HAL
open science

MBAPPE: MCTS-Built-Around Prediction for Planning Explicitly

Raphael Chekroun, Thomas Gilles, Marin Toromanoff, Hornauer Sascha, Fabien Moutarde

To cite this version:

Raphael Chekroun, Thomas Gilles, Marin Toromanoff, Hornauer Sascha, Fabien Moutarde. MBAPPE: MCTS-Built-Around Prediction for Planning Explicitly. 2023. hal-04222148

HAL Id: hal-04222148

https://hal.science/hal-04222148v1

Preprint submitted on 29 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MBAPPE: MCTS-Built-Around Prediction for Planning Explicitly

Raphael Chekroun*[,1,2,3], Thomas Gilles*[,1]
Marin Toromanoff[2], Sascha Hornauer[1], Fabien Moutarde[1]

***Abstract*— We present MBAPPE, a novel approach to motion planning for autonomous driving combining tree search with a partially-learned model of the environment. Leveraging the inherent explainable exploration and optimization capabilities of the Monte-Carlo Search Tree (MCTS), our method addresses complex decision-making in a dynamic environment. We propose a framework that combines MCTS with supervised learning, enabling the autonomous vehicle to effectively navigate through diverse scenarios. Experimental results demonstrate the effectiveness and adaptability of our approach, showcasing improved real-time decision-making and collision avoidance. This paper contributes to the field by providing a robust solution for motion planning in autonomous driving systems, enhancing their explainability and reliability. Code is available under `https://github.com/raphychek/mbappe-nuplan`.**

I. INTRODUCTION

Innovations in machine learning techniques have led to significant advancements in self-driving technology. Particularly, the use of deep learning has greatly improved the perception stage of autonomous driving. These developments have been complemented by progress in sensor technology and mapping methods. As a result, the focus is now shifting to the next challenges of autonomous driving, and motion planning emerges as a pivotal component. After identifying roads and monitoring nearby vehicles and object entities, the autonomous driving system must now decide its future path and plan its trajectory accordingly to ensure a collision-free route while respecting traffic rules.

Therefore, this study centers on the mid-to-end stage of autonomous driving, presuming that perception tasks have already been accomplished and working toward an efficient and explainable motion planning. In this realm, recent research mostly focus on Imitation Learning (IL) [1]–[3] or hybrid IL and rule-based methods [4], [5].

However, rule-based methods for autonomous driving are limited by their lack of scalability, adaptability, robustness in complex and ambiguous situations, and their inability to handle unconventional scenarios. This contrasts with machine-learning based approaches that address these limitations through data-driven learning and adaptability.

Nonetheless, while Neural Networks (NN) provide a powerful and flexible tool for learning to drive using supervised labels with IL methods [1], [6], [7], they remain limited in the long-term understanding of the consequences of their actions. Therefore, they may not comprehend the full scope of interactions with the map and other agents. Deep Reinforcement Learning (Deep RL) based methods [8]–[10] aim to incorporate long-term returns of such consequences in the training of these networks. However, this causal understanding remains implicit and not guaranteed, and Deep RL training is most often sample inefficient.

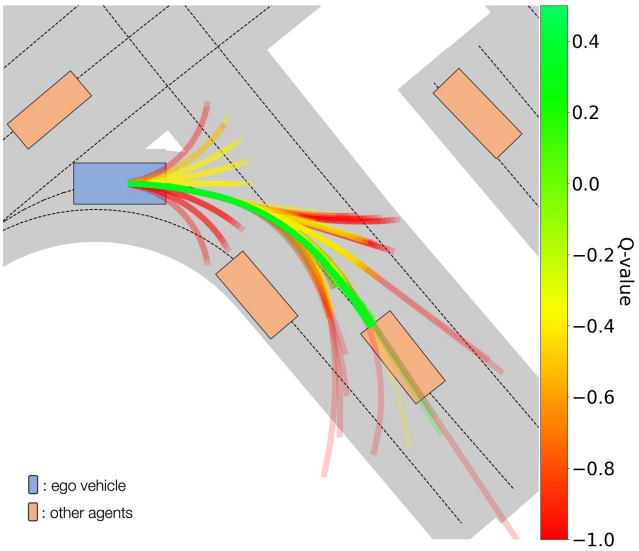

Fig. 1. Visualization of the exploration done by MBAPPE in one planning step. We display the bird-eye-view trajectory pieces in xy coordinates. As the road is turning right, the MCTS explores multiple steering angle and acceleration configurations to correctly take the turn. MBAPPE finally selects the path which maximizes the Q-value (in green).

