Online and Continuous Neuromorphic Robotic Navigation

Harrison Espino[1] and Robert Bain[2] and Jeffrey L. Krichmar[1,2]

***Abstract*—Mapping an environment and planning paths based on this map are important tasks in autonomous navigation. We present a neuromorphic algorithm, called spiking wavefront propagation, that is capable of concurrently mapping large complex environments and adaptively planning paths depending on the context. The algorithm is tested on a mobile robot platform in two different environments with varying terrain. Results indicate that the algorithm is capable of discerning differences in terrain and obstacles using two measures of cost, (1) energy expenditure by the wheels and (2) time spent in the presence of obstacles. The spiking wavefront planner rapidly generates contextual maps suitable for a robot navigating varied environments. After learning an environment, the robot exhibits different behaviors depending on which cost it considers to plan paths. Using the current cost map, the robot may travel longer physical distances in favor of minimizing energy expenditure, whereas the obstacle cost map may take a more direct route while planning paths around potential obstacles. This work demonstrates a novel solution for adaptive planning without needing rounds of retraining and deploying.**

***Index Terms*—Navigation, Planning, Mapping, SLAM, Neuromorphic, Spiking Network**

I. Introduction

Finding one's way around is an important part of everyday life. Similarly, robots and other autonomous systems require this capability. Researchers and industry have made considerable progress developing navigation systems. However, critical open issues remain, such as: 1) Mapping and planning concurrently, 2) Continuous mapping and planning without the need for retraining with offline updates when conditions change, and 3) Flexibility in the face of uncertainty and change. In the proposed work, we address these issues by taking inspiration from biology and by developing a novel approach to taking different environmental costs into account when navigating.

Prior solutions include imitation learning and reinforcement learning for end to end navigation. Imitation learning methods do not allow continuous or autonomous learning, as they rely on multiple rounds of human annotated data for each new environment [1]–[3]. Reinforcement learning (RL) methods closely mimic humans learning from interaction with their environments, but require extensive offline training and expensive hardware to run in real-time [4]–[7].

This work was supported by the Air Force Office of Scientific Research (AFOSR) Contract No. FA9550-19-1- 0306, and by the National Science Foundation (IIS-RI award 1813785 and NSF-FO award IIS-2024633).

[1]Harrison Espino is with the Department of Computer Science, University of California, Irvine, Irvine, CA, USA espinoh@uci.edu

[2]Robert Bain is with the Department of Cognitive Sciences, University of California, Irvine, Irvine, CA, USA rkbain@uci.edu

[1,2]Jeffrey Krichmar is with the Department of Cognitive Sciences and Department of Computer Science, University of California, Irvine, Irvine, CA, USA jkrichma@uci.edu

We present a neuromorphic algorithm, called spiking wavefront propagation [8], [9], that simultaneously and continuously constructs environmental cost maps and uses those maps to plan efficient paths. The algorithm is tested on a ground robot in rugged, varied terrains with cost maps for obstacles and for the robot's effort based on the motor's current draw. We show that the robot learns to plan paths around impassable regions with an obstacle cost map, and to plan smoother paths with a cost map based on the robot's energy expenditure.

II. Methods

The robot is tasked with exploring the environment using the spiking wavefront planner to guide navigation. As the robot explores, it learns features of the environment through experience and uses this to update the planner. In this section, we cover details about the spiking wavefront planner, the cost maps, the robot platform used in the experiments, and the environments.

A. Spiking Wavefront Planner

Here we briefly describe the spiking wavefront planner model. For more details, see [8], [9].

The spiking wavefront propagation algorithm assumes a grid representation of space, where connections between units represent the ability to travel from one grid location to a neighboring location. Each unit in the grid is represented by a simplified integrate and fire neuron. The activity of neuron i at time $t+1$ is represented by (1):

$$v_i(t+1) = u_i(t) + I_i(t+1) \tag{1}$$

in which $u_i(t)$ is the recovery variable and $I_i(t)$ is the input current at time t.

