Lightweight Machine Learning for Real-Time Gear Change Prediction in Autonomous Parking

Ahmed A. Kamel*, Reda Alkhoribi, M. Shoman, Mohammed A. A. Refaey
Faculty of Computers and AI, Cairo University, Egypt

***Abstract*—Real-time motion planning for autonomous parking on embedded advanced driver-assistance system (ADAS) platforms faces a fundamental computational bottleneck: transformer-based approaches (e.g., Motion Planning Transformer, Diffusion-based planners) achieve strong performance but incur prohibitive computational costs unsuitable for resource-constrained automotive systems. This work proposes a lightweight alternative machine learning approach using Random Forest classifiers and regressors to predict parking trajectory regions and vehicle orientations, enabling accelerated Rapidly-exploring Random Trees (RRT) planning without sacrificing robustness. The approach is trained on a dataset of 10,725 synthetic perpendicular backward parking scenarios generated via Rapidly-exploring Random Tree Star (RRT*) in the Reeds-Shepp configuration space. Using Random Forests with 20 trees and maximum depth 8, the method achieves 98.3–100% success rate in multi-direction-change scenarios with planning times of 0.15–0.25 seconds, compared to 2.81 seconds for unconstrained RRT. In scenarios with insufficient prediction guidance, the constrained planner can maintain a fallback mechanism that preserves RRT's probabilistic completeness guarantees. This work demonstrates that simpler machine learning models can match transformer-based approaches while remaining practical for embedded deployment.**

***Keywords*—Autonomous parking; direction change detection; embedded systems; machine learning; motion planning; random forest; rapidly-exploring random trees; rapidly-exploring random tree star**

I. Introduction

Autonomous parking represents a critical advanced driver-assistance system (ADAS) capability that enhances vehicle usability and road safety. The task requires generating collision-free, kinematically feasible trajectories that navigate vehicles into parking spaces under strict real-time constraints. Perpendicular backward parking, the most common real-world scenario, demands multiple direction reversals (gear changes) and precise trajectory execution, presenting a non-trivial motion planning challenge.

Sampling-based motion planners (SBMPs) such as Rapidly-exploring Random Trees (RRT) [1] and Rapidly-exploring Random Tree Star (RRT*) [2] provide probabilistic completeness guarantees—if a collision-free path exists, these algorithms will find it given sufficient computation time. RRT* additionally offers asymptotic optimality, enabling generation of near-optimal parking trajectories. However, computational requirements for these algorithms often reach several seconds per planning instance, exceeding real-time requirements of automotive systems. Current ADAS processors allocate computational budgets primarily to perception tasks (computer vision, LiDAR processing) and end-to-end learning models for driving decisions, leaving minimal resources for classical motion planning.

Recent advances have demonstrated that learning-augmented planning can bridge this efficiency gap. The Motion Planning Transformer (MPT) framework learns spatial path structure from SBMP outputs [3], enabling faster RRT exploration through intelligent guidance. Transformer-Enhanced Motion Planner (TEMP) extends this with environmental encoders [4] that further reduce search space. Concurrently, diffusion-based approaches (e.g., Diffusion-based Parking Planner, MultiPark) have shown promise in learning complex parking distributions. However, all these approaches introduce substantial computational overhead: transformer inference requires millions of parameters and iterative attention mechanisms, while diffusion models necessitate multiple denoising iterations. For ADAS platforms where transformer or diffusion inference could dominate the planning budget, these solutions remain impractical.

This work addresses this gap by proposing a lightweight, learning-augmented planning framework specifically optimized for autonomous parking on embedded systems. Rather than employing global transformer models or diffusion processes, the approach trains per-grid-cell Random Forest classifiers and regressors to predict path occupancy (0=empty, 1=path, 2=state change, 3=direction change) and vehicle orientation. These lightweight models (20 trees, maximum depth 8) require minimal inference time while providing spatial guidance that accelerates RRT exploration. The framework may also preserve the theoretical completeness properties of RRT through a fallback mechanism that permits unconstrained exploration when learned guidance is insufficient.

