Intelligent Path Planning for Robotics Using Software Defined Networks

Shahad Jumaa Shaaban and Nadia Adnan Shiltagh Al-Jamali
Department of Computer Engineering, University of Baghdad, 10071 Baghdad, Iraq
gs22.sjshaban@coeng.uobaghdad.edu.iq, Nadia.aljamali@ coeng.uobaghdad.edu.iq

Keywords: Software-Defined Networking (SDN), Spiking Wavefront Planner (SWP), Super Elman Neural Network (SE-SNN), E-Prop, Path Planning.

Abstract: This paper proposes a hybrid learning-based navigation framework that integrates classical path planning with a spiking neural network model to achieve scalable and computationally efficient robot navigation in grid-based environments. Optimal reference trajectories are generated using the Spiking Wavefront Planner (SWP) over occupancy maps derived from the KITTI Odometry dataset. These expert paths are employed to train a Super Elman Spiking Neural Network (SE-SNN) using the e-prop learning rule to capture sequential navigation decisions. The framework is evaluated across map sizes ranging from 100×100 to 400×400 and compared with classical planners such as A* and D* Lite. Results show that the proposed SWP-SE-SNN model reduces planning time by up to 94% compared to SWP and approximately 49% relative to A* on large-scale maps, while maintaining comparable or shorter path lengths with a 100% goal-reaching rate. The inference stage exhibits stable runtime behavior as environment size increases, supporting real-time deployment within SDN-based robotic architectures.

1 INTRODUCTION

Autonomous mobile robot navigation has emerged as an important research field in the last years with applications to large-scale robotic systems, and cyber-physical environments [1]. Path planning is an essential task for autonomous robot navigation that provides collision-free and feasible trajectories in terms of environmental danger with respect to environmental boundaries and robot kinematics [2]. Recent surveys report that classical graph-based path planning techniques, such as A* and incremental search planners, are still popular, but they do not scale well in computational time with the size of the environment, especially when it is large or obstacle-dense, thus restricting their applicability to real-time performance. Learning based navigation approaches have recently become popular for addressing these limitations, where the goal is to minimize the online planning time by learning navigation policies from data [3], [4]. Spiking Neural Network (SNN) models have come to prominence in the recent past as biologically inspired and low energy computing models that are applicable for real-time robot control [5], [6]. Recurrent SNN schemes such as ESNN have proposed based on SNN and e-prop that at generated from biological learning rule which do not rely on backpropagation up to our knowledge is more efficient for sequential task like navigation than other word [7], [8]. Considerable interest is being generated around the recent innovations in network control, such as Software Defined Networking (SDN), which offers a flexible and centralized control architecture for networking and cyber-physical systems to have intelligent control logic realized as application-level controllers on the control plane [9], [10]. Inspired by these advances, this paper introduces a two-stage navigation mechanism in which the SWP engines pre-compute collision-free paths and the online decision making is abstracted to a Super Elman Spiking Neural Network (SE-SNN) using e-prop, which conceptually behaves like an SDN-based controller.

The proposed approach achieves reduced path length and planning time compared to classical planners, demonstrating its suitability for real-time and network-assisted robotic navigation.

2 RELATER WORK

Recent work in the area of mobile robot navigation has been concentrating on the improvement of path planning performance in large-scale and high-dimensional environments. D* Lite and A* are classical planning methods that are still considered baselines for their deterministic nature, completeness and optimality guarantees, but some studies show a degradation in their performance with increasing map size or obstacle density [11], [12].

To address scalability and real-time limitations, learning-based navigation approaches have been extensively studied. Deep RL and imitation learning techniques have shown potential in minimizing the online planning time by directly learning navigation policies from interaction data [13], [14].

Although successful, these techniques tend to rely on large training sets and may struggle to generalize to unfamiliar conditions [15]. Look-up table navigation with learning-based models. Recently, hybrid frameworks combining classical planning with learned policies have become of interest. In most of these approaches, the planners are classical, offline-generated, and safe, and are feasible for experience acquisition to train neural networks, thereby avoiding the need to use them at test time [16], [17]. These frameworks aim to uphold the safety guarantees of classical algorithms while leveraging learned inference efficiency.