Our approach aims to get the best of both worlds by using an IL prior to guide a MCTS [11], [12] into explicitly exploring the consequences of actions, validating the NN trajectory if it respects driving constraints, or exploring new actions if required, see Figure 1. The main challenge in running a MCTS is that it assumes environment transitions to be deterministic and perfectly known. While this is true for the displacement of the ego vehicle given its actions, and for the update of the map that remains the same, other agents will also move on their own accord. In order to have a realistic world model, we developed an IL model to predict all the other agents future trajectories. This way we get an approximate of the future transitions that enables us to roll out the consequences of our chosen actions on multiple time-steps.

In this paper, we extend the MCTS paradigm to partially-learned environment and apply it to autonomous driving.

[1] Mines Paris - PSL University, Centre for Robotics, Paris, France
[2] Valeo Driving Assistance Research, Créteil, France
[3] Department of Civil and Environmental Engineering, University of California, Berkeley, USA
*Equal contributions

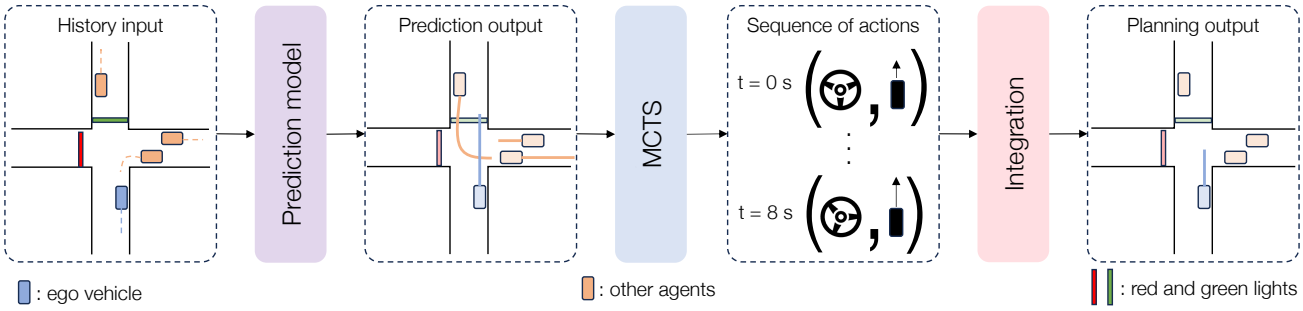

Fig. 2. **MBAPPE pipeline** A prediction model infers future trajectories of other agents in the scene. This information is fed to the MCTS which outputs a sequence of consecutive actions. Those are integrated to form an improved trajectory planning for the ego.

Next, we validate our performance on nuPlan [13] simulation environment and compare to other existing baselines. Lastly, we highlight the explainability of our approach which allows easy observation and analysis of the steps leading to any given decision via its decision tree.

II. Related work

MBAPPE seeks to leverage imitation learning (IL) to guide a MCTS model in exploring the outcomes of its actions. As such, this section is dedicated to examining rule-based and learning-based motion planning techniques, and strategies integrating MCTS with deep learning.

a) Rule-based methods: Rule-based methods employ explicit rules to dictate the behavior of autonomous vehicle, making them interpretable by nature [14]–[16]. A notable instance is the Intelligent Driver Model (IDM) [17], designed to track leading vehicles while maintaining safe distances through computation of optimal acceleration based on the leading vehicle's speed. Rule-based methods were extended in predictive rule-based approaches which anticipate future environmental states to improve collision avoidance [18]–[20]. However, rule-based methods are inflexible and rely on perfect and consistent representation of the environment. This characteristic make them struggle with generalization to novel scenarios or with the inherent variability of real-world conditions.

b) Imitation learning methods: Imitation learning methods allow to learn how to drive from supervised data, leading to more generalizability than rule-based methods. Some of these methods directly create driving plans or commands [7], [21], but they suffer from a lack of interpretability and general robustness. To address these issues, some other approaches focus on making the planning decisions more interpretable. For instance, Dauner et al. developed Predictive Driver Model (PDM) [4] to combine an interpretable IDM with a simple neural network. Some methods deal with the robustness problem by generating multiple planning options with deep learning and then choosing the best one with the lowest cost [22]–[25] or by refining deep-based predictions [2], [26]. However, IL methods still suffers from distribution mismatch where agent fails to recover from accumulation error thus leading to increasingly out of expert distribution states, and lacks of long-time reasoning.