The main contributions of this work are:

- A large-scale synthetic dataset of 10,725 perpendicular backward parking scenarios with ground-truth RRT* trajectories in the Reeds-Shepp configuration space [5];
- A lightweight two-tier learning framework combining Random Forest classifiers [6] for spatial path occupancy prediction with regressors for vehicle orientation estimation;
- An enhanced RRT algorithm that integrates learning-based predictions to constrain state sampling while maintaining probabilistic completeness;
- Comprehensive experimental validation demonstrating 17-fold planning time reduction (2.81 to 0.16 seconds)

*Corresponding author.

and improved success rates (98–100%) in direction-change-intensive scenarios, with performance scaling as a function of prediction confidence;

- Evidence that lightweight machine learning models achieve comparable benefits to transformer-based methods while remaining feasible for embedded ADAS deployment.

The remainder of this paper is organized as follows. Section II reviews related work in sampling-based motion planning, learning-augmented approaches, and autonomous parking systems. Section III presents the methodology, including dataset generation, model architecture, and the constrained RRT algorithm. Section IV presents experimental results with quantitative performance breakdown by prediction confidence. Section V discusses findings, computational feasibility, limitations, and future research directions. Section VI concludes with a summary of contributions and practical implications.

II. Related Work

A. Classical Sampling-Based Motion Planning

Sampling-based motion planners have dominated robotic and autonomous vehicle path planning for over two decades. The Rapidly-exploring Random Tree (RRT) algorithm [1] provides probabilistic completeness and handles high-dimensional configuration spaces effectively. RRT* [2] extends this framework with rewiring to achieve asymptotic optimality. For vehicles with kinematic constraints (e.g., non-holonomic differential steering), variants incorporating Dubins [7] and Reeds-Shepp [5] steering enable feasible trajectory generation. While theoretically robust, classical SBMPs typically require seconds to minutes on embedded platforms, limiting real-time automotive applications.

B. Learning-Augmented Motion Planning

Recent research has explored augmenting classical planners with learned models. The Motion Planning Transformer (MPT) [3] demonstrates that transformer networks trained on SBMP outputs can generalize motion planning constraints to novel configurations and environments. By predicting feasible regions, MPT enables RRT to focus exploration on high-probability areas, achieving substantial acceleration. Building on this work, Transformer-Enhanced Motion Planner (TEMP) [4] introduces an Environmental Information Semantic Encoder (EISE) to further reduce required search space through semantic understanding of scene structure.

However, transformer-based approaches incur significant computational costs. A typical transformer model contains millions of parameters and requires forward passes involving matrix multiplications across multiple attention layers, often necessitating seconds of inference time on embedded platforms. In ADAS systems where a single processing chip allocates computational budget across perception (computer vision, object detection, tracking) and driving decision models (end-to-end learning, reinforcement learning), the overhead of transformer inference can exceed motion planning itself, making such approaches impractical for real-time deployment.

C. Diffusion Models in Motion Planning

Emerging diffusion-based planning approaches have shown promise in complex motion planning tasks. Diffusion-based planning models [8] [9] leverage learned score functions to generate collision-free trajectories by iterative refinement. In the parking domain, models such as Diffusion-based Parking Planner [8] and MultiPark [9] apply diffusion processes to learn parking distributions from expert demonstrations. While effective, these methods similarly suffer from computational cost: each planning instance requires multiple denoising iterations (typically 10–50 steps), each involving forward passes through the neural network, rendering real-time inference infeasible on embedded platforms.

D. Lightweight and Embedded Planning Approaches

Prior work on embedded motion planning has explored heuristic-based methods (e.g., Hybrid A* [10] with handcrafted heuristics) and simplified geometric approaches. While computationally efficient, these methods sacrifice generality and theoretical guarantees. The current work bridges this gap by demonstrating that lightweight supervised learning models (Random Forests with limited tree depth and count) can provide sufficient spatial structure to accelerate sampling-based planning while maintaining theoretical guarantees and remaining practical for embedded systems.

III. Materials and Methods

A. Problem Formulation

This work addresses motion planning in perpendicular backward parking scenarios. The configuration space is defined as $\mathcal{C} = SE(2) \times \{F, B\}$, where a configuration $\mathbf{x} = (x, y, \theta, g) \in \mathcal{C}$ represents vehicle position $(x, y) \in \mathbb{R}^2$, heading angle $\theta \in [0, 2\pi]$, and gear state $g \in \{F \text{ (forward)}, B \text{ (backward)}\}$. The parking task is specified by start configuration $\mathbf{x}_s$ and goal configuration $\mathbf{x}_g$. The environment contains polygonal obstacles representing adjacent vehicles and parking lot boundaries.