On the other hand, Spiking Neural Networks (SNNs) have received growing attention for robotic control and navigation, thanks to their temporal processing abilities and energy efficiency [18]. For navigation and motion control tasks such as this, it has been found that one class of SNNs, namely the recurrent SNN, is well-suited in recent studies [19], [20]. Further, such biologically plausible learning rules, such as e-prop (with derived learning time scales), allow for training recurrent SNNs in a computationally efficient manner without the need for backpropagation through time and offer promise for real-time and adaptive robotic applications [21], [22].

At the systems level, Software-Defined Networking (SDN) has been considered as an enabling technology of networked and cyber-physical robotic systems. New literature has also focused on the ability of SDN to enable centralized, programmable control, thereby enabling intelligent algorithms to be employed as application-level controllers for robotic coordination and control [23], [24]. However, the integration of SNN-based navigation models with SDN-centered control architectures has received less attention.

In contrast to existing studies, this work proposes a two-stage navigation framework that integrates offline path generation using a Spiking Wavefront Planner with an online Super Elman Spiking Neural Network trained using e-prop, conceptually designed to operate as an intelligent controller within an SDN-based architecture.

3 DATASET AND ENVIROMENT MODELING

To guarantee a realistic and reproducible evaluation, this study uses real-world motion data (obtained from the KITTI Odometry Dataset) that features ground-truth vehicle pose and trajectory acquired in outdoor scenarios. The dataset is used to extract realistic motion constraints and spatial features that reflect navigational conditions in a real-world environment [25]. The grid-based maps are built to model the navigation world using the estimated parameters. The grid has cells for free space and obstacles, where obstacle size and distribution are determined based on motion properties learned from the dataset. This model-driven method enables the simulation environment to simulate both natural navigation constraints and not just invalid conditions [23]. They produce trajectories that respect the structure of the environment and movement limitations learned from real-world data; these generated motions are used as expert demonstrations to train a navigation controller with learning [25]. We do not simulate dynamics but rather learn from real-world trajectories to generate meaningful trajectories that are environment-aware. This data-based approach provides a fairly sound basis for verifying the effectiveness of the presented navigation controller in an environment similar to real robot navigation.

4 PROPOSED FRAMEWORK

This section introduces the learning-based navigation framework that couples classical path planning with spiking neural learning. The framework is two-stage: (i) generate expert trajectories with the SWP, and (ii) refine the trajectories by training the SE-SNN using the e-prop learning rule.

4.1 Overview of the Framework

The proposed system is deployed within a layered SDN-based robotic architecture that ensures logical separation between application, control, and infrastructure layers. The SE-SNN navigation module operates at the application layer, while the SDN controller provides centralized orchestration, resource management, and task coordination across distributed robotic agents. The focus of this work is the intelligent navigation mechanism within this SDN-enabled framework rather than network-layer performance benchmarking. The navigation pipeline we introduce is tailored to integrate the robustness of classical planning with the flexibility of learning-based approaches. First, SWP is used to compute a feasible navigation path in a grid-based motion environment created from real-world movements. However, instead of being the ultimate answer, this initial trajectory is considered an expert solution for training the controller through learning.

Then, the trained expert trajectories are used to train the constructed SE-SNN. By learning with e-prop, the network learns to capture spatiotemporal navigation strategies and fine-tune local decision-making. evaluated on a trained SE-SNN, generates navigation trajectories independently, and is compared with classical planning algorithms.

Figure 1 illustrates the layered SDN-enabled robotic architecture, highlighting the interaction between the navigation intelligence module and centralized control orchestration.

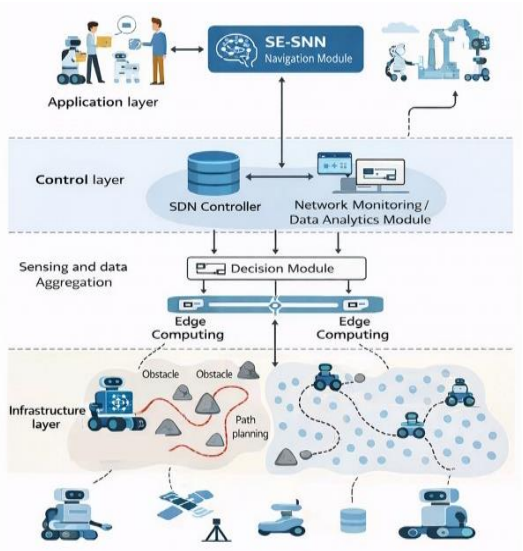

Figure 1: Overall pipeline of the proposed SWP+SE-SNN navigation framework.