The recovery variable $u_i(t+1)$ is described by:

$$u_i(t+1) = \begin{cases} -10 & if\ v_i(t) = 1 \\ min(u_i(t)+1, 0) & otherwise \end{cases} \tag{2}$$

such that immediately after a membrane potential spike, the recovery variable starts as a negative value and linearly increases toward a baseline value of 0.

The input current I at time $t+1$ is given by:

$$I_i(t+1) = \sum_{j=1}^{N} \begin{cases} 1 & if\ d_{ij}(t) = 1 \\ 0 & otherwise \end{cases} \tag{3}$$

such that $d_{ij}(t)$ postpones the integration of input, I, from neighboring neuron j to neuron i. This delay is given by:

$$d_{ij}(t+1) = \begin{cases} D_{ij} & if\ v_j(t) \geq 1 \\ max(d_{ij}(t)-1, 0) & otherwise \end{cases} \tag{4}$$

The value of $D_{ij}(t)$ is the propagation delay between neurons i and j, and denotes the expected cost of traveling from location i to j. Cost is an open parameter, which could depend on a number of variables. In the present paper, cost represents how easy it is to move in a given direction. A low cost could be an open traversable region of space, and a high cost could signify the presence of a barrier.

B. E-Prop

In the present path planning algorithm, E-Prop [10] was used to learn a map of the environment and to plan paths based on that map knowledge. The E-Prop learning rule was developed to learn sequences in spiking neural networks by using an eligibility trace to implement backpropagation through time to minimize a loss function. For path planning purposes, the active neurons after a wave propagation are eligible for updates. An eligibility trace based on time elapsed since the wave reaches the goal destination, dictates the eligibility.

E-Prop is applied to weights projecting from neurons along the calculated path. It is assumed that the robot can sense the features (e.g., traversal cost) at map locations along the path taken. In this way, E-Prop solves the credit assignment problem by rewarding paths that lead to goal locations, while also learning about environmental structure.

The E-Prop algorithm is applied to weights of the spiking neural network by:

$$D_{ij}(t+1) = D_{ij}(t) + \delta(e_i(t)(map_{xy} - D_{ij}(t)) \quad (5)$$

where δ is the learning rate, set to 0.5, $e_i(t)$ is the eligibility trace for neuron i, and map_{xy} represents the observed cost for traversing the location (x, y), which corresponds to neuron i. This rule is applied for each of the neighboring neurons, j, of neuron i. The loss in Eqn. 5 is map_{xy} - D_{ij}.

The eligibility trace for neuron i is given by (6):

$$e_i(t+1) = \begin{cases} 1 & if\ v_j(t) \geq 1 \\ e_i(t) - \frac{e_i(t)}{\tau} & otherwise \end{cases} \quad (6)$$

where τ is the rate of decay for the eligibility trace, set to 25.

To determine a path from the robot's current location to a destination, a signal is sent originating from the neuron corresponding to the robot's current location. This signal propagates to the origin neuron's neighbors, delayed by the associated weight of the origin's outgoing connection. This is repeated for each of these neurons and their neighbors until a signal reaches the neuron corresponding to the destination for the first time. The origin of this signal is recursively traced backwards until the first neuron is reached again. This sequence of neurons represents the path of least cost given the robot's information about the environment based on its learned cost map. Further details on the original spiking wavefront propagation planner can be found in [8].

C. Robot Platform

The robot platform used in the experiments was the Jackal Unmanned Ground Vehicle (UGV) from Clearpath Robotics (see Fig. 1). The Jackal is capable of navigating through difficult or uneven terrain such as tall grass or hills. In order to protect against forwards or backwards tipping, we added additional bumpers extending from the front and back of the robot. To localize the robot in its environment and determine the distance to waypoints we used a NovaTel GPS unit, which has a precision of approximately 1.2 meters. To determine the robot's heading and bearing, we used a Lord Microstrain 3DMGX5 inertial measurement unit.

Fig. 1. The Clearpath Jackal robot platform and our modifications.