c) Reinforcement learning methods: Instead of copying human behavior like IL, RL models use a reward system to judge how good a strategy is. This can lead to improved decision-making, sometimes even outperforming humans [27]. Model-free reinforcement learning focuses on learning optimal actions directly from observed states and rewards without creating an explicit model of the driving environment. Even though RL is successful for simple autonomous driving tasks [8], up to now, no published work has reported sucess of exclusively RL-based method in autonomous driving for complex urban environments [28]. Furthermore, RL suffers from sample inefficiency and lack of convergence guarantees and interpretability. Recent works leveraged supervised learning in RL pipelines to overcome these limitations [10], [29], thus compensating the weakness of the RL gradient during training.

d) Methods integrating MCTS with deep learning: Integrating MCTS with deep learning techniques has emerged as a compelling approach to enhance decision-making processes in various domains. Silver et al. [27] pioneered this fusion by combining MCTS with deep supervised learning to achieve groundbreaking results in the game of Go with AlphaGo. This paradigm was extended with AlphaZero [30] by relying solely on self-play and RL. MuZero [31] finally embraced implicitness and extended the generality of these approaches by employing learned models to simulate outcomes and inform strategic decision-making.

In the realm of autonomous driving, Chen et al. [32] integrated MCTS with deep learning but relied on implicitness for the tree transitions and prior computation, possibly leading to inexplicable behaviors which are not desirable for this domain of application. Other published methods lack generalizability and constraint their applicative fields to simplified custom environments such as highway driving without possibility for public benchmarks comparison [33], [34], or high level tactical decisions [35].

III. Method

In this section, we introduce MBAPPE and its components. In particular, we present the known and learned features of the world model, and technical details of our MCTS design and exploration steps.

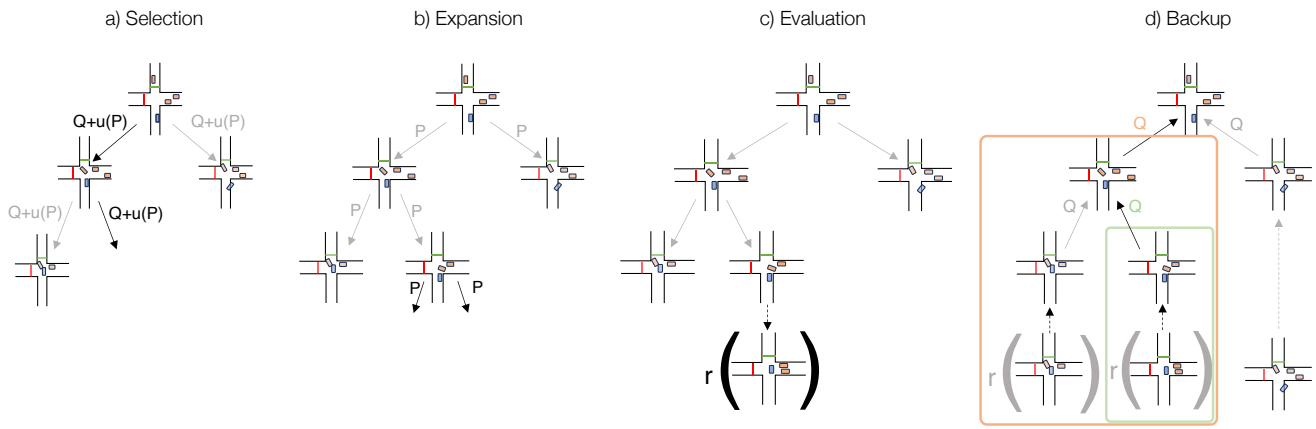

Fig. 3. **MCTS steps** a) Each simulation pass in the tree follows a trade-off between exploitation of the best Q value of an action, and the exploration term $u(P)$ that encourages to explore nodes with less visits N along the prior P. b) The leaf node is possibly expanded following some probabilities depending on the prior P and the continuity constraints. c) After the simulation, the leaf node is evaluated by explicitly computing the reward r described in Section III-C. d) Q-values are updated so means of the rewards r in the sub-tree below each actions are tracked.

A. MBAPPE framework

At each time-step, a neural network (based on an open-loop version of Urban Driver [21]) predicts an estimation of the ego trajectory and of the future trajectories of every other agents around the ego. This information is fed to the MCTS, which will deploy an internal lightweight simulation where the ego trajectory is used as a prior to guide the first steps of exploration, and other agents trajectories are leveraged to build the world model. At each simulation-step, which follow a planning time axis inside the tree, the MCTS explores the possible actions and internally simulates the evolution of the environment to check how those explored actions will impact its driving performances (driving out of area, check for collisions with static objects, check collisions with other agents thanks to their estimated trajectory, etc).

The global pipeline is represented in Figure 2.

