regulator
MAF - Moving average filter
MPC - Model-predictive control
NN - Neural Network
PID - Proportional, integral, derivative (regulator)
ROS - Robot Operating System
USB - Universal Serial Bus

Chapter 1

Introduction

The BRIO labyrinth, a commercially available game, involves guiding a metal ball through a maze of holes and walls by tilting the board along two axes. Traditionally played by humans, this game serves as an intriguing platform for testing model-based and learning-based control systems [1].

Several approaches have been explored for autonomously controlling labyrinth games. In one instance, a PID controller is initially employed to generate system data. The inverse dynamics of the system are then learned, enabling labyrinth control via a locally-weighted-projection-regression method [2]. Another approach utilizes a gain-scheduled LQR controller, attempting to follow a spline path through the maze [3]. A hierarchical learning architecture is presented in [4], where the labyrinth is learned region by region. In a notable advancement, a data-augmented, model-based reinforcement learning controller outperforms the best human player in solving the labyrinth [5].

Model-predictive-control (MPC) calculates optimal control inputs respecting system and environmental constraints. An MPC controller was developed for the labyrinth game by a student in a different semester project at IDSC. The model utilizes a ball-on-plate model for system dynamics and plans a path that avoids nearby holes and walls. However, the computational expense of MPC makes real-time implementation challenging. One strategy to enhance computational efficiency without sacrificing accuracy is through imitation learning.

Imitation learning, a domain in machine learning, involves a robotic system mimicking demonstrations by an expert, offering an intuitive approach to train autonomy [6]. The imitation pipeline comprises collecting data from the expert, preprocessing the data, and training a model to replicate the expert's behavior. Imitation learning has been applied to various domains, including 3D games [7], autonomous driving, robotic simulations, and object manipulation [8].

In this semester project, labyrinth control is achieved through a neural network trained to imitate the MPC controller. Control input labels are calculated using the MPC solver from a dataset, and the neural network is trained in a supervised fashion. The resulting neural network controller performs comparably to the MPC while significantly increasing the controller bandwidth.

Chapter 2

Methodology

2.1 Preliminaries

2.1.1 Labyrinth Game

The labyrinth testbench along with its coordinate frame that served as the experimental setup for this project is seen in figure 2.1. The same motorized Brio labyrinth board as in [5] is used. In detail, two Dynamixel MX-12W motors actuate the two tilt axes of the board. A Dynamixel U2D2 converter enables to control the motors in series through a single USB interface. A wide-angle See3CAM_24CUG camera is mounted 20 cm above the center of the labyrinth and feeds the state estimation pipeline 1920x1200 color images at 55 Hz. The state estimation algorithm, based on a Kalman filter [9], tracks the ball's position (x, y) and velocity $(\dot{x}, \dot{y})$ alongside the board tilt angles (α, β). The algorithm had already been developed by a student in a different project. ROS 2 [10] facilitates communication between nodes. Finally, a desktop workstation is used to do all the online and offline computing. The workstation is equipped with an AMD Threadripper Pro 5955WX CPU and two RTX4090 GPUs. In particular, the training of the neural network and the offline dataset generation are conducted on the GPUs.

2.1.2 Model Predictive Controller

A fellow student at the IDSC had designed a linear and a nonlinear MPC controller for the labyrinth game. The system dynamics of both MPC controllers leveraged the linearized model of a ball-on-plate system as presented in [11]. Additionally, the linear MPC controller tried to follow the a-priori known reference path without taking any obstacles into consideration by planning a 20-step path to the next waypoint as shown in figure 2.2a. On the other hand, the nonlinear MPC controller planned a 100-step path three waypoints into the future, as seen in figure 2.2b, where the path avoided the five nearest holes, four nearest horizontal walls and six nearest vertical walls. These constraints introduced nonlinearity into the controller. However, the computational time of the nonlinear MPC exceeded the state estimate frequency, leading to the incorporation of a low-level controller to ensure real-time execution. The low-level controller planned an 18-step path from the current state estimate to the point of the high-level path that was 18-steps in the future of the current position.