4.2 Expert Trajectory Generation Using SWP

The process begins with environment modeling based on real-world motion characteristics derived from the KITTI dataset. The SWP serves as a tool for initial navigation paths in the simulated grid environment. SWP transmits wavefront signals originating from the goal to free cells up to the start position, ensuring collision-free, feasible paths.

While SWP can generate accurate trajectories, it does not directly optimize for learning-based refinement or real-time execution time. Hence, observed paths are treated as near-optimal expert demonstrations rather than as the final navigation solutions.

4.3 C. Super Elman Spiking Neural Network Architecture

The learning-based controller we consider is a particular implementation of the SE-SNN that extends the classical Elman recurrent architecture with spiking neurons and recurrent context connections. A feedback loop enables the network to converge and learn the temporal dynamics of navigation decisions, thereby tackling sequential path-planning problems.

The SE-SNN takes local spatial information extracted from a grid map as input, and produces reactive navigation decisions or actions at all time-steps. By utilizing spiking dynamics and temporal spiking through a recurrent memory, the network effectively encodes both spatial and temporal elements of the navigation task.

4.4 Training with e-prop Learning Rule

The SE-SNN is trained using the e-prop learning rule, enabling bio-plausible and efficient online learning in spiking neural networks. SWP learned expert trajectories are used as supervision signals during training.

In e-prop, synaptic updates can be computed from local eligibility traces and global learning signals, enabling the network to generalize beyond mere imitation. The SE-SNN-trained model can therefore further improve navigation behavior, alleviate unwanted motions, and generate smoother, shorter trajectories compared to the expert planner.

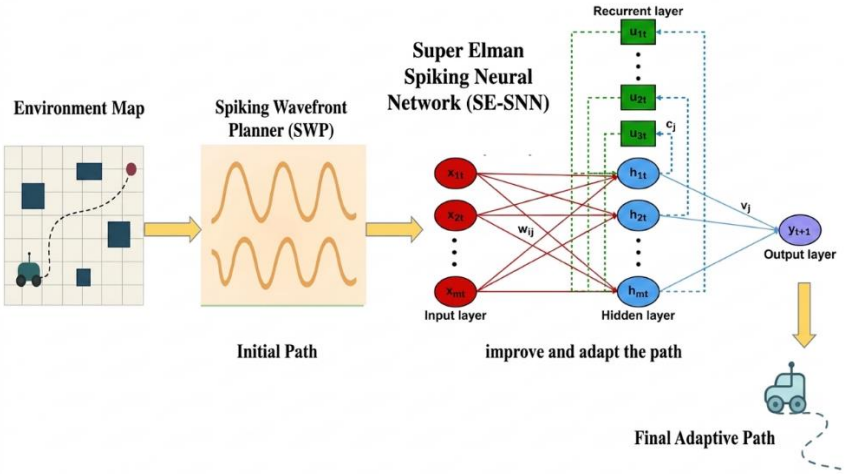

Figure 2: Architecture of the proposed Super Elman Spiking Neural Network trained using the e-prop learning rule.

Figure 2 illustrates the overall pipeline of the proposed navigation framework. The process begins with environment modeling based on real-world motion characteristics derived from the KITTI dataset. An initial feasible trajectory is generated using the SWP, which serves as expert guidance rather than a final solution. These expert trajectories are then used to train the proposed SE-SNN using the e-prop learning rule. After training, the SE-SNN independently generates refined navigation trajectories that are evaluate. d

4.5 Evaluation Strategy

After training, the SE-SNN is evaluated independently in the same grid-based environments. The generated trajectories are quantitatively compared with classical planning algorithms, including A* and D* Lite, using metrics such as path length and planning time. Experiments are conducted across multiple map sizes to assess robustness and scalability. Figure 3 presents the complete operational flow of the proposed hybrid navigation framework, including both offline training and online inference stages.

5 MATHEMATICAL FORMULATION OF THE PROPOSED SE-SNN

The mathematical formulation of the proposed SE-SNN is presented in this subsection. The formulation is fully consistent with the network architecture illustrated in Figure 2 and describes the dynamics of the input, hidden recurrent, and output layers, as well as the learning mechanism.