B. World Model

The Monte-Carlo tree search leverages an internal simplified representation of the world where it can quickly iterate to explore possible sequences of actions and their consequences. This environment is made of two categories of features:

- Known features:
  - The map information, including traffic light,
  - Static objects such as traffic cones and barriers
  - Dynamic objects such as neighboring vehicles, traffic cones or pedestrians, which we will consider as other agents evolving in the simulated environment
- Learned features:
  - Estimated future trajectories of other agents given by the NN prediction.

C. MCTS design and tree steps

Our MCTS is based on a kinematic bicycle model of the vehicle. Actions are defined as a tuple (a, δ), where a is the acceleration and δ the steering angle. Accelerations and steering are discretized in 13 values each, in the respective range of $[-3, 3]$ m.s^{-2} and $[-\pi/4, \pi/4]$ rad. Actions are integrated every 0.1 s.

The simulation process of our tree search is detailed in Fig. 1. The tree is initialized with a single root node representing the current context. Each tree node stores 3 values: Q the expected return, P the action prior and N the number of visits. The nodes are built and evaluated iteratively through the following steps:

- **Selection**: We follow the PUCTS [27] formula to select the next action following a trade-off between the exploitation of Q and the exploration of unvisited nodes with low N.
  At a node state $\mathcal{S}$ the action $\mathcal{A}$ is chosen using the following formula:
  $$\mathcal{A}_t = \underset{\mathcal{A}}{\operatorname{argmax}}\, Q(\mathcal{S}, \mathcal{A}) + c_{puct} P(\mathcal{S}, \mathcal{A}) \frac{\sqrt{\sum_{\mathcal{B}} N(\mathcal{S}, \mathcal{B})}}{1 + N(\mathcal{S}, \mathcal{A})} \tag{1}$$
  with c_{puct} an hyper-parameter balancing the trade-off between exploration and exploitation. We found $c_{puct} = 2$ to perform the best in our experiments.
- **Expansion**: We expand leaf nodes by all physically possible actions from the state of the leaf node, following a prior P and some continuity constraints. These constraints ensure both comfort and physical feasibility of successive actions. Prior design and continuity constraints are described in Section III-D.
- **Evaluation**: We consider that driving rewards are rather short term (crash or not, exit road or not within the next 6 or 8 seconds). Therefore they do not need to be bootstrapped by a learned value network, but rather can be evaluated at the current simulation step by checking for them directly. Our computed reward r_t at state s_t is made of these main components:
  - Progress: distance advanced since the last node,

normalized by maximum allowed speed limit ($[0, 1]$),
  - Collision: penalty for collision with car and pedestrian (-5) or object (-2),
  - Route: -0.5 if the vehicle is not on the expected road,
  - Drivable area: -1 if the vehicle is not on the drivable area,
  - Center of the road:
    * $-sin(\theta)/2$ where θ is the angle difference between the ego heading and the closest centerline heading,
    * $-d/2$ where d is the distance between the ego position and the closest centerline.
- **Back up**: We update the Q values using the cumulative reward as in MuZero [31]:

$$
\begin{aligned}
G^k &= \sum_{\tau=0}^{l-1-k} \gamma^\tau r_{k+1+\tau} \\
Q\left(s^{k-1}, a^k\right) &:= \frac{N\left(s^{k-1}, a^k\right) \times Q\left(s^{k-1}, a^k\right) + G^k}{N\left(s^{k-1}, a^k\right) + 1} \\
N\left(s^{k-1}, a^k\right) &:= N\left(s^{k-1}, a^k\right) + 1
\end{aligned}
\tag{2}
$$

We use a discount factor γ of 1.

D. Prior and continuity constraints

An efficient MCTS exploration process can be achieved by leveraging two approaches.

Firstly, providing the MCTS an intuition over actions to explore to prioritize the more probable ones. This issue is tackled using a prior over the distribution of actions for each node. This prior is usually learned and inferred for every node [31], which is computationally expensive, or handcrafted. Secondly, to further streamline the exploration process, we narrowed down the action space, thereby reducing the overall actions that need to be explored to the most critical ones. To achieve this, we integrated continuity constraints into the MCTS to ensure not only the physical feasibility of the actions explored but also to enhance comfort and to reduce the exploration time.

1) The prior: We designed a prior which relies on both handcrafted rules and learned rules, all without incurring any additional computational overhead.

The prior function is made of two parts:

- The handcrafted prior P_h prioritizes exploration around the constant speed with null steering angle,
- The learned prior P_l is obtained by deriving the prediction of the ego trajectory by the NN into consecutive actions. This prior advantages the possibility of following NN actions for the first T time steps of the internal simulation of the MCTS. We found $T = 1$ s to perform the best in our experiments.