With obstacle avoidance enabled, the presence of obstacles in the robot's path would increase the obstacle cost while leaving the current cost unchanged. When obstacle avoidance was disabled the robot would attempt to move through the obstacles in its path. In the case that the obstacle was traversable, such as tall grass, the obstacle cost would increase and the current cost would remain unchanged. If the obstacle was not traversable, the impact and continual attempt to move while stuck would put strain on the wheels, causing both the obstacle cost and the current cost to increase.

D. Cost Maps

We consider two measures of cost. The first measure uses the amount of current flowing from the battery to the left and right wheels. This is determined internally using the status messages automatically published by the Jackal. For each incoming current reading, we take the minimum value between the left and right side. This is to prevent spikes in current caused by turning from influencing the cost. The cost is calculated as the average of these current readings by Eqn. 7.

$$map_{xy} = (c_{avg} * 10)^2 \quad (7)$$

Where c_{avg} is the average current draw by the wheels (minimum of left or right at each time step) over the course of navigating to the waypoint. The average is scaled by 10 to obtain values greater than one, and squared to greater discern between small differences.

The second measure is based on the presence of obstacles during traversal. Obstacles are detected using the LiDAR as described previously. To recognize obstacles we used the SICK LMS111 2D LiDAR, which has an aperture angle of 270 degrees and an angular resolution of 0.5 degrees. Objects were considered obstacles if the LiDAR detected a number of

consecutive data points below a threshold of 0.9 meters. The cost is calculated as the fraction of time spent with obstacles in the field of view by Eqn. 8.

$$map_{xy} = (\frac{t_{obs}}{t_{total}} * 4) + 1 \tag{8}$$

Where t_{obs} is the time spent with obstacles in the presence of the LiDAR and t_{total} is the total time spent navigating. This fraction is then rescaled to be between 1 and 5 so it can be used to train the model weights, which need to be integer representing delays.

E. Navigation Trials

A single trial with the robot proceeded as follows. First, an end point was randomly determined using a Levy Flight distribution. This distribution is commonly used to model foraging patterns in animals [11]. The Levy Flight distribution will tend to focus search in a local area and then occasionally jump to a distant area. This was a better exploration strategy than a more random search pattern, such as Brownian motion. Next, one of the cost types was randomly selected. The spiking wavefront planner planned a path from the robot's current location to the end point using the weights associated with the chosen cost type. The eligibility trace generated by the spiking wavefront planner was used to update weights as described in the Methods. The robot navigated through waypoints determined by the path generated from the starting to ending location.

To reach a waypoint, the robot oriented to the direction of the waypoint by rotating until the robot's heading (orientation with respect to a global reference frame) matched the desired bearing (orientation with respect to the waypoint). When the heading was suitably close to the bearing, the robot would proceed towards the waypoint. Periodically, the robot would recalculate its heading and adjust its trajectory.

Upon reaching a waypoint, both costs were calculated and stored for the weight update step. Once all waypoints are reached and the robot has reached its final destination, E-Prop was used with the saved eligibility trace and observed costs of traversal in order to update the weights of the spiking wavefront planner.

After each trial, the weights and a log of the GPS data are stored. The saved weights are used in simulated experiments to analyze possible paths taken by the robot. We test the spiking wavefront planner in two environments: 1) Aldrich Park: A hilly park with plenty of foliage. 2) University hills: A rugged region with scrub brush and dirt trails.

1) Aldrich Park: Aldrich Park is located at the center of the University of California, Irvine. A 7x7 grid, 4.7 meters apart were used as waypoints (see Fig. 2). Terrain in the environment was hilly and varied between thick grass, paved road, and dirt road. Notably, due to runoff from rainwater, there were a number of crevasses along the dirt road causing uneven terrain. Images of the environment are shown in Fig. 2. The environment contains a number of trees which serve as obstacles for the robot. Foot traffic along the road or occasionally in the grass would be detected as obstacles as well. All obstacles in the Aldrich environment could not be safely moved through so obstacle avoidance was enabled and the robot would move in the direction opposite the obstacle. In the case that the robot would be unable to avoid an obstacle or person, manual intervention using the controller was used to stop or maneuver the robot. The robot explored this environment for 100 trials over 3 days.