5.1 Network Layers Description

The SE-SNN is composed of three primary layers. The input layer represents the navigation state obtained from the environment and the initial route computed by SWP. The hidden layer is composed of spiking neurons with Elman-style recurrent context connections for extracting temporal dependencies. There's a discrete action output layer that maps each neuron to a potential direction.

5.2 Spiking Hidden Layer Dynamics

Let:

$$\mathrm{x}(t) = [x_1(t), x_2(t), \dots, x_m(t)], \qquad (1)$$

denote the input layer, where $x_i(t)$represents the encoded navigation state at time t. The hidden spiking neurons are denoted by :

$$\mathrm{h}(t) = [h_1(t), h_2(t), \dots, h_n(t)], \qquad (2)$$

The membrane potential of each hidden neuron follows the Leaky Integrate-and-Fire (LIF) model:

$$\tau_{\mathrm{m}} \frac{dh_j(t)}{dt} = -h_j(t) + \sum_i w_{ij}^{xh} x_i(t) + \sum_k w_{kj}^{hh} h_k(t-1), \qquad (3)$$

where w_{ij}^{xh} are the input-to-hidden synaptic weights, w_{kj}^{hh} are the recurrent context weights, and τ_{m} is the membrane time constant. A spike is generated when $h_j(t)$ exceeds a firing threshold, after which the neuron is reset.

This recurrent formulation enables the network to exploit temporal information from previous navigation states.

5.3 Output Layer and Action Selection

The output layer is represented by

$$y(t) = [y_1(t), y_2(t), \dots, y_K(t)], \quad (4)$$

where each output neuron corresponds to a discrete navigation action. The output activations are computed as:

$$y_k(t) = \sum_j w_{jk}^{hy} h_j(t), \quad (5)$$

with w_{jk}^{hy} denoting the hidden-to-output weights. The navigation action at time t is selected as:

$$a(t) = \arg\max_k y_k(t), \quad (6)$$

which allows direct mapping from the network activity to a motion command.

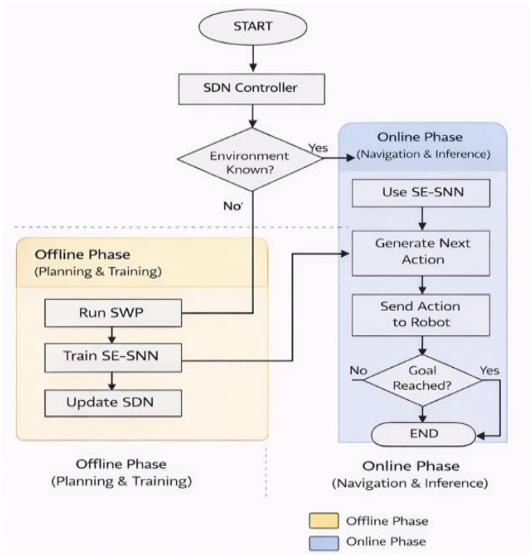

Figure 3: Operational flowchart of the proposed hybrid SWP-SE-SNN system.

5.4 Learning and Weight Update Using e-prop

Learning in the SE-SNN is performed using the e-prop learning rule, which updates the synaptic weights based on local eligibility traces and a global learning signal:

$$\Delta w_{ij} = \eta \sum_t L(t) e_{ij}(t), \quad (7)$$

where η is the learning rate, $L(t)$ is the learning signal from the expert trajectory, and s is the eligibility trace for the synapse between neurons i and j .This update rule enables efficient temporal credit assignment, and the network is built on the initial SWP-generated path by reinforcing good state-action transitions. This formula shows how encoding, decoding spiking neuron dynamics, recurrence in time, memory, and e-prop learning are combined in the SE-SNN. The model presented enables efficient learning from expert traces and facilitates the generation of more optimal, computationally efficient navigation paths. The training process of the SE-SNN remains stable due to the localized nature of the e-prop update rule, which prevents uncontrolled weight divergence. During online inference, no adaptive updates occur, ensuring deterministic and stable navigation decisions.

6 EXPERIMENTAL SETUP AND RESULTS

6.1 Experimental Setup

The experimental results are conducted in realistic grid-based environments using real-world motion data from the KITTI Odometry Dataset. To evaluate the generalization and scalability, trials were conducted on four map sizes of 100×100, 240×240, 320×320, and 400×400 with 8-connected motion.