B. Dataset Generation

1) Vehicle and scene parameterization: A vehicle model based on the Audi A4 sedan was parameterized with the following specifications: length 4.76 m, width 1.84 m, wheelbase 2.82 m. Each parking scenario is geometrically described by a feature vector $\mathbf{f} \in \mathbb{R}^{26}$ containing:

- Ten vertices (20 coordinates) representing obstacle boundaries: two triangular regions flanking the parking slot (shown in magenta in Fig. 1), one rear boundary line, and one forward maneuvering boundary line (both shown in grey in Fig. 1).
- Start frame position (2 coordinates) and orientation (1 angle): 3 values total.
- Goal frame position (2 coordinates) and orientation (1 angle): 3 values total.

Data augmentation was performed by randomly translating obstacles within a bounded region $[-10, +10]$ m^2 and rotating obstacle configurations. This procedure generated 10,725 unique parking scenarios, with failed path generations (path not found within 30-second timeout) filtered post-generation.

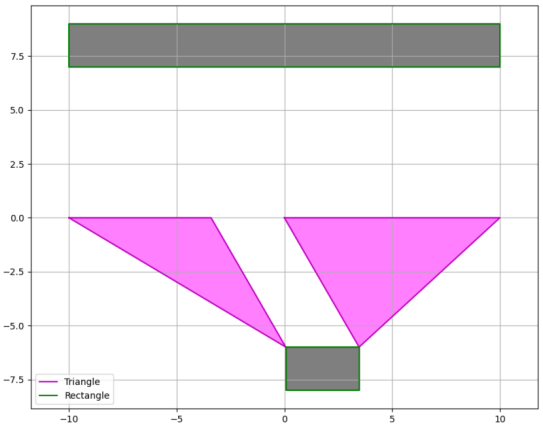


Fig. 1. Representation of the parking scene obstacles.

2) Path generation and preprocessing: For each scenario, a ground-truth parking trajectory was computed using the Open Motion Planning Library (OMPL) [11] with the RRT* algorithm and Reeds-Shepp steering to respect vehicle kinematics. Path generation was limited to a 30-second timeout per scenario.

Each resulting path was represented as a sequence of states in the Reeds-Shepp space and uniformly interpolated to exactly 100 waypoints for standardization across all scenarios (Fig. 2a).

From these interpolated waypoints all the direction/gear changes are extracted as well (Fig. 2b).

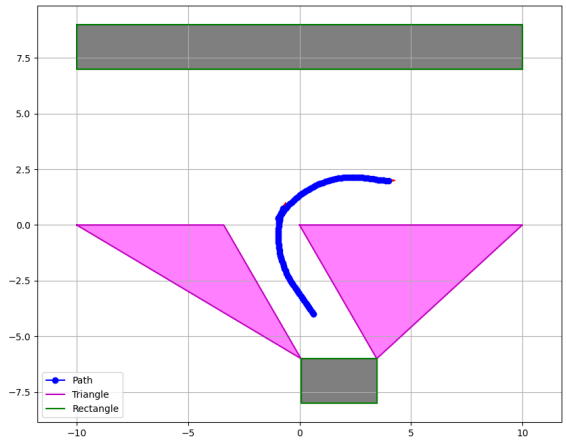
Of the initial 10,725 scenarios attempted, the 2,252 scenarios where RRT* failed to generate valid paths (e.g., scenarios with insufficient maneuvering space) were excluded from further analysis. The remaining 8,473 valid scenarios were split into training (80%, 6778 scenarios) and test (20%, 1695 scenarios) sets prior to feature extraction.

C. Grid-Based Learning Framework

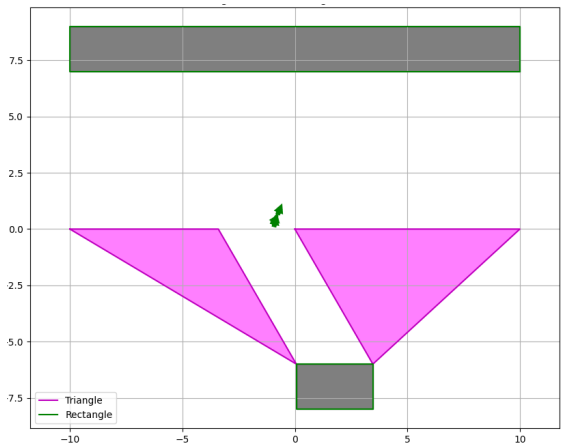
1) Spatial grid representation and labeling: A uniform spatial grid of 1 m $\times$ 1 m cells was overlaid on each parking scenario. Grid resolution was selected based on preliminary experiments: coarser resolutions (2 m) provided insufficient spatial discrimination, while finer resolutions (0.5 m, 0.25 m) increased model complexity and inference latency without improving planning success rates.