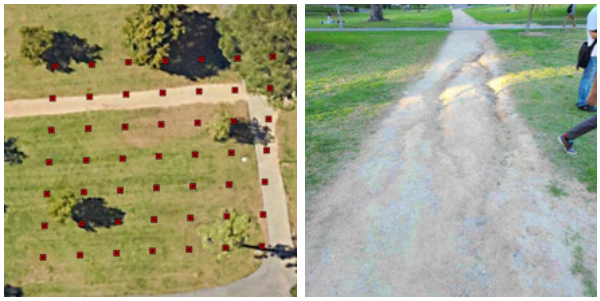

Fig. 2. Images of the Aldrich Park environment. Left image shows a top down satellite image of the environment and the location of waypoints. Right image shows the uneven terrain as a result of runoff from rain. Imagery©2022 Google.

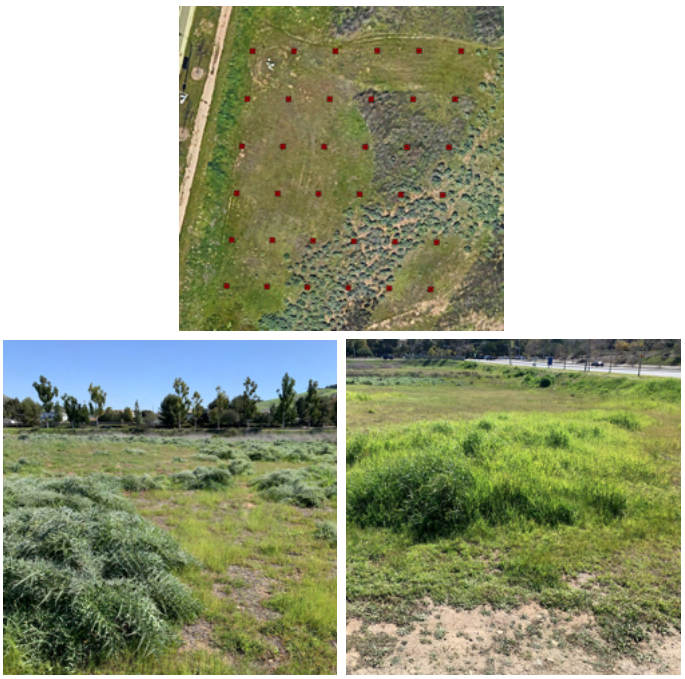

Fig. 3. Images of the University Hills environment. Top image shows a top down satellite image of the environment and the location of waypoints. Bottom left image shows an example of impassable scrub brush. Bottom right image shows traversable tall grass. Imagery©2022 Google.

2) University Hills: University Hills is in an off-road environment located near the Irvine campus. Waypoints for this environment spanned a 6x6 grid, with a distance of 8.3 meters between waypoints. The terrain was primarily scrub brush, dirt, hills and a dirt road along the North side of the region. The environment is shown in Fig. 3. Obstacles consisted of scrub brushes mainly localized in one section of the environment.

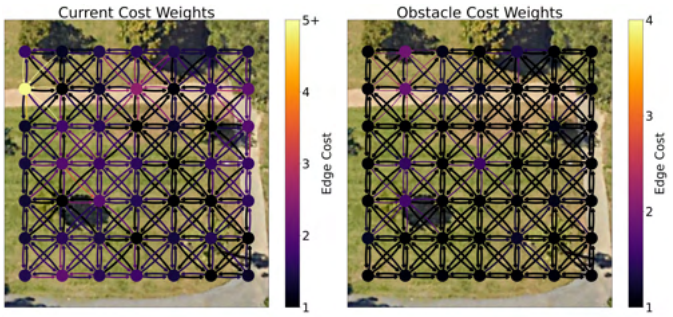

Fig. 4. Visualization of weights between neurons in the spiking wavefront model after 100 trials in the Aldrich Park environment using current (Left) and obstacles (Right) as the cost. Nodes are colored according to the average cost of incoming weights. Weights are colored and scaled according to cost, where smaller arrows represent higher cost weights.