Both P_l and P_h are Gaussians centered on the chosen action. The Gaussian are parametrized with a very high variance ($\sigma^2 = 100$) to encourage an almost uniform exploration.

The designed prior can be written:

$$
P^t = \begin{cases} P_h^t + P_l^t & \text{if } t \leq T \\ P_h^t & \text{if } t > T \end{cases} \tag{3}
$$

2) Continuity constraints: To ensure the output trajectory is physically feasible and to minimize the total number of actions to explore, we implemented continuity constraints in the MCTS.

These constraints are two folded:

- The Tree Constraint: At a given step t of the real-world vehicle movement, the root node of the novel tree will be constrained to explore neighboring accelerations and steering angles relatively to the actions taken at time $t-1$ by the previous tree. This constraint favors a behavior continuity between successive time-steps and corresponding MCTS.
- The Node Constraint: During the MCTS internal expansion phase, exploration only focuses on neighboring accelerations and steering angle values relatively to the actions of his parent node. This constraint favors a behavior continuity during the expansion phase of a given MCTS.

We formulate both continuity constraints as restricting the following action (a_{t+1}, δ_{t+1}) to be within a range of $a_t \pm 0.15$ m.s^{-2} for the acceleration and $\delta_t \pm \pi/240$ rad for the steering angle with (a_t, δ_t) the action at the previous time-step.

IV. Experimental results

Dataset: We show results on the nuPlan dataset. It encompasses 1300 hours worth of real vehicle motion data along with its corresponding simulator. Within the nuPlan framework, we chose to assess the performance of planners on closed-loop non-reactive agents benchmark. We focus on this benchmark, as evaluations conducted in closed-loop more effectively assess an agent's driving capabilities without the need to compare them to a flawed 'ideal' behavior as typically seen in open-loop assessments. Additionally, we chose non-reactive agents for our study, as preliminary experiments and other performance benchmarks [4], [13] have demonstrated that outcomes are largely consistent between reactive and non-reactive agents. All simulations are ran on 100 scenarios of each of the 14 scenarios types (totaling 1,118 scenarios in practice, as all 14 types do not have 100 available scenarios) of the nuPlan challenge, following the Val14 benchmark validation set [4].

Score and metrics: We use the nuPlan official score, which measures driving quality between 0 and 100 through a combination of 16 normalized driving metrics related to infraction rate, ego comfort, or progress toward the goal. We decided to put a special emphasis on the metrics of collision rate (CR), driving area non-compliance (DA) and ego progress (EP) in our experiments, as they are key elements for a safe and efficient autonomous driving system.

Implementation details: For ablations studies, the number of simulation steps is limited to 256 in each MCTS. In our setup (Intel Core i7-9700K CPU @ 3.60GHz) the whole pipeline inference time is ~ 0.15 seconds for this setup,

including input pre-processing, prediction model, MCTS and post-processing. The pipeline runs on CPU only. For inference speed purposes, we only expand new possible actions every 1 s. We observed no drop of performance.

A. Ablation study over the prior

An ablation study over the choice of prior is presented table I. Continuity constraints are the one described section III-D.

Prior		Metrics			
Learned	Crafted	CR ↓	DA ↓	EP ↑	Score ↑
-	-	6%	4%	31%	26%
✓	-	11%	3%	88%	65%
-	✓	6%	4%	95%	82%
✓	✓	**5%**	**2%**	**96%**	**86%**

TABLE I

ABLATION OVER THE PRIOR.

We can see from results of Table I that MCTS without prior is inefficient. Exploration being unguided, the expansion phase does not create node leading to a good reward *a priori*. Following P_l for the first steps of the simulation allowed to significantly improve the exploration phase by guiding the MCTS to stay within the driving area. Indeed, thanks to continuity constraints, a good beginning of the trajectory allows to stay on the road and reach acceptable metrics. Interestingly, leveraging only P_l leads to an increase of the collision rate: if the MCTS first actions differ from the prior's, there will be a mismatch between the guidance it provides and the actual scenes which can lead to collisions.

Leveraging P_h allows the MCTS to prioritize exploration of the most common behavior on average (staying at around the same velocity with a null steering angle), therefore minimizing collisions and optimizing overall progress. Notably, using only this naive prior without any kind of learning already yields very good performance, highlighting the power of guided exploration in the MBAPPE method. Finally, leveraging $P_h + P_l$ allows to prioritize this kind of behavior while starting with a better heads up and leads to best results on this set of experiments.