2.2 MPC Imitation Model

An artificial neural network was employed to imitate the MPC controller. The network, schematically shown in figure 2.3, comprised a normalized state estimate as input, two velocity motor commands as output, and two fully-connected hidden layers each with 1024 units and utilizing ReLU activation

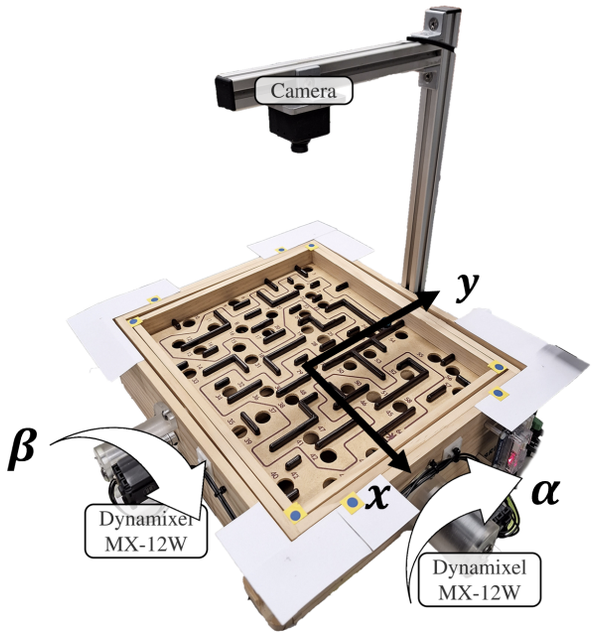
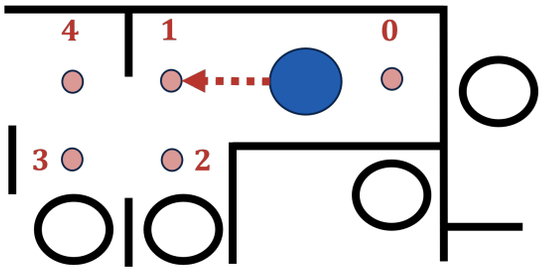
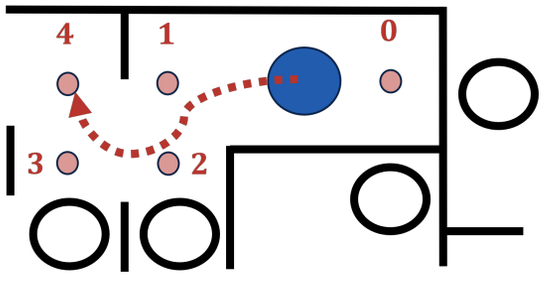


Figure 2.1: Motorized BRIO labyrinth game. Two servos actuate the tilt axes while a camera provides images for state estimation at 55Hz. The origin of the coordinate frame is in the center of the board.



(a) The linear MPC plans a 20-step path to the next waypoint without looking at holes or walls.



(b) The nonlinear MPC plans a 100-step path three waypoints into the future while avoiding the nearest holes and walls.

Figure 2.2: Linear vs. nonlinear MPC controllers.

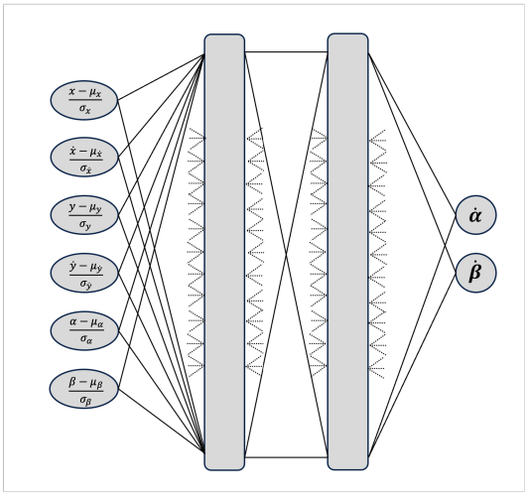


Figure 2.3: Schematic of the neural network that is used to imitate the nonlinear MPC controller. It consists of two hidden layers each with 1024 units and ReLU activation functions, and takes a normalized state estimate as input to output the two motor commands.

functions. The network underwent supervised training, minimizing the mean-squared-error. Prior to each training epoch, the dataset was shuffled and split up into batches of 512 data points. The model was trained for a total of 300 epochs.