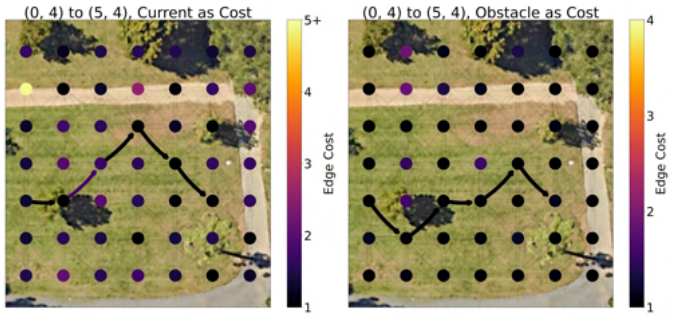

Fig. 5. Example paths in the Aldrich Park Environment using the learned spiking wavefront planner. Left image shows the path using current as the cost. Right image shows the path using obstacles as the cost.

Small patches of tall grass were also detected as obstacles, but were able to be traversed by the robot. This environment was explored for 50 trials over a single day. In order to allow for information to be gained by passing through some obstacles, obstacle avoidance was disabled. If the robot became stuck, it was moved manually using the controller.

III. Results

For each environment, the robot learned cost maps while planning paths using the spiking wavefront algorithm. Destinations were chosen using the Levy-Flight distribution [11] as described in the Methods. After exploration, we visualized the weights by overlaying the learned weight values of neurons on top of their corresponding waypoint locations. We analyzed how the learned weights reflected characteristics of the environment, such as differences in terrain or the presence of obstacles in different areas.

To better understand how the learned cost maps affect robot navigation, we exhaustively planned paths using all possible start and end locations of a length of 4 waypoints or farther by simulating those paths with the spiking wavefront planner. The total cost and path length of these paths were compared to a naive planner, which has the same weights for all connections between waypoints resulting in a path of the shortest Euclidean distance. P-values were calculated using the Kolmogorov-Smirnov goodness-of-fit hypothesis test to compare the naive cost maps with the learned one.

A. Aldrich Park

After several trials of robot exploration, as was described in the Methods, learned neural network weights reflected environmental features based on different cost. The cost map weights from the robot's exploration of Aldrich park are shown in Fig. 4. Using the current cost criteria, we found similar weight values between the grass and roads, indicating they are both similarly traversable. However, the uneven dirt road produced high current leading to large weights. Current cost weight values were higher for incoming weights than outgoing weights at waypoints which correspond to the crevasses. This suggests that the robot struggled more when approaching the crevasses than climbing out of them. The obstacle cost produced a sparser map, with high costs mostly in waypoints near trees. Due to the unevenness of the road, the robot was also at an angle steep enough to briefly detect the ground as an obstacle as well, causing a higher obstacle cost near the crevasses. Although this cost was not as high for obstacles as it was for current.

The learned maps had an effect on robot navigation strategies. In Fig. 5, we show how the behavior differs based on the different learned cost maps. The robot is tasked with planning a path from (0, 4) to (5, 4). When using the current cost map to plan paths, the robot disregarded the presence of the tree and instead took the path of least energy expenditure. In contrast, when planning a path with the obstacle cost map, the robot traveled straight except to avoid the tree. From these behaviors, we see the robot prioritizing different objectives depending on the chosen cost map.

To better understand how the different cost maps would affect robot navigation, we simulated all starting and ending locations that were more than 4 waypoints apart. The overall cost during navigation and the path lengths from these simulations of the Aldrich Park environment are shown in Table I. We found that the trained spiking wavefront planner planned produced paths with significantly lower current and obstacle costs than a naive planner. Paths planned using current and obstacle costs were only slightly longer than the shortest distance paths.