The presented method was evaluated against classical path planning algorithms (A* and D* Lite) that serve as benchmarks for robotic navigation. The evaluation of all algorithms is done using the same set of environment parameters to ensure a fair comparison. Three main performance measures were employed in the evaluation:

1) Path Length, which measures the total distance traveled by the robot from start to goal.
2) Planning Time, which represents the computational time required to generate a feasible navigation path.
3) Represents the success of the navigation algorithm in reaching the goal state, reflecting the reliability of the planned trajectory.

Classical planners exhibit computational complexity that scales with grid expansion and environment size. In comparison, the trained SE-SNN performs forward inference based on fixed network parameters, making online decision time largely independent of exhaustive map exploration. This distinction explains the improved runtime scalability observed in multi-query evaluations.

6.2 Single-Query Performance Analysis

This subsubsection examines the performance of the proposed navigation framework using a single-query planning setup with varying map sizes (100×100, 240×240, 320×320, and 400×400) and an 8-connected motion model, in terms of planning time. The previous numbers can be readily explained by the internal workings of each planning algorithm. On the 100×100 scale, SWP finds a solution with a path length of 68.46, which can be attributed to wavefront propagation first ensuring feasibility without regard to short distances. When trained, our SWP+SE-SNN recovers the path length to 59.57 by learning state action transitions from the expert trajectory and removes redundant changes of direction through recurrent temporal context, thereby following more direct paths towards the goal.

At planning time, SWP costs 0.1902 s , making use of the full wavefront expansion to generate a motion path, and the trained SE-SNN generates motion decisions using feedforward inference, reducing the planning time down to 0.0119 performance: Though A* gets a relatively short path (67.23), its graph search overhead still results in higher computational cost (0.0314s) over neural inference. 51 longer paths (70.12) and higher planning time levels (0.2023 s) are observed in D* Lite because of cost update redundancy. This behavior persists as the map size increases to 240×240, 320×320, and 400×400. The length of the path produced by SWP increases with environment size (e.g., 194.47, 250.02, and 285.12), since its propagation respects each individual's step. By utilizing temporal memory and learned motion patterns, which predict future states rather than react locally, the SE-SNN outperforms on path lengths (175.45, 239.87, and 279.55).

The difference in planning time becomes less pronounced as the problem size increases. Although planning time of SWP and D* Lite exponentially increases (up to 3.5788 s and 3.2546 s, for the 400×400 map), method SE-SNN has a low fixed time planning cost (0.2015 s) given that neural inference does not depend on the size of map. While A* is guided by heuristics, its planning time (0.3968 s) exhibits a reliance on map complexity and consequently does not perform as fast as the learned method. In conclusion, the numbers in Table 1 show that SWP+SE-SNN learns shorter paths by tuning expert trajectories and delivers much lower planning times thanks to spiking neural inference replacing search-based planning, while achieving 100% goal reachability across all environments.

Figure 4 illustrates the path length comparison across all tested map sizes. The proposed SWP+SE-SNN consistently achieves the shortest path length in all environments compared to SWP, A*, and D* Lite. This improvement indicates that the learning-based model not only follows the expert trajectories generated by SWP but also refines them by eliminating redundant movements and smoothing local navigation decisions. Notably, the advantage of the proposed approach becomes more evident as the map size increases, demonstrating its ability to maintain path optimality in larger and more complex environments. And Figure 5 compares planning time for the single-query case. As shown in and D* Lite, classical planners (such as SWP and D* Lite) take exponential time to compute with the size of environment because repeated search is required. We remark that our SWP+SE-SNN, on the other hand, mostly has uniform planning costs across map sizes, with a very slight increase as the map size grows. It shows that our learned navigation controller is scalable and can be applied in real time.

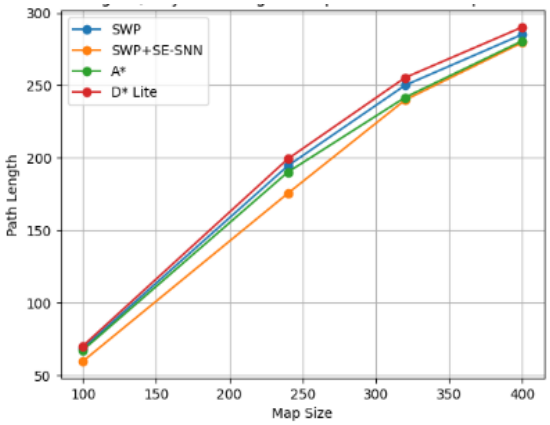

Figure 4: Single-query path length comparison across different map sizes (8-connected motion).