Goals

The goals of the project were to show that when the neural network instead of the nonlinear MPC controls the labyrinth, the controller has a

- Higher bandwidth
- Similar labyrinth completion rate

Thus, the primary objectives were to showcase that imitation learning may be used to increase the bandwidth of a controller without losing accuracy. Two different imitation learning pipelines - online and an offline - were proposed. The fundamental difference between the two pipelines was that the online pipeline generated data with the neural network on the real system while the offline pipeline generated data beforehand based on a successful MPC run.

2.3 MPC Imitation Learning Pipelines

2.3.1 Online MPC Imitation Learning

The online imitation learning pipeline adopts an incremental training approach of the neural network π which is similar to the dataset aggregation algorithm in [12]. Figure 2.4 shows the communication flow for the online imitation learning. The execution node receives a state estimate S from the camera from which it calculates a velocity control command for the servo motors with the neural network π. This state estimate S is then passed on to the MPC solver which calculates the optimal control input U for the given state estimate. First, the nonlinear MPC solver is used to look for a solution. If the optimization problem fails (because the ball is too close to an obstacle or the initial state can't find a feasible path), then the linear MPC solver is used to calculate the control input label U. Once enough data has been collected, the training node retrains the neural network with the state and control input label pairs D. The training is conducted in a supervised fashion, where the mean-squared-error loss is minimized. The model π in the execution node is updated after each training iteration. As a result, the model incrementally learns how to navigate through the

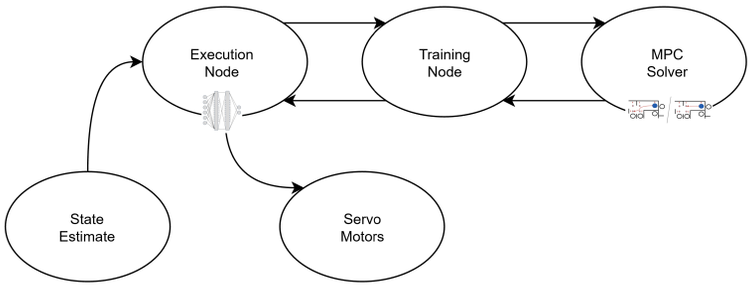


Figure 2.4: Online MPC imitation learning procedure. The execution node controls the labyrinth with the neural network. The MPC solver calculates the optimal control input of the state estimate. The training node trains and updates the neural network of the execution node using state estimate - control input label pairs.

labyrinth. Gradually, the ball makes it closer to the goal of the labyrinth. Algorithm 2.1 summarizes the online MPC imitation learning pipeline.

Algorithm 2.1 Online MPC Imitation Learning Pipeline

```
Initialize model π
Initialize D ← ∅
while ball not in goal do
    Collect state S with π
    Calculate control input U from MPC solver
    D ← D ∪ {S, U}
    if 1000 new samples then
        Train π with D
        Update π in execution node
    end if
end while
return π
```