For each grid cell, the following information was extracted from the RRT* reference path:

- Occupancy class (Displayed in Fig. 3a) $c_i \in \{0, 1, 2, 3\}$: Maximum class among cells crossed by the path, where:
  - Class 0 (empty): cell contains no path waypoints
  - Class 1 (path): cell contains interpolated waypoints



(a) Representation of the 100-waypoint interpolated parking path.



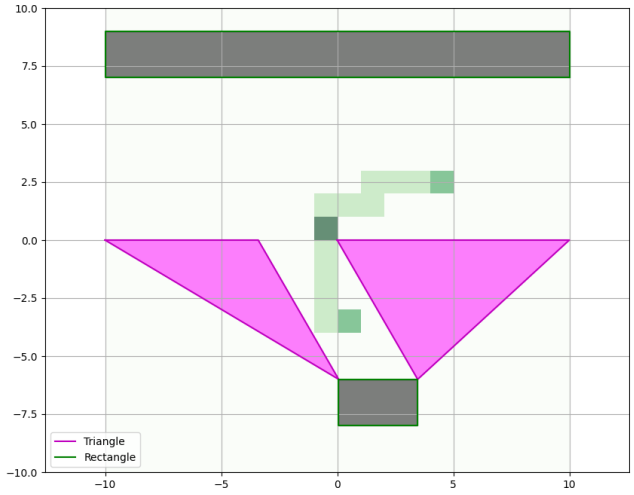
(b) Representation of direction changes in the path.

Fig. 2. Extracted information from parking path.

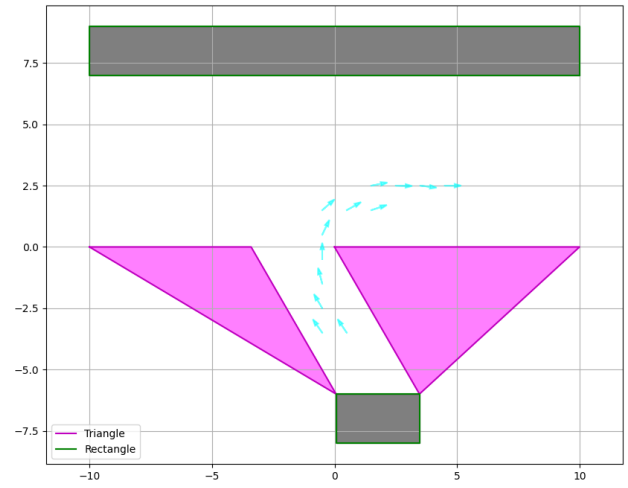
  - Class 2 (state change): cell contains a configuration where the path generates or ends a Reeds-Shepp curve (this includes the start and end locations)
  - Class 3 (direction change): cell contains a gear change
- Orientation (Displayed in Fig. 3b) θ_i: Average heading angle of all waypoints within the cell

A direction change is formally defined as a transition in gear state (forward $\rightarrow$ backward or backward $\rightarrow$ forward), equivalent to a reversal in driving direction.

2) Model architecture and training: For each grid cell, a separate Random Forest classifier was trained using the feature vector $\mathbf{f} \in \mathbb{R}^{26}$ as input and the occupancy class c_i as target. Random Forest models were selected for their non-parametric nature, robustness to feature scaling, and minimal inference overhead—critical for embedded deployment. The hyperparameter selection was: 20 decision trees with maximum tree depth of 8.



(a) Representation of the path grid overlay.



(b) Representation of the orientation overlay.

Fig. 3. Representation of the path data after preprocessing.

Orientation prediction was performed separately. Initial cells lacking waypoint coverage were marked as missing. A K-Nearest Neighbors imputer ($k = 5$) was applied to estimate missing orientations from neighboring cells. Subsequently, a separate Random Forest regressor (20 trees, maximum depth 8) was trained for each grid cell to predict heading angles from the feature vector $\mathbf{f}$, using the imputed orientations as training targets. This output data is displayed in Fig. 4.