6.3 Multi-Query Performance Evaluation

In this configuration, the goal location remains fixed while multiple start positions are evaluated to simulate repeated navigation tasks toward a common target. Table 2: Performance evaluation in a multi-query planning scenario where multiple navigation queries are solved in the same environment. Here, all algorithms obtain the same mean path length (250.12), meaning that the optimality of paths is maintained over queries. Nevertheless, the proposed SWP+SE-SNN framework still far outperforms the

basic competitive algorithms in terms of computational efficiency. The average planning time per query is 0.1166 s, compared to 0.1665 s for SWP and 0.3212 s for A*. Meanwhile, the sum of offline and amortized planning time is reduced to 0.3258 s, further demonstrating the robustness of learning-based planning in repeated query cases. No crashes were detected for any method, indicating the robustness of the assessment.

Figure 6 demonstrates that SWP+SE-SNN significantly reduces the average planning time per query by replacing repeated search with amortized neural inference, while preserving path optimality.

Table 1: Performance comparison across different map sizes (8-connected motion).

Map Size	Method	Path Length	Planning Time (s)	Goal Reached
100×100	SWP	68.46	0.1902	1
	SWP+SE-SNN	59.57	0.0119	1
	A*	67.23	0.0314	1
	D* Lite	70.12	0.2023	1
240×240	SWP	194.47	1.1641	1
	SWP+SE-SNN	175.45	0.0500	1
	A*	190.12	0.0905	1
	D* Lite	199.25	1.1768	1
320×320	SWP	250.02	2.2526	1
	SWP+SE-SNN	239.87	0.1344	1
	A*	241.65	0.3343	1
	D* Lite	255.32	2.2615	1
400×400	SWP	285.12	3.5788	1
	SWP+SE-SNN	279.55	0.2015	1
	A*	280.65	0.3968	1
	D* Lite	290.15	3.2546	1

Table 2: Quantitative performance in multi-query planning.

Method	Mean Path Length	Mean Time / Query	Offline + Amortized Time	Failures
SWP	250.1	0.1665	0.41563	0
SWP+SE-SNN	250.1	0.1166	0.32575	0
A*	250.1	0.3211	0.53033	0
D* Lite	250.1	0.1565	0.41565	0

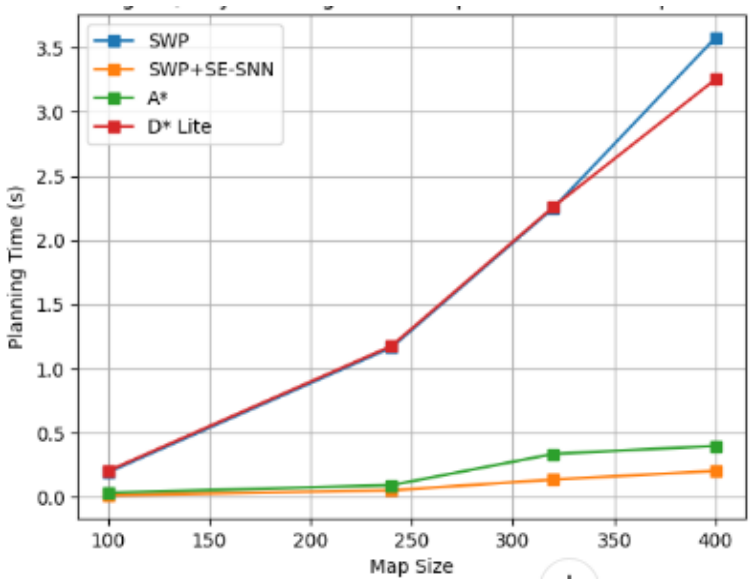

Figure 5: Single-query planning time comparison across different map sizes (8-connected motion).