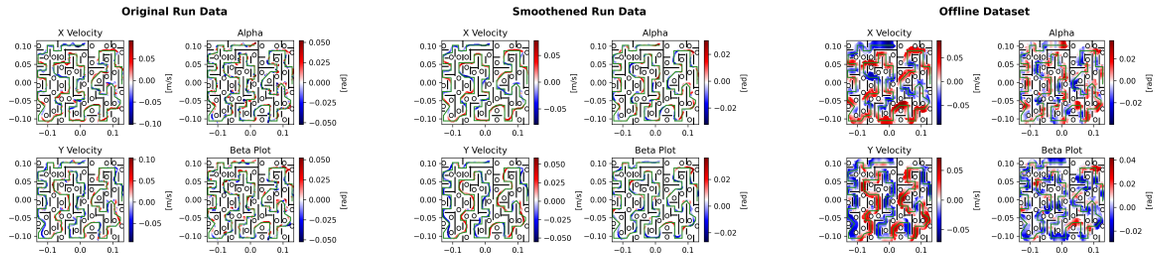
2.3.2 Offline MPC Imitation Learning

The offline imitation learning pipeline trains the neural network π with an augmented offline dataset from a successful nonlinear MPC run R. The pipeline starts with an offline data from a successful nonlinear MPC run R. This offline data is augmented into a rich dataset S by a method which is described later. The nonlinear MPC solver is used to calculate optimal control input labels U for all of the states in the augmented dataset S. If the nonlinear MPC solver fails to find a control input, then the linear MPC solver is used. This happens when the initial state from the dataset doesn't allow a feasible path. Once every state has an optimal control input label, the neural network is trained in a supervised fashion with the state S and control input U pairs. Finally, the labyrinth is controlled with the trained neural network π in the execution node. Algorithm 2.3 summarizes the offline MPC imitation learning pipeline.

Offline Dataset Method

The aim of the offline dataset method is to generate a dataset which contains many different states in an efficient manner. The method follows these steps:

1. Collect successful MPC run
2. Smoothen data with moving average filter (MAF)
3. Interpolate to have data points in 1 mm intervals



(a) Data from successful run of MPC-expert.

(b) Smoothened data from successful run of MPC-expert.

(c) Offline dataset with points populated as defined by the method.

Figure 2.5: Offline dataset generation from a successful nonlinear MPC-expert run.

4. Define maximum and minimum trajectory bounds
5. Interpolate between maximum and minimum bounds
6. Augment each data point with noise

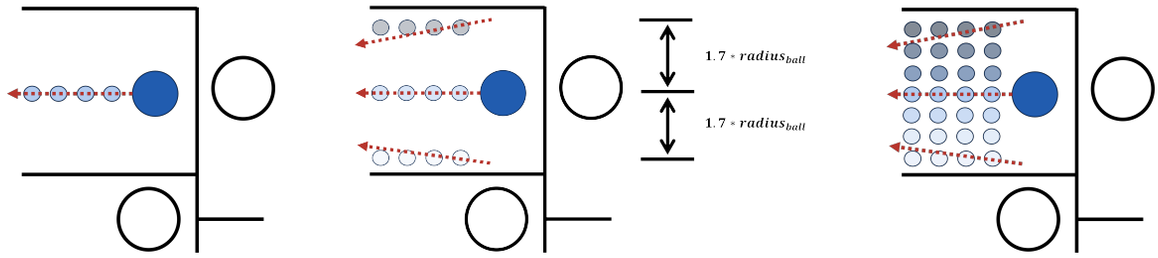
The original velocity and tilt angle data of a successful MPC run is plotted in figure 2.5a. As can be seen, there are spikes and oscillations in certain areas. This is mainly due to surface irregularities which make the ball bump into walls and take trajectories different to the planned ones. A MAF was applied to the raw data to get rid of these spikes and oscillations.

$$\text{Moving Average Filter (MAF):} \quad x^k_{avg} = \frac{1}{N} \sum_{i=k-\frac{N}{2}}^{k+\frac{N}{2}-1} x^i \tag{2.1}$$

Each data point x^k in the original MPC dataset was replaced by the average x^k_{avg} of the N nearest neighbors. To deal with boundary issues, the first and last $\frac{N}{2}$ data points were removed from the original dataset. The tuning parameter in (2.1) was the window size N. It was empirically set to fifty. The smoothened data is plotted in figure 2.5b. New data points were added around the smoothened trajectory in a tube-like fashion. The final offline dataset is seen in figure 2.5c.