D. Constrained RRT with Learned Guidance

The proposed planning algorithm augments standard RRT with a custom state sampler that integrates predictions from the trained models:

- Given a parking scenario, extract the feature vector $\mathbf{f}$
- Query the trained classifiers to obtain per-cell occupancy predictions: $\hat{c}_i$ for all grid cells i
- Query the trained regressors to obtain per-cell orientation predictions: $\hat{\theta}_i$ for all grid cells i

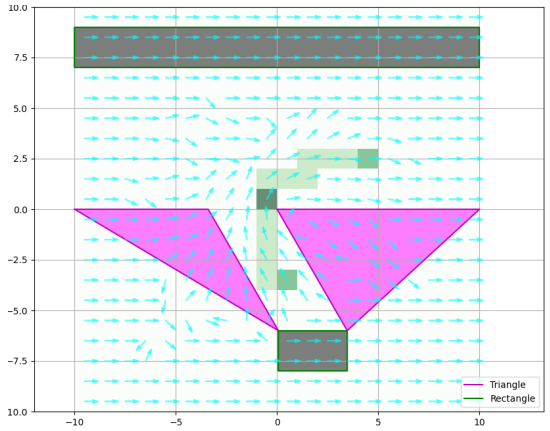


Fig. 4. Representation of the output vector data.

- During RRT tree expansion:
  - (Standard RRT sampling) Generate a random configuration $\mathbf{x}_{\text{rand}}$ in the configuration space
  - (Guided sampling) If $\mathbf{x}_{\text{rand}}$ projects to a grid cell with predicted occupancy $\hat{c}_i > 2$ (gear change) or part of the start/end cells, accept the state; otherwise, reject and resample
  - (Orientation constraint) If accepted, perturb the heading component of $\mathbf{x}_{\text{rand}}$ to lie within $[\hat{\theta}_i - 45°, \hat{\theta}_i + 45°]$
  - (Fallback mechanism) If valid state sampling fails after $N = 100$ consecutive rejection attempts, switch to unconstrained uniform sampling for a configurable n sampling attempts
- Continue standard RRT tree extension toward the goal configuration

This formulation preserves the probabilistic completeness of RRT: if a solution exists and sufficient time remains, the fallback mechanism ensures that the algorithm explores the full configuration space and eventually discovers a path.

E. Experimental Validation Protocol

For each of the 1695 test scenarios:

- Execute unconstrained RRT (baseline) with 30-second planning timeout
- Execute constrained RRT (proposed method) with identical timeout
- Record: success/failure, planning duration, number of tree nodes created
- Post-hoc: count number of direction changes predicted in that scenario's grid

Success is defined as finding any collision-free path within the timeout. All experiments were conducted on a standard

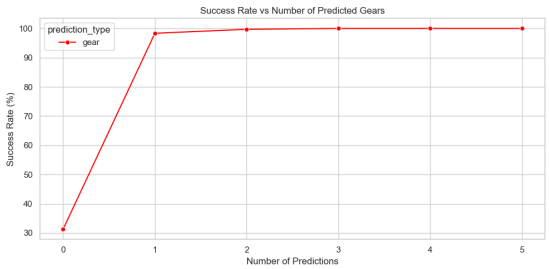
desktop workstation (AMD Ryzen 5 CPU, 32 GB RAM, no GPU acceleration) using OMPL with Python bindings and scikit-learn [12] for model training and inference.

IV. Results

A. Overall Performance Summary

Table I presents aggregate results across all 1695 test scenarios. Unconstrained RRT achieved 96.4% success rate with mean success time of 2.81 seconds and overall mean time of 3.79 seconds (including failed attempts). The constrained RRT approach achieved 71.9% overall success with a mean success time of 0.35 seconds and an overall mean time of 8.68 seconds; however, this aggregate masks important performance variation correlated with prediction confidence (detailed in Section IV-B).

The low overall success rate and high mean time for constrained RRT in the aggregate reflect scenarios with sparse or no direction change predictions (detailed in Section IV-C). These scenarios comprise approximately 40.18% of the test set, where prediction guidance is unreliable, and the constrained sampler restricts exploration too severely, causing timeout failures.



(a) Success graph with respect to number of predictions.

(b) Timing graph with respect to number of predictions.

Fig. 5. Constrained RRT performance with respect to number of predictions.

B. Performance Breakdown by Prediction Confidence

The key insight emerges when analyzing performance conditional on the number of direction change predictions provided by the learned model. Table II presents results grouped by direction change prediction count.