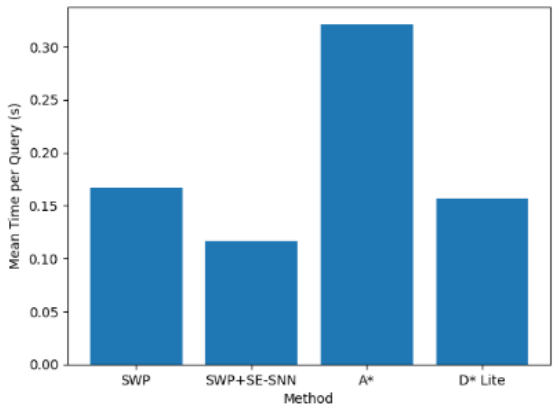

Figure 6: Mean planning time per query in a multi-query navigation scenario.

7 DISCUSSION

The gains in performance we have observed may be due to the combination of expert-based planning and spike-based neural learning. Although SWP is good at generating initial trajectories, it does not focus on local refinement. The SE-SNN trained with the e-prop learning rule learns spatiotemporal navigation patterns and removes redundant movements, resulting in shorter, smoother trajectories.

This is in contrast to traditional planners that rely on repeated searching of the graph space, and our learned SE-SNN generates navigation behaviors in one pass through a feedforward network, allowing for fast planning when optimizing its policy specifically in large-scale environments. The results also justify that the proposed framework not only accelerates path planning but also generates better trajectories than expert demonstrations.

8 CONCLUSIONS

This paper presents a learning-based navigation framework that integrates expert route planning with impulse neural networks. Expert route planning (SWP) was used to generate expert routes, which were then trained on a Super-Elman SNN using the e-prop learning framework. The Super-Elman SNN acted as an intelligent control layer within the SDN, not only making navigation decisions but also predicting anomalies and supporting fault management by adapting routing decisions locally without requiring a complete map redraw. The framework was evaluated in real-world environments derived from the KITTI Odometry dataset. Simulations were conducted in network environments with map sizes ranging from 100×100 to 400×400. Within this framework, the results showed that the proposed SWP+SE-SNN approach consistently achieved shorter route lengths and faster planning times than traditional schemes, including A* and D* Lite, across all tested map sizes. Performance gains increase with the size of the environment, demonstrating the robustness and scalability of the proposed method while maintaining reliable access to the destination. Finally, the results presented in this paper confirm that combining learning with classical planners is a viable approach to guarantee efficient, portable, and robust robotic navigation.

9 FUTURE WORK

Future research directions will focus on improving the adaptability, scalability, and practical applicability of the proposed framework. The following aspects should be investigated to further enhance system performance and facilitate real-world deployment:

1) Extension to dynamic environments with moving obstacles.
2) Integration of online learning for adaptive navigation.
3) Real-world validation on a physical robotic platform.
4) investigate hardware-level energy efficiency and power profiling of the proposed framework.

REFERENCES

[1] Y. Tang, M. A. Zakaria, and M. Younas, "Path Planning Trends for Autonomous Mobile Robot Navigation: A Review," Sensors, vol. 25, no. 4, p. 1206, 2025, [Online]. Available: https://doi.org/10.3390/s25041206.

[2] N. Nasti, S. Shoaib, Z. Najar, and M. A. Chishti, "A Comprehensive Review of Path Planning Techniques for Mobile Robot Navigation in Known and Unknown Environments," International Journal of Computational and Experimental Science and Engineering, vol. 11, 2024, doi: 10.22399/ijcesen.797.

[3] Y. Zhu, W. Z. Wan Hasan, H. R. Harun Ramli, N. M. H. Norsahperi, M. S. Mohd Kassim, and Y. Yao, "Deep Reinforcement Learning of Mobile Robot Navigation in Dynamic Environment: A Review," Sensors, vol. 25, no. 11, p. 3394, 2025.

[4] H. Espino, R. Bain, and J. L. Krichmar, "A Rapid Adapting and Continual Learning Spiking Neural Network Path Planning Algorithm for Mobile Robots," 2024, [Online]. Available: https://doi.org/10.48550/arXiv.2404.15524.

[5] J. Wu, Y. Wang, Z. Li, L. Lu, and Q. Li, "A Review of Computing with Spiking Neural Networks," Computers, Materials & Continua, vol. 78, pp. 2909-2939, 2024, doi: 10.32604/cmc.2024.047240.

[6] B. A. Abubaker et al., "Spiking neural network for enhanced mobile robots' navigation control," in 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS), IEEE, 2023.