The process for populating data points around the smoothed trajectory is illustrated in figure 2.6. Initially, the smoothened dataset underwent adjustments to ensure data points were spaced at 1 mm intervals. Interpolated values at specific points were computed as averages of their two neighboring points. Subsequently, maximum and minimum bounds were established around the nominal trajectory, depicted in figure 2.6b. The distance from the nominal trajectory to these bounds was set at 1.7 times the ball radius r, creating a reasonable tube of states for subsequent neural network training, as observed in figure 2.5c. To encourage the ball's movement towards the nominal trajectory, the velocities at the maximum and minimum bounds were modified. This adjustment, shown in figure 2.7, involved adapting the velocity value at the bound point using the weighted normal vector between two data points along the nominal trajectory. Similar adjustments were made for tilt angles, with the difference being the use of a distinct Δ for weighting the normal vector in tilt angle adjustments. Lastly, interpolation was employed to fill in missing values between the maximum and minimum bounds, yielding the result depicted in figure 2.6c.

To make the dataset richer, each data point x was augmented with fifty noisy values. In each instance, noise was randomly drawn from a uniform distribution $\mathcal{U}(a, b)$ and added to the current value. For the velocity noise, Δ represented the maximum velocity identified in the original MPC run data, while for the angle noise, Δ was the maximum tilt angle from the same dataset. The uniform distribution's range (a, b) was dynamically adapted based on the current value to maintain a consistent span across all data points. The noise algorithm is summarized in algorithm 2.2.

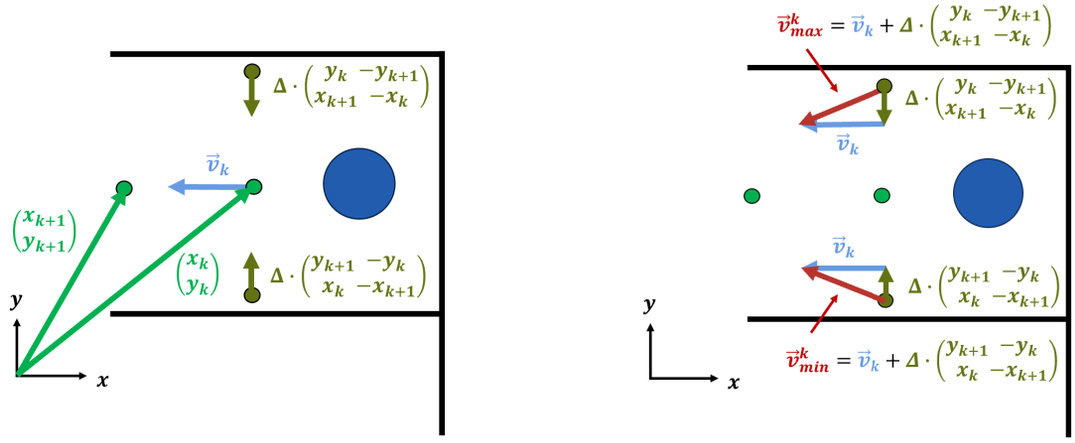


(a) Start with smoothened MPC run which has one data point every millimeter.

(b) Set maximum and minimum bounds and adjust the velocities and tilt angles to push the ball back to the center.

(c) Interpolate between the maximum and minimum bounds.

Figure 2.6: Illustration of the method used to generate data around the smoothened, nominal trajectory.



(a) Define the normal vector (olive green vector) with the vector between the $k+1$' and k'th point (green vectors). The velocity vector at the k'th point is shown in blue.

(b) Adjust the velocity vector (blue vector) in proportion to the normal vector between $k+1$' and k'th point (olive green vector). This gives the adapted velocity (red vector).

Figure 2.7: Illustration of the calculation to change the velocity at the maximum and minimum bound to push the ball back to the nominal, smoothened trajectory. The tilt angle is adapted the same way.

Algorithm 2.2 Noise With Adaptive Range

Initialize $x \leftarrow$ Data value
Initialize $\Delta \leftarrow$ Maximum noise value
$(a, b) = (-0.5 \cdot \Delta, +0.5 \cdot \Delta)$
if $x > 0.5 \cdot \Delta$ **then**
    $\delta = x - 0.5$
    $(a, b) = (-(0.5 + \delta) \cdot \Delta, (0.5 - \delta) \cdot \Delta)$
else if $x < -0.5 \cdot \Delta$ **then**
    $\delta = |x| - 0.5$
    $(a, b) = (-(0.5 - \delta) \cdot \Delta, (0.5 + \delta) \cdot \Delta)$
end if
return $x + \mathcal{U}(a, b)$ with 50 samples

The final offline dataset contained 960'700 data points. Figure 2.5c visually represents the offline dataset derived from the original data without noise.