We can see in Fig. 5a the positive trend of success rate with number of predictions and in Fig. 5b we can see the inverse relation in runtime as the number of predictions increase.

Performance exhibits a clear correlation with prediction confidence:

- No predictions (0 cells): 33.2% success. In these scenarios, the constrained sampler restricts states to empty grid cells, preventing effective exploration. Timeout failures dominate.
- Single prediction (1 cell): 98.3% success, 0.15 s mean time. Minimal guidance proves sufficient to accelerate planning significantly.
- Two predictions: 99.7% success, 0.15 s mean time.
- Multiple predictions (3–5 cells): 100% success, 0.17–0.25 s mean time.

For scenarios with at least one direction change prediction, the constrained RRT dramatically outperforms unconstrained RRT in both success rate and planning time, achieving 17-fold speedup (2.81 to 0.16 seconds).

C. Failure Mode Analysis

Two distinct failure modes emerge:

1) Constrained RRT failures: Occur primarily when direction change predictions are sparse or absent (0-1 cells). In these scenarios, the learned model failed to capture the true solution structure, either because 1) the test scenario poorly matches the training distribution, or 2) the direction change count is inherently low, providing minimal guidance to the state sampler. The fallback mechanism prevents complete failure but provides no advantage over unconstrained search, explaining the degraded performance.

2) Unconstrained RRT failures: 3.6% of scenarios exceed the 30-second timeout without finding a valid path. These are intrinsically difficult configurations (e.g., very tight parking spaces) that require extensive exploration.

V. Discussion

A. Interpretation and Trade-offs

The results reveal a fundamental trade-off between search coverage and computational efficiency. Unconstrained RRT explores the entire configuration space uniformly, providing robust coverage but requiring extensive computation. Constrained RRT leverages learned predictions to focus exploration toward high-probability path regions, dramatically reducing computational cost but at a loss of coverage in low-prediction scenarios.

The strong performance in high-prediction scenarios (98.3–100% success with 0.15–0.25 s planning) demonstrates that direction changes create a highly structured planning problem. The learned models effectively capture this structure, providing guidance that exceeds random exploration in value. This supports the hypothesis that parking geometry imposes strong constraints on feasible solution paths—constraints that modest machine learning models can capture with reasonable accuracy.

TABLE I. OVERALL PLANNING PERFORMANCE (ALL TEST SCENARIOS)

Method	Success Rate	Mean Success Time	Mean Overall Time
Unconstrained RRT	96.4%	2.81 s	3.79 s
Constrained RRT (Proposed)	71.9%	0.35 s	8.68 s

TABLE II. PERFORMANCE BY NUMBER OF DIRECTION CHANGE PREDICTIONS

Prediction Count	Percentage of Scenarios	Success Rate	Mean Success Time	Mean Overall Time
0 (no prediction)	40.18%	31.2%	1.22 s	21.05 s
1	21.30%	98.3%	0.15 s	0.65 s
2	19.82%	99.7%	0.15 s	0.24 s
3-5	18.70%	100.0%	0.17–0.25 s	0.17–0.25 s
All >0 predictions	59.82%	99.3%	0.16 s	0.37 s

B. Computational Feasibility for Embedded ADAS

Studies on the deployment of Random Forest (RF) models in embedded environments have demonstrated their suitability for embedded ADAS applications. Kuppers et al. reported achieving an inference time of approximately 700 microseconds per RF classifier on a test platform [13]. Accordingly, the 400 RF classifiers used in this work result in a total inference time of about 0.28 seconds. The analysis further indicates that many grid cells remain unutilized, suggesting that targeted grid pruning could reduce computational overhead and enhance runtime efficiency. While this overhead remains negligible compared to the overall planning time observed in unconstrained RRT, the models retain a lightweight footprint, as summarized below:

- Model size: Daghero et al. successfully deployed RF models on microcontrollers with only 520 kB of available memory [14]. Also, recent work by Kochel et al. demostrates efficient tree ensemble inference (including Random Forest) on ARM-based embedded devices [15].
- Memory access pattern: Tree traversal operations demonstrate high CPU cache locality.
- No specialized hardware required: Inference executes efficiently on standard CPUs, reducing both power consumption and thermal load.

In contrast, transformer-based approaches require millions of parameters and iterative attention computations, often necessitating GPU acceleration and power budgets unsuitable for ADAS platforms. Diffusion-based methods similarly require multiple denoising iterations, compounding computational overhead. The lightweight Random Forest framework enables deployment on resource-constrained embedded processors currently used in many production ADAS systems.