Taken together, these robot experiments and simulations show that the robot can rapidly learn environmental costs and these costs can lead to context-dependent navigation strategies. Aldrich Park is a fairly homogenous environment. To further test this idea, we explored University Hills, which is a more rugged environment using the same protocols.

B. University Hills

University Hill near the UC Irvine campus has more obstacle and difficult terrain than Aldrich Park, and thus may provide a more stringent test of the present navigation algorithm. After 50 trials of exploration, the cost map weights for University Hills environment showed more environmental features and more dramatic differences between the two types

TABLE I
ALDRICH PARK - COMPARISON BETWEEN LEARNED PLANNER AND NAIVE PLANNER

Percentile	25th	50th	75th	p <
Current Cost	5.0654	5.9006	6.8691	
Naive-Current Cost	5.5796	6.6699	8.0526	9.8972e-23
Obstacle Cost	4.0	5.0	6.0	1.0148e-171
Naive-Obstacle Cost	8.0	9.0	12.0	
Current Length	5.0	6.0	6.0	
Naive-Current Length	5.0	6.0	6.0	0.00013626
Obstacle Length	5.0	6.0	6.0	
Naive-Obstacle Length	5.0	6.0	6.0	0.12287

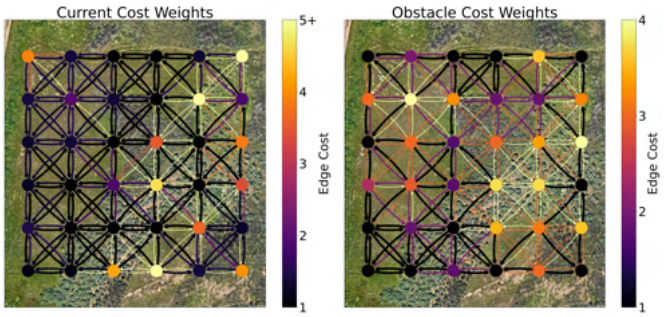

Fig. 6. Visualization of weights between neurons in the spiking wavefront model after 50 trials in the University Hills environment using current (Left) and obstacle (Right) as the cost. Nodes are colored according to the average cost of incoming weights. Weights are colored and scaled according to cost, where smaller arrows represent higher cost weights.

of cost maps than Aldrich Park (see Fig. 6). High current cost shown through the weight values were observed in the Southeast portion of the environment. This reflects the high density of scrub brush in that area, which the robot struggled to traverse over and often got stuck on. This is also represented as a high obstacle cost, as the scrub brush was high enough to be detected on the LiDAR as an obstacle. Additionally, tall grass found in the northwestern portion of the environment was able to be easily traversed, but would be detected as an obstacle by the LiDAR. This resulted in high obstacle cost weights but low current cost weights in those areas.

We show how the differences in the paths taken depend strongly on the learned cost maps through the example paths in Fig. 7. When planning a path using the current map, the robot took a shorter path through thick vegetation, including through patches of tall grass. Planning a path using the obstacle map led to a longer path that avoided as much vegetation as possible. In general, the robot was more careful to avoid possible collisions using the obstacle cost weights at the expense of traveling a longer physical distance and requiring more energy.

As was done for the Aldrich Park neural networks, we simulated all starting and ending locations that were more than 4 waypoints apart in University Hills (see Table II). Similar to Aldrich Park , paths taken by the trained planners had significantly lower costs than a naive planner. Path lengths by the trained planner were also only slightly longer than the shortest path.

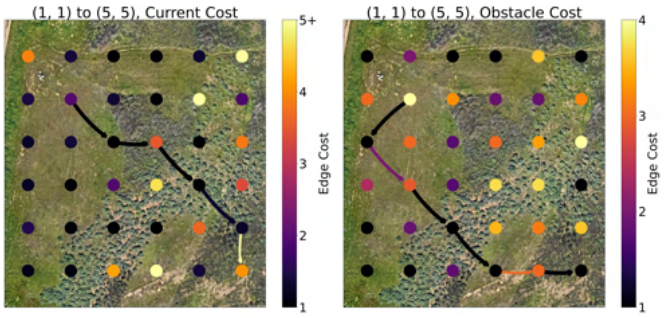

Fig. 7. Example paths in the University Hills environment using the learned spiking wavefront planner. Left image shows the path using current as the cost. Right image shows the path using obstacles as the cost.