The offline imitation learning pipeline with a neural network π is summarized in algorithm 2.3

Algorithm 2.3 Offline MPC Imitation Learning Pipeline

Initialize model π
Initialize $R \leftarrow$ MPC successful run
$S \leftarrow$ METHOD(R)
$U \leftarrow$ MPC(S)
$\pi \leftarrow$ Train(S,U)
return π
function METHOD(X)
    $X \leftarrow$ MAF(X)
    $X \leftarrow$ Bounds(X)
    $X \leftarrow$ Noise(X)
    return X
end function

Chapter 3

Results

3.1 Mixture-of-Experts

The evaluation of the two imitation learning pipelines took place on the same motorized Brio labyrinth board as in [5]. The outcomes were compared and benchmarked against test runs conducted with the nonlinear MPC controller. A mixture-of-experts network architecture for the final control policy is proposed in [13]. Test runs revealed that employing distinct models for different sections of the labyrinth yielded the best results.

In the visualization depicted in figure 3.1, the blue area utilized a model trained online, a process that consumed approximately half a day. Meanwhile, the green area employed a model trained with the complete offline dataset, consisting of 960,700 points, requiring nearly two days for control input label calculation with the MPC solver and the hardware described in section 2.1.1. Model training for the offline dataset took 34 minutes. The yellow area, prone to frequent ball falls, saw an improvement in controller reliability by training a neural network exclusively with data from the offline dataset in this region. Lastly, the red area involved training the neural network with an augmented offline dataset, combining online-collected data with the existing offline dataset before model training.

3.2 Performance

Using the hardware detailed in section 2.1.1, the performance of the mixture-of-experts model was compared with the nonlinear MPC across 15 runs. The results of the 15 test runs are shown in table 3.1. The first row illustrates a substantial reduction in control input calculation time when utilizing the neural network. The second row shows that the NN on average covers a greater distance in the labyrinth than the nonlinear MPC. The MPC, however, managed one successful run more than the NN. Additionally, the analysis in the fourth row indicates that the MPC exhibited a slightly faster progression from one waypoint to the next, with the successful runs being marginally shorter. The figures in figure 3.2 highlight the last waypoint reached for each run by the MPC and NN controllers. Figure 3.2a suggests that MPC failures are dispersed, primarily attributed to surface irregularities not modeled, causing deviations from the planned path and potential falls into holes. On the other hand, figure 3.2b indicates specific areas where the NN controller faced greater challenges.

The results showcase notable reductions in control input time and similar performance metrics in labyrinth navigation between the NN trained via imitation learning and the nonlinear MPC.