C. Comparison with Learning-Based Planning Methods

The proposed method achieves competitive results relative to transformer-based approaches while remaining practical for embedded deployment. MPT and TEMP require transformer inference, estimated at 50–500 ms per planning instance on embedded platforms (depending on model size and hardware). MultiPark and diffusion-based methods require multiple denoising iterations, further increasing latency. The proposed constrained RRT achieves 0.15–0.25 s planning in high-prediction scenarios, comparable to or better than these methods.

This efficiency gain comes from task specialization: unlike MPT (general motion planning) or diffusion models (learning complex distributions), the current work targets a narrower domain (parking) with a simpler learning objective (binary occupancy and continuous orientation prediction). This trade-off—generality for efficiency—is appropriate for ADAS applications where parking represents a well-defined, repeatedly executed task.

D. Limitations

Several important limitations merit discussion:

- Parking type specificity: Models are trained exclusively on perpendicular backward parking. Generalization to parallel, angled, or diagonal parking requires retraining on scenario-specific data. The 10,725 scenario budget was chosen for manageable training time; larger datasets could improve robustness but at higher computational cost.
- Vehicle specificity: Models are parameterized for the Audi A4 sedan. Vehicles with different wheelbase lengths, turning radii, or dimensions require new training. Transfer learning approaches (pre-training on diverse geometries, fine-tuning) may mitigate this limitation but were not explored here.
- Prediction reliability in distribution shift: When test scenarios poorly match the training distribution (e.g., parking slots with unusual obstacle arrangements not seen during training), learned predictions become unreliable, degrading constrained RRT performance below baseline.
- Grid resolution trade-off: The 1 m grid balances spatial accuracy with model complexity. Finer grids (0.5 m, 0.25 m) could improve precision but increase the number of per-cell models proportionally, increasing inference latency and memory footprint. Coarser grids reduce models but sacrifice spatial resolution.
- Baseline comparison limitation: Comparison is against unconstrained RRT only. Quantitative comparisons with transformer-based methods (MPT, TEMP) or

diffusion models on identical scenarios are not available, limiting direct competitive assessment. These comparisons would require access to proprietary implementations.

- Hyperparameter sensitivity not analyzed: The choice of 20 trees and depth 8 was based on quick manual comparisons but ablation studies comparing alternative configurations (5 trees, 50 trees, varying depths) are not presented, limiting understanding of sensitivity to these choices.

E. Future Research Directions

Future work should address:

- Multi-parking-type unification: Development of a single learning model that handles perpendicular, parallel, and angled parking simultaneously, trading per-type accuracy for unified generality.
- Grid Cell Predictor Pruning: In depth analysis of grid cell predictor utility for pruning and possibly explore a learning model to determine which grid cells are worth inference.
- Transfer learning: Pre-training on diverse synthetic vehicle geometries and parking types, then fine-tuning for specific vehicles, to reduce per-vehicle training data requirements.
- Adaptive constraint relaxation: Dynamic adjustment of sampling constraints (grid cell restrictions, orientation bounds) based on prediction confidence, maintaining completeness while improving robustness.
- Alternative SBMP variants: Extension to RRT*, Bidirectional RRT, and other planners to assess whether learned predictions similarly accelerate different algorithm variants.
- Real-world validation: Deployment on actual autonomous vehicles with real sensor data, dynamic obstacles, and sensor noise to assess practical performance beyond simulation.
- Hardware implementation: Optimization of inference on actual ADAS processors (e.g., Qualcomm Snapdragon, NVIDIA DRIVE) to quantify practical deployment feasibility.

VI. Conclusion

This paper presented a lightweight, learning-augmented motion planning framework for real-time autonomous parking on embedded ADAS platforms. By training Random Forest classifiers and regressors to predict parking trajectory structure (occupancy classes and vehicle orientation) from scenario geometry, the approach enables accelerated RRT planning without sacrificing theoretical completeness guarantees. Experimental validation on 1695 test scenarios demonstrated that scenarios with sufficient prediction guidance achieve 98.3–100% success rates with 17-fold planning time reduction (2.81 to 0.15–0.25 seconds) compared to unconstrained RRT.

The core contribution is demonstrating that lightweight supervised learning models (20 trees, maximum depth 8) can achieve planning acceleration comparable to transformer-based approaches while remaining practical for embedded automotive deployment. This addresses a critical gap in current ADAS systems, where computational budgets are dominated by perception and end-to-end learning, leaving minimal resources for classical motion planning.

As autonomous parking systems mature toward production deployment, efficient planning methods that balance computational constraint, memory footprint, and solution quality will become increasingly critical. This work demonstrates a practical path toward real-time, reliable parking planning suitable for resource-constrained ADAS platforms, with clear pathways toward generalization to multiple parking types and vehicle platforms through transfer learning and multi-task approaches.

Acknowledgment

The authors gratefully acknowledge the Open Motion Planning Library (OMPL) developers for providing a comprehensive, open-source motion planning framework essential for this research.

References

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *IEEE Computer Society*, vol. 98, no. 11, pp. 1–4, 1998, technical Report 98-11.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[3] J. J. Johnson, U. S. Kalra, A. Bhatia, L. Li, A. H. Qureshi, and M. C. Yip, "Motion planning transformers: A motion planning framework for mobile robots," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1–7.

[4] L. Zhuang *et al.*, "Transformer-enhanced motion planner," *IEEE Robotics and Automation Letters*, 2024.

[5] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[6] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[8] M. Jiang, Y. Li, J. Zhang, S. Zhang, and M. Yang, "A diffusion-refined planner with reinforcement learning priors for confined-space parking," *arXiv preprint arXiv:2510.14000*, 2025.

[9] H. Zheng, Z. Zhou *et al.*, "Multipark: Multimodal parking transformer with next-segment prediction," *arXiv preprint arXiv:2508.11537*, 2025.

[10] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[11] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] F. Küppers, J. Kronenberger, A. Shantia, and A. Haselhoff, "Multivariate confidence calibration for object detection," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1322–1328.

[14] F. Daghero, A. Burrello, C. Xie, L. Benini, A. Calimera, E. Macii, and M. Poncino, "Adaptive random forests for energy-efficient inference on microcontrollers," *IEEE Transactions on Computers*, vol. 71, no. 11, pp. 2924–2937, 2022.

[15] S. Koschel, S. Buschjäger, C. Lucchese, and K. Morik, "Fast inference of tree ensembles on arm devices," *arXiv preprint arXiv:2305.08579*, 2023.

APPENDIX A
DATASET STATISTICS AND DISTRIBUTION

The parking scenario dataset exhibits the following detailed characteristics:

- **Initial scenarios generated:** 10,725
- **Waypoints per trajectory:** 100 (uniformly interpolated)
- **Scenarios with valid paths:** 8,473 (79% after filtering failures)
- **Training set:** 6778 scenarios (80% of valid paths)
- **Test set:** 1695 scenarios (20% of valid paths)
- **Feature dimension:** 26 $\{10$ obstacle vertices (x, y) + 1 start frame (x, y, θ) + 1 goal frame $(x, y, \theta)\}$
- **Spatial domain:** $[-10, +10]$ m^2
- **Grid cells per scenario:** 400 (20×20 cells with 1 m resolution)

APPENDIX B
HYPERPARAMETER SELECTION AND MODEL PERFORMANCE

Random Forest models were trained using scikit-learn

- **Number of trees:** 20; 100 trees tested as alternative
- **Maximum tree depth:** 8
- **Minimum samples per leaf:** 1
- **Feature sampling:** $(\sqrt{n_{\text{features}}})$ for classification; (n_{features}) for regression
- **KNN imputer neighbors:** $k = 5$

APPENDIX C
IMPLEMENTATION DETAILS AND REPRODUCIBILITY

A. Software and Hardware

Training and inference experiments were conducted using:

- **Motion Planning:** OMPL 1.7.0 with Python 3.9 bindings
- **Machine Learning:** scikit-learn 1.7.2, NumPy 2.3.2, Pandas 2.3.1
- **Hardware:** AMD Ryzen 5 CPU, 32 GB RAM
- **Operating System:** Ubuntu 24.04 LTS

B. Computational Time Breakdown

- **Reference path generation (RRT*):** 90 hours total (exhausting 30-second timeout per scenario for 10,725 scenarios)
- **Data preprocessing (waypoint interpolation, grid generation):** 2–3 hours
- **Model training (KNN imputation + RF classifier/regressor):** 15–30 minutes

C. Code Availability

Custom RRT state samplers were implemented as extensions to OMPL's C++ API. All code is modular, well-documented, and available upon request for reproducibility and future research.