[7] B. Ayasi, C. J. Carmona, M. Saleh, and A. M. García-Vico, "A Practical Tutorial on Spiking Neural Networks: Comprehensive Review, Models, Experiments, Software Tools, and Implementation Guidelines," Eng, vol. 6, p. 304, 2025, [Online]. Available: https://doi.org/10.3390/eng6110304.

[8] G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "Biologically inspired alternatives to backpropagation through time for learning in recurrent neural networks," Nature Communications, vol. 11, 2020.

[9] B. Lin, "Toward an AI-Enabled SDN-based 5G & IoT Network," Network and Communication Technologies, vol. 5, p. 7, 2020, doi: 10.5539/nct.v5n2p7.

[10] M. Rahouti, K. Xiong, and Y. Xin, "Secure Software-Defined Networking Communication Systems for Smart Cities: Current Status, Challenges, and Trends," IEEE Access, pp. 1-1, 2020, doi: 10.1109/ACCESS.2020.3047996.

[11] S. Patle, D. Pandey, A. Jagadeesh, and D. R. Parhi, "Path planning in uncertain environments: A comprehensive review," Defence Technology, vol. 16, no. 2, pp. 327-344, 2020.

[12] N. Rathi and K. Roy, "Lite-SNN: Leveraging inherent dynamics to train energy-efficient spiking neural networks for sequential learning," IEEE Transactions on Cognitive and Developmental Systems, vol. 16, no. 6, pp. 1905-1914, 2024.

[13] Y. Zhao, Y. Zhang, and S. Wang, "A Review of Mobile Robot Path Planning Based on Deep Reinforcement Learning Algorithm," Journal of Physics: Conference Series, vol. 2138, p. 012011, 2021, doi: 10.1088/1742-6596/2138/1/012011.

[14] A. Martin, L. Furieri, F. Dörfler, J. Lygeros, and G. Ferrari-Trecate, "Follow the Clairvoyant: an Imitation Learning Approach to Optimal Control," IFAC-PapersOnLine, vol. 56, no. 2, 2023.

[15] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Map-less Navigation by Leveraging Prior Demonstrations," IEEE Robotics and Automation Letters, pp. 1-1, 2018, doi: 10.1109/LRA.2018.2869644.

[16] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning Navigation Behaviors End-to-End with AutoRL," IEEE Robotics and Automation Letters, pp. 1-1, 2019, doi: 10.1109/LRA.2019.2899918.

[17] X. Jiang et al., "Fully spiking neural network for legged robots," in ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2025.

[18] W. Ponghiran and K. Roy, "Spiking neural networks with improved inherent recurrence dynamics for sequential learning," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 7, 2022.

[19] N. Aljamali and M. Abdullah, "Smart element aware gate controller for intelligent wheeled robot navigation," International Journal of Electrical and Computer Engineering (IJECE), vol. 11, no. 4, pp. 3022-3031, 2021, doi: 10.11591/ijece.v11i4.pp3022-3031.

[20] A. Perrett et al., "Online learning in SNNs with e-prop and Neuromorphic Hardware," in Proceedings of the 2022 Annual Neuro-Inspired Computational Elements Conference, 2022.

[21] A. Rostami et al., "E-prop on SpiNNaker 2: Exploring online learning in spiking RNNs on neuromorphic hardware," Frontiers in Neuroscience, vol. 16, p. 1018006, 2022.

[22] Z. E. Kanoon, A. S. Al-Araji, and M. N. Abdullah, "An Intelligent Path Planning Algorithm and Control Strategy Design for Multi-Mobile Robots based on a Modified Elman Recurrent Neural Network," International Journal of Intelligent Engineering and Systems, vol. 15, no. 5, pp. 400-415, 2022, [Online]. Available: https://doi.org/10.22266/ijies2022.1031.35.

[23] K. Katona, H. A. Neamah, and P. Korondi, "Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot," Sensors, vol. 24, no. 11, 2024, [Online]. Available: https://doi.org/10.3390/s24113573.

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI Vision Benchmark Suite," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3354-3361, 2012, doi: 10.1109/CVPR.2012.6248074.

[25] T. D. Barfoot, State Estimation for Robotics, 2nd ed. Cambridge, UK: Cambridge University Press, 2024.