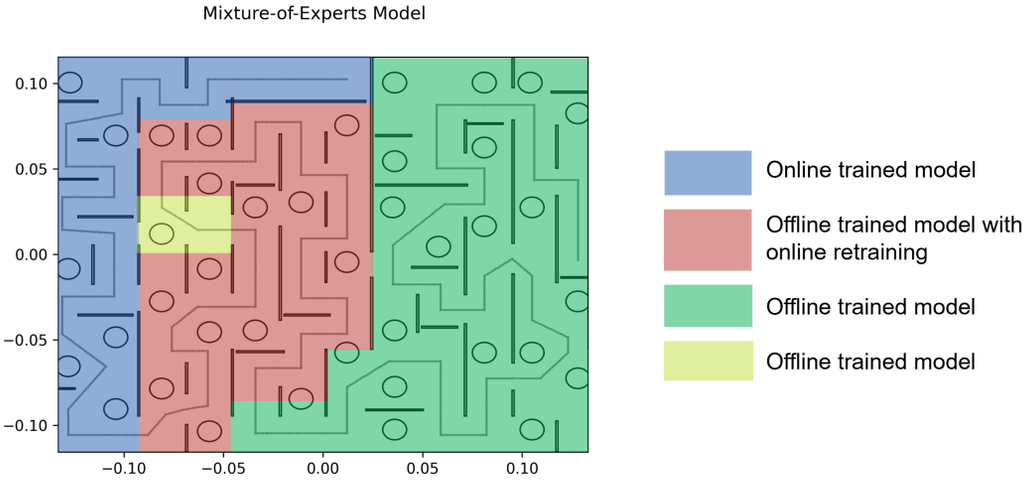
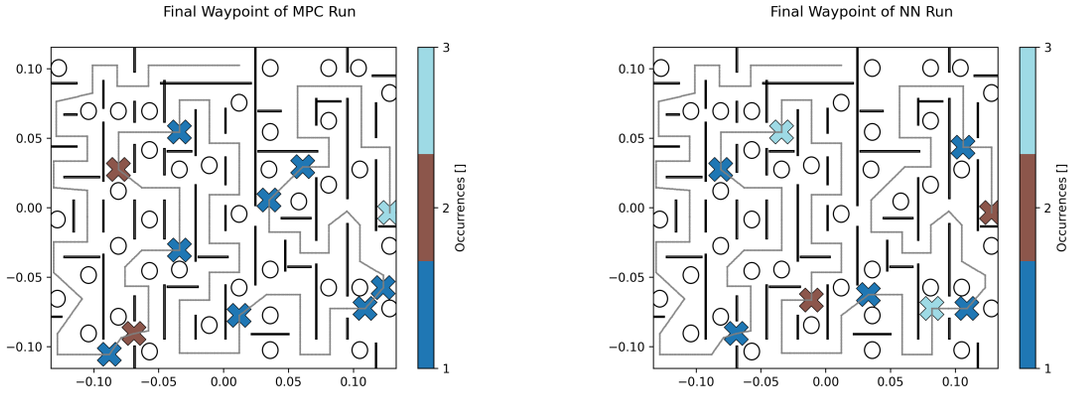


Figure 3.1: Mixture-of-experts. Different models are used for different areas of the labyrinth.

Metric	**Neural Network**	**Nonlinear MPC**
Control Input Calculation Time	$\mu = 1ms, \sigma = 0.1ms$	$\mu = 46ms, \sigma = 32ms$
Average Distance	64.4%	61.5%
Successful Run Percentage	13.3%	20.0%
Average Time per Waypoint	$0.76s$	$0.71s$
Average Time of Successful Run	$59.0s$	$57.8s$

Table 3.1: NN vs. MPC performance evaluation of 15 runs.



(a) Illustration of the final waypoints reached per run by the nonlinear MPC.

(b) Illustration of the final waypoints reached per run by the NN.

Figure 3.2: Comparison of the final waypoints reached per run by the nonlinear MPC and the NN.

Chapter 4

Conclusion

Model-predictive control is a methodology that computes optimal control inputs for a system while considering system and environmental constraints. In the context of the labyrinth game, the controller employs a ball-on-plate model to capture system dynamics and plans a path that navigates around nearby holes and walls. However, the computational expense of calculating control inputs makes real-time execution challenging. To address this, imitation learning, specifically an offline and online pipeline, was proposed to train a neural network to replicate the MPC controller. The final implementation utilized a mixture-of-experts to control the labyrinth, employing different models trained by the two pipelines in various regions. The outcomes of the project demonstrated that the neural network successfully achieved the defined goals by:

- Increasing the controller bandwidth
- Maintaining a similar labyrinth completion rate

Further work could focus on improving the labyrinth completion rate. One proposition is exploring the utilization of more local neural network experts within the mixture-of-experts framework, training each expert with region-specific data. Additionally, instead of just training the models with offline data, the offline data could be augmented with online data. Training the local expert with augmented data could enhance the robustness of the controller. Another idea would be to create a correlated offline dataset. Currently, the velocity noise and the tilt angle noise are uncorrelated and random. It would be interesting to train with an offline dataset where the velocity noise is correlated to the tilt angle value. This may improve the nonlinear MPC's optimization solutions. Augmenting the neural network inputs with information about the nearest walls and holes in the labyrinth could provide additional insights as the nonlinear MPC respects these constraints.