TABLE II
UNIVERSITY HILLS - COMPARISON BETWEEN LEARNED PLANNER AND NAIVE PLANNER

Percentile	25th	50th	75th	p <
Current Cost	4.67	5.6921	9.064	
Naive-Current Cost	5.715	11.2681	49.4592	2.428e-21
Obstacle Cost	6.2869	7.3596	8.6272	
Naive-Obstacle Cost	6.7998	8.2976	9.9473	6.776e-09
Current Length	5.0	5.0	6.0	
Naive-Current Length	5.0	5.0	6.0	0.11168
Obstacle Length	5.0	5.0	6.0	
Naive-Obstacle Length	5.0	5.0	6.0	0.00031267

Taken together, these results further show how neural networks trained with different cost criteria can lead to different behavioral outcomes, as well as reflecting environmental features. It should also be stated that only a few training trials were necessary to generate these maps.

IV. DISCUSSION

The spiking wavefront planner rapidly generated contextual maps suitable for a robot navigating varied environments. Through varied measures of cost, the model was able to differentiate between traversable and untraversable regions through experience and exhibit different behaviors depending on which cost it considers to plan paths. Compared to a naive planner, a learned planner will choose to travel longer physical distances in favor of minimizing energy expenditure or obstacle encounters. In two different environments, these maps were created with few trials of simultaneous path planning and mapping. Furthermore, the weight matrices provide explainable maps reflecting environmental features.

We take inspiration from animal behavior and neuroscience to develop a navigation system capable of operating over large areas for extended time periods. Specifically, we model animal foraging patterns for exploration, and hippocampal forward replay of prior experiences, and deciding between potential paths by weighing their costs [12]. The wavefront propagation algorithm for spiking neural networks used here has comparable performance to other path planning algorithms but is more efficient [8]. In general, it is lightweight and learns rapidly. Furthermore, since it is a spiking neural network it can

be deployed on power efficient neuromorphic hardware as was demonstrated by [13].

The dominant approach for autonomously navigating mobile robots is to have it build a map, localize itself in the map, and use that map in order to plan and execute actions towards its goal. This technique is known as simultaneous localization and mapping (SLAM) and planning. One major limitation of SLAM approaches is that they do not continually learn from interactions with the environment like humans do. Learning-based methods have shown promise in addressing these limitations by learning from data. Imitation learning solutions have been developed, but they do not support continuous or autonomous learning from interactions with the environment [1]–[3]. Several reinforcement learning (RL) methods have also been proposed to more closely mimic this aspect of human behavior [4]–[7]. One of these methods, BADGR, required extensive offline training [4]. It took 42 hours and 720k policy data points for it to learn 2 environments. By comparison, our robot took roughly 7 hours and less than 10k data points to learn 2 different environments. This discrepancy in sample efficiency would only grow over deployment time, as new learning is required every time after an environment changes. ViKiNG is another offline RL approach to autonomously control a Clearpath Jackal [7]. ViKiNG traverses impressive distances by incorporating overhead satellite imagery as input to its deep neural network (DNN). [5] uses a similar DNN architecture to BADGR to classify terrain as "smooth" or "rough". Like ViKiNG, they too add aerial images, but it comes from drones instead of satellite data. Our design implicitly does the same type of classification over terrain when path planning as [5] and [6]. A central drawback of these RL approaches is the need for real-time computing to control the vehicle. Because of this, training is done offline in a discontinuous fashion. Our method is efficient enough not only to run in real-time, but can continuously learn several independent cost maps online.

Our design paradigm of using multiple cost maps represents contextually dependent cost values, which allows it to support a wide range of navigation strategies dependent on need. Others have successfully used parallel cost maps to autonomously navigate too. For example, [14] won the 2007 DARPA Urban Challenge using multiple cost maps that acted independently, but these were combined in a heuristic search fashion. [15] incorporated multiple costmaps to effectively navigate in people-dense environments. Each map separated different data into semantically-separated layers. Each layer tracked one type of obstacle or constraint, and then modified a master cost map used for path planning.

Which cost map we use for planning dictates our robot's current behavior. It has been shown that the neuromodulator serotonin regulates impulsiveness and patience in animals. [16] simulated serotonin levels and showed the effects on route planning behaviors in GPS-compromised environments just like ours. They used a DQN network and trained offline though. Our learning rule is reminiscent of the delta learning rule of temporal difference learning models, the spiritual ancestor of most modern RL techniques (DQN included), in that updating of weights only occurs when prediction errors about future events occur. Future work is planned to add similar neuromodulation to more smoothly combine and weight our cost maps to effectively and naturally modulate behavior.

In summary, this work demonstrates our efficient algorithm for off-road navigation that learns continuously from interacting with its environment in real-time using CPU alone, without the need for multiple rounds of training and deployment, or expensive hardware needed by previous state of the art solutions. The simplicity of the software stack supports much future development, perhaps including using context from overhead satellite images like ViKiNG [7], integrating IMU data into our cost maps, variable throttle and turning velocities, memory replay, more robust obstacle detection and avoidance like BADGR [4], and neuromodulation to control combining cost maps.

REFERENCES

[1] F. Codevilla, M. Müller, A. Dosovitskiy, A. M. López, and V. Koltun, "End-to-end driving via conditional imitation learning," *CoRR*, vol. abs/1710.02410, 2017.

[2] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[3] S. Ross, N. Melik-Barkhudarov, K. Shankar, A. Wendel, D. Dey, J. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," *Proceedings - IEEE International Conference on Robotics and Automation*, 11 2012.

[4] G. Kahn, P. Abbeel, and S. Levine, "BADGR: an autonomous self-supervised learning-based navigation system," *CoRR*, vol. abs/2002.05700, 2020.

[5] T. Manderson, S. Wapnick, D. Meger, and G. Dudek, "Learning to drive off road on smooth terrain in unstructured environments using an on-board camera and sparse aerial images," 2020.

[6] K. Zhang, F. Niroui, M. Ficocelli, and G. Nejat, "Robot navigation of environments with unknown rough terrain using deep reinforcement learning," *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–7, 2018.

[7] D. Shah and S. Levine, "ViKiNG: Vision-based kilometer-scale navigation with geographic hints," Robotics: Science and Systems Foundation, jun 2022.

[8] T. Hwu, A. Y. Wang, N. Oros, and J. L. Krichmar, "Adaptive robot path planning using a spiking neuron algorithm with axonal delays," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 2, pp. 126–137, 2018.

[9] J. L. Krichmar, N. A. Ketz, P. K. Pilly, and A. Soltoggio, "Flexible path planning through vicarious trial and error," *bioRxiv*, 2021.

[10] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature Communications*, July 2020.

[11] X.-S. Yang, "Random walks and optimization," in *Nature-Inspired Optimization Algorithms*, ch. 3, pp. 45–65, Academic Press, 2014.

[12] A. D. Redish, "Vicarious trial and error," *Nature Review Neuroscience*, vol. 17, no. 3, pp. 147–59, 2016.

[13] K. D. Fischl, K. Fair, W.-Y. Tsai, J. Sampson, and A. Andreou, "Path planning on the truenorth neurosynaptic system," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, 2017.

[14] D. Ferguson and M. Likhachev, "Efficiently using cost maps for planning complex maneuvers," 2008.

[15] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 709–715, 2014.

[16] J. Xing, X. Zou, and J. L. Krichmar, "Neuromodulated patience for robot and self-driving vehicle navigation," *CoRR*, vol. abs/1909.06533, 2019.