B. Ablation study over continuity constraints

An ablation study over the choice of continuity constraints is presented Table II. For these experiments, prior is $P_h + P_l$.

It becomes apparent that when applied separately, continuity constraints offer only marginal improvements to our method. A possible explanation is that the handcrafted identity prior already directs the MCTS towards a form of constrained exploration similar to what is achieved through node constraints. However, utilizing both node and tree constraints independently does enhance the exploration process. Importantly, the combined effects of these constraints not only substantially increase performance but also ensure a consistent selection of actions, both within a single tree and across multiple trees that correspond to sequential planning steps.

Constraints		Metrics			
Tree	Node	CR ↓	DA ↓	EP ↑	Score ↑
-	-	7%	4%	95%	79%
✓	-	8%	3%	96%	82%
-	✓	8%	2%	94%	82%
✓	✓	**5%**	**2%**	**96%**	**86%**

TABLE II

ABLATION OVER CONTINUITY CONSTRAINTS.

C. Comparison with state-of-the-art methods

We compare MBAPPE's performance with other state-of-the-art method on the validation scenario of the Val14 benchmark [4]. See Table III.

Baselines: Urban Driver [21] utilizes PointNet [36] layers to process polyline and employs a MLP following a multi-head attention block to forecast the ego trajectory. GameFormer Planner [2] exploits a Transformer to predict all agents trajectories before refining ego planning via non-linear optimization. PlanCNN [3] leverages a CNN on rasterized inputs to predicts the ego trajectory. PDM [4] leverages an improved IDM [17] model combined with a simple MLP to generate several trajectories which are then scored to return the optimal one. GC-PGP [5] categorizes proposed plans according to their traversal of a route-constrained lane graph, and then identifies the most probable cluster center.

Method	CR ↓	DA ↓	EP ↑	Score ↑
Urban Driver MA [21]	34%	26%	96%	47%
GameFormer Planner [2]	6%	4%	98%	84%
PDM-Hybrid [4]	**2%**	**0%**	**99%**	**93%**
IDM [17]	12%	6%	95%	76%
GC-PGP [5]	-	-	-	57%
PlanCNN [3]	-	-	-	73%
MBAPPE (GameFormer)	3%	2%	98%	90%
MBAPPE (Urban Driver)	5%	2%	96%	86%

TABLE III

VAL14 BENCHMARK ON NUPLAN

For this comparison, we extended Urban Driver to predict trajectories of all other agents in the scene in addition to the ego's. We name this updated version Urban Driver Multi-Agent (Urban Driver MA). Then, we evaluated two versions of MBAPPE. One leverages Urban Driver MA as prediction and prior model (c.f. Figure 2), and the other a GameFormer model. Other components of those systems are identical.

In our experiments, we found that enhancing a prediction model with MBAPPE consistently results in improved planning. Specifically, when integrated with GameFormer, MBAPPE yields a substantial improvement in key metrics

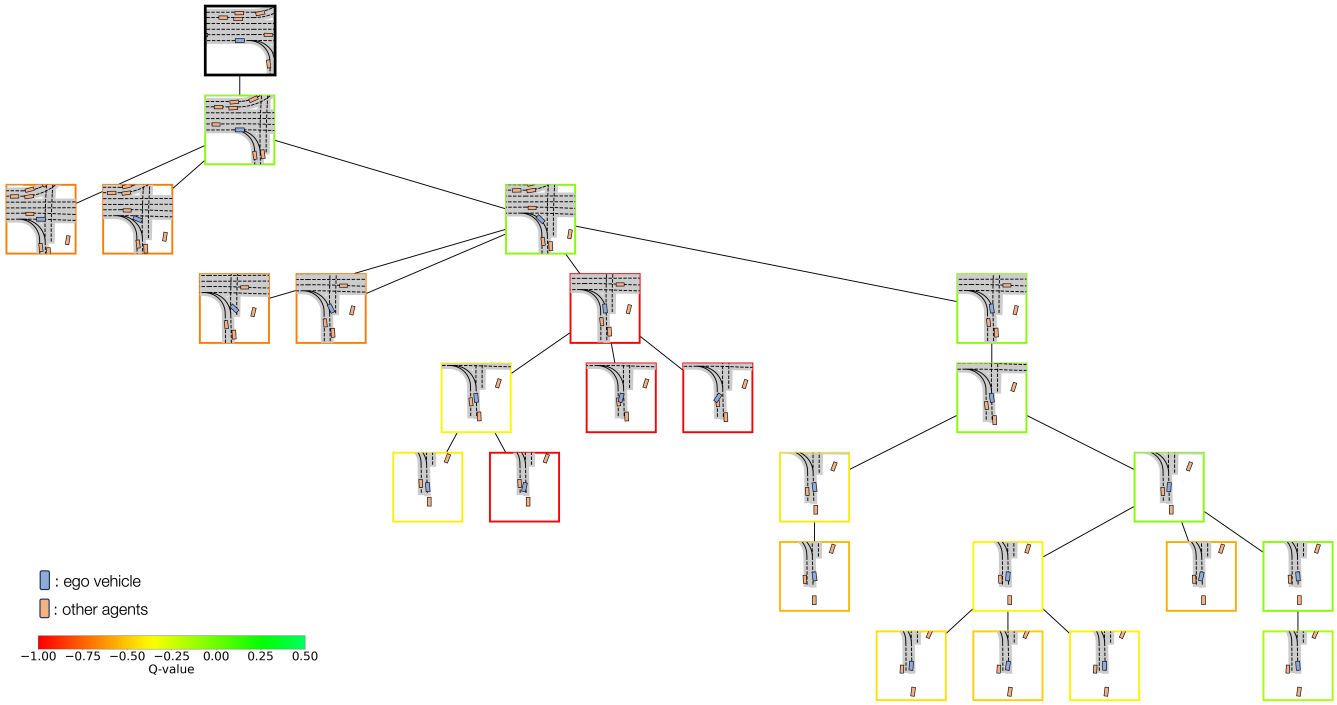

Fig. 4. A subset of a decision tree obtained with MCTS exploration. Nodes are colored according to their Q-value. The root node correspond to the present state of the vehicle in the nuPlan simulator. We observe that the orange left branch exploration leads to the ego leaving the expected route, hence the low Q-value. The red middle branch exploration leads to a collision, thus explaining the low Q-value. The green right branch exploration presents the expected behavior and therefore has the highest Q-value. The explored planning can also be observed in Figure 1.

compared to using non-linear optimization techniques as done with the GameFormer Planner.

Thus, MBAPPE not only delivers state-of-the-art performance, but is also an explainable and interpretable operator when applied to predictive models. This dual benefit both refines decision-making policies and provides added adaptability.

V. An explicit and explainable method

A key benefit of this technique is its simplicity: it requires only basic high-level directives in the form of a reward function (e.g., move ahead, avoid collisions, stick to the route, and remain on the road). Despite its vague prior, the method yields highly effective and realistic planning. This eliminates the need for specific, hard-to-generalize rules, like basing decisions on the road's curvature or the speed of the car ahead, as well as the use of hardly interpretable neural networks. As a result, our approach is highly flexible, adaptable, and explainable.

Indeed, decisions of the MCTS are explainable and the internal process that led to those decisions can be easily observed and analyzed. Figure 4 provides an example of a decision tree of the MCTS in which we can observe several exploration branches and their consequences on the tree expansion. In particular, we observe on the green right branch that internal exploration leading to desirable behavior yields the highest Q-value and further exploration of that branch. When exploration leads to collisions or to the ego leaving its expected route, the Q-value is low and exploration stops, as shown in the red middle and orange left branches. Figure 4 shows that MCTS decisions-making process is transparent and explainable, thus leading to an explicit and safe planning.

VI. Conclusion

This paper presents MBAPPE, a novel approach extending MCTS for planning within a partially learned environment in the context of autonomous driving. Through ablation studies, we highlighted the advantages of incorporating the designed priors and continuity constraints into the MCTS tree. Comparative analysis using a benchmark on the nuPlan simulator revealed that MBAPPE is an effective refinement operator for planning models, consistently outperforming vanilla models across all evaluation metrics. Finally, we emphasize the interpretability provided by this technique, a critical attribute for ensuring the safety and reliability of autonomous vehicles.

In terms of future work, as MBAPPE improves planning model capabilities, one could fine-tune the prior network similarly to the approach used in AlphaGo [27]. This would enable the network to better emulate the MCTS output, thereby refining its priors and initiating a cycle of self-improvement. Better results could also be achieved with a more complex learned prior inferred for each node [31], [32], as well as learning a bootstrapped value network to estimate node expected returns in addition to the current reward. However this would require more network inferences and could harm the execution time.

Acknowledgments Thank you to Valentin Charraut for his fruitful help on implementation and optimization. We are also thankful to Thibault Buhet and Dzmitry Tsishkou for their insights, revisions and relevant comments.

REFERENCES

[1] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Gohome: Graph-oriented heatmap output for future motion estimation," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 9107–9114.

[2] Z. Huang, H. Liu, and C. Lv, "Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving," *arXiv preprint arXiv:2303.05760*, 2023.

[3] K. Renz, K. Chitta, O.-B. Mercea, A. S. Koepke, Z. Akata, and A. Geiger, "Plant: Explainable planning transformers via object-level representations," in *Conference on Robot Learning*, PMLR, 2023, pp. 459–470.

[4] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, "Parting with misconceptions about learning-based vehicle motion planning," *arXiv preprint arXiv:2306.07962*, 2023.

[5] M. Hallgarten, M. Stoll, and A. Zell, "From prediction to planning with goal conditioned lane graph traversals," *arXiv preprint arXiv:2302.07753*, 2023.

[6] M. Bojarski, D. Del Testa, D. Dworakowski, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[7] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[8] A. Kendall, J. Hawke, D. Janz, *et al.*, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8248–8254.

[9] D. Chen, V. Koltun, and P. Krähenbühl, "Learning to drive from a world on rails," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 590–15 599.

[10] R. Chekroun, M. Toromanoff, S. Hornauer, and F. Moutarde, "Gri: General reinforced imitation and its application to vision-based autonomous driving," *Robotics*, vol. 12, no. 5, 2023, ISSN: 2218-6581. DOI: 10.3390/robotics12050127.

[11] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," vol. 4630, May 2006, ISBN: 978-3-540-75537-1. DOI: 10.1007/978-3-540-75538-8_7.

[12] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293, ISBN: 978-3-540-46056-5.

[13] K. T. e. a. H. Caesar J. Kabzan, "Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," in *CVPR ADP3 workshop*, 2021.

[14] S. Thrun, M. Montemerlo, H. Dahlkamp, *et al.*, "Stanley: The robot that won the darpa grand challenge," in *The 2005 DARPA Grand Challenge: The Great Robot Race*, M. Buehler, K. Iagnemma, and S. Singh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–43, ISBN: 978-3-540-73429-1. DOI: 10.1007/978-3-540-73429-1_1. [Online]. Available: https://doi.org/10.1007/978-3-540-73429-1_1.

[15] J. Leonard, J. How, S. Teller, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[16] A. Bacha, C. Bauman, R. Faruque, *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[17] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[18] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "Diffstack: A differentiable and modular control stack for autonomous vehicles," in *Conference on Robot Learning*, PMLR, 2023, pp. 2170–2180.

[19] M. H. Danesh, P. Cai, and D. Hsu, "Leader: Learning attention over driving behaviors for planning under uncertainty," in *Conference on Robot Learning*, PMLR, 2023, pp. 199–211.

[20] W. Zeng, W. Luo, S. Suo, *et al.*, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[21] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *Conference on Robot Learning*, PMLR, 2022, pp. 718–728.

[22] A. Cui, S. Casas, A. Sadat, R. Liao, and R. Urtasun, "Lookout: Diverse multi-future prediction and planning for self-driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 107–16 116.

[23] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, Springer, 2020, pp. 414–430.

[24] W. Zeng, W. Luo, S. Suo, *et al.*, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[25] W. Zeng, S. Wang, R. Liao, Y. Chen, B. Yang, and R. Urtasun, "Dsdnet: Deep structured self-driving network," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, Springer, 2020, pp. 156–172.

[26] G. Aydemir, A. K. Akan, and F. Güney, "Adapt: Efficient multi-agent trajectory prediction with adaptation," *arXiv preprint arXiv:2307.14187*, 2023.

[27] D. Silver, A. Huang, C. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016. DOI: 10.1038/nature16961.

[28] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *arXiv preprint arXiv:2306.16927*, 2023.

[29] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7153–7162.

[30] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017. arXiv: 1712 . 01815. [Online]. Available: http://arxiv.org/abs/1712.01815.

[31] J. Schrittwieser, I. Antonoglou, T. Hubert, *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[32] J. Chen, C. Zhang, J. Luo, J. Xie, and Y. Wan, "Driving maneuvers prediction based autonomous driving control by deep monte carlo tree search," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7146–7158, 2020. DOI: 10.1109/TVT.2020.2991584.

[33] K. Shu, H. Yu, X. Chen, *et al.*, "Autonomous driving at intersections: A critical-turning-point approach for left turns," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 1–6.

[34] T. Ha, K. Cho, G. Cha, K. Lee, and S. Oh, "Vehicle control with prediction model based monte-carlo tree search," in *2020 17th International Conference on Ubiquitous Robots (UR)*, 2020, pp. 303–308. DOI: 10.1109/UR49135.2020.9144958.

[35] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE transactions on intelligent vehicles*, vol. 5, no. 2, pp. 294–305, 2019.

[36] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.