Experimenting with the neural network architecture is another avenue, such as replacing tilt angles with accelerations. This adjustment would require adapting the offline dataset to include information from the previous velocity for each state. Introducing more discrete waypoints in the labyrinth path might enhance robustness, considering that the MPC's offline-calculated control inputs permit shorter planned paths, resulting in a less aggressive controller.

Improving the MPC model by incorporating friction between the ball and labyrinth surface could address the main reason for the ball falling into a hole — surface irregularities. Finally, allowing the ball to bump into walls in the MPC problem formulation could increase the robustness of the neural network controller, acknowledging that, in practice, walls are not untouchable obstacles.

Bibliography

[1] J. H. Metzen, E. Kirchner, L. Abdenebaoui, and F. Kirchner, "The BRIO R Labyrinth Game - A Testbed for Reinforcement Learning and for Studies on Sensorimotor Learning."

[2] K. Öfjäll and M. Felsberg, "Combining Vision, Machine Learning and Automatic Control to Play the Labyrinth Game," Feb. 2016, arXiv:1604.00975 [cs]. [Online]. Available: http://arxiv.org/abs/1604.00975

[3] E. Frid and F. Nilsson, "Path Following Using Gain Scheduled LQR Control."

[4] L. Abdenebaoui, E. A. Kirchner, Y. Kassahun, and F. Kirchner, "A Connectionist Architecture for Learning to Play a Simulated Brio Labyrinth Game," in *KI 2007: Advances in Artificial Intelligence*, J. Hertzberg, M. Beetz, and R. Englert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4667, pp. 427–430, iSSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-540-74565-5_32

[5] T. Bi and R. D'Andrea, "Sample-Efficient Learning to Solve a Real-World Labyrinth Game Using Data-Augmented Model-Based Reinforcement Learning."

[6] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An Algorithmic Perspective on Imitation Learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018, arXiv:1811.06711 [cs]. [Online]. Available: http://arxiv.org/abs/1811.06711

[7] J. Harmer, L. Gisslén, J. del Val, H. Holst, J. Bergdahl, T. Olsson, K. Sjöö, and M. Nordin, "Imitation Learning with Concurrent Actions in 3D Games," Sep. 2018, arXiv:1803.05402 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1803.05402

[8] B. Zheng, S. Verma, J. Zhou, I. Tsang, and F. Chen, "Imitation Learning: Progress, Taxonomies and Challenges," Oct. 2022, arXiv:2106.12177 [cs]. [Online]. Available: http://arxiv.org/abs/2106.12177

[9] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[10] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abm6074

[11] M. Nokhbeh and D. Khashabi, "Modelling and Control of Ball-Plate System."

[12] S. Ross, G. J. Gordon, and J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," Mar. 2011, arXiv:1011.0686 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1011.0686

[13] J. Carius, F. Farshidian, and M. Hutter, "MPC-Net: A First Principles Guided Policy Search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, Apr. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9001182/

ETH zürich

Institute for Dynamic Systems and Control

Prof. Dr. R. D'Andrea, Prof. Dr. E. Frazzoli, Prof. Dr. Lino Guzzella, Prof. Dr. C. Onder, Prof. Dr. M. Zeilinger

Title of work:

MPC-Imitation Learning for a Labyrinth Game

Thesis type and date:

Semester Project, December 2023

Supervision:

Prof. Dr. Raffaello D'Andrea
Thomas Bi

Student:

Name: Philippe Brigger
E-mail: briggerp@student.ethz.ch
Legi-Nr.: 18-927-822
Semester: HS 2023

Statement regarding plagiarism:

By signing this statement, I affirm that I have read and signed the Declaration of Originality, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Declaration of Originality:

https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/declaration-originality.pdf

Zurich, 15.12.2023: