Adaptive Robot Path Planning Using a Spiking Neuron Algorithm With Axonal Delays

Tiffany Hwu, Alexander Y. Wang, Nicolas Oros, and Jeffrey L. Krichmar

***Abstract*—A path planning algorithm for outdoor robots, which is based on neuronal spike timing, is introduced. The algorithm is inspired by recent experimental evidence for experience-dependent plasticity of axonal conductance. Based on this evidence, we developed a novel learning rule that altered axonal delays corresponding to cost traversals and demonstrated its effectiveness on real-world environmental maps. We implemented the spiking neuron path planning algorithm on an autonomous robot that can adjust its routes depending on the context of the environment. The robot demonstrates the ability to plan different trajectories that exploit smooth roads when energy conservation is advantageous, or plan the shortest path across a grass field when reducing distance traveled is beneficial. Because the algorithm is suitable for spike-based neuromorphic hardware, it has the potential of realizing orders of magnitude gains in power efficiency and computational gains through parallelization.**

***Index Terms*—Neuromorphic chips, path planning, plasticity, robotics, spiking neurons.**

I. Introduction

Navigation in biology requires acquiring a map, and then using that map to make intelligent decisions on where to go and what to do [1], [2]. The idea of a cognitive map, which was proposed by Tolman [3] in the last century, is where the animal takes costs, context, and its needs into consideration when moving through its environment.

Path planning involves calculating an efficient route from a starting location to a goal, while avoiding obstacles and other impediments. Despite much advancement over several decades of robotic research, there are still many open issues for path planners [4]. Classic path planning algorithms include Dijkstra's algorithm, A Star (A*), and D*. Dijkstra's algorithm uses a cost function from the starting point to the desired goal. A* additionally considers the distance from the start to the goal "as the crow flies" [5]. D* extends the A* algorithm by starting from the goal and working toward the start positions. It has the ability to readjust costs, allowing it to replan paths in the face of obstacles [6]. These algorithms can be computationally expensive when the search space is large. Rapidly exploring random trees are a less expensive approach because they can quickly explore a search space with an iterative function [7]. Still these path planners are computationally expensive and may not be appropriate for autonomous robots and other small, mobile, embedded systems.

Because of their event driven design and parallel architecture, neuromorphic hardware holds the promise of decreasing size, lowering weight and reducing power consumption, and may be ideal for embedded applications [8]. These systems are modeled after the brain's architecture and typically use spiking neural elements for computation [9]. Spiking neurons are event driven and typically use an address event representation (AER), which holds the neuron ID and the spike time, for communicating between neurons. Since spiking neurons do not fire often and post-synaptic neurons do not need to calculate information between receiving spikes, neuromorphic architectures allow for efficient computation and communication.

Path planning approaches that may be a good fit for neuromorphic applications are wavefront planners [10], [11] and diffusion algorithms [12]. In a standard wavefront planner, the algorithm starts by assigning a small number value to the goal location. In the next step, the adjacent vertices (in a topological map) or the adjacent cells (in a grid map) are assigned the goal value plus one. The "wave" propagates by incrementing the values of subsequent adjacent map locations until the starting point is reached. Typically, the wave cannot propagate through obstacles. A near-optimal path, in terms of distance and cost of traversal, can be read out by following the lowest values from the starting location to the goal location.

We recently introduced a spiking neuron wavefront algorithm for path planning that adapts to changes in the environment [13]. The adaptive element is inspired by recent empirical findings supporting experience dependent plasticity of axonal conduction velocities [13]. Unlike prior implementations of spiking wavefront path planners, our algorithm introduces an adjustable spike delay that could potentially allow for dynamic online adaptation to realistic environmental costs, while maintaining a temporally sparse coding of

Manuscript received August 22, 2016; revised November 12, 2016; accepted December 21, 2016. Date of publication January 19, 2017; date of current version June 8, 2018. This work was supported in part by the National Science Foundation under Award 1302125, in part by the Northrop Grumman Aerospace Systems, in part by the Telluride Neuromorphic Cognition Engineering Workshop, Institute of Neuromorphic Engineering, in part by the National Science Foundation, in part by DoD, and in part by Industrial Sponsors.

T. Hwu is with the Department of Cognitive Sciences, University of California at Irvine, Irvine, CA 92697 USA, and also with Northrop Grumman, Redondo Beach, CA 90278 USA.

A. Y. Wang is with the Department of Mechanical and Aerospace Engineering, University of California at Irvine, Irvine, CA 92697 USA.

N. Oros is with BrainChip Inc., Aliso Viejo, CA 92656 USA.

J. L. Krichmar is with the Department of Cognitive Sciences, University of California at Irvine, Irvine, CA 92697 USA, and also with the Department of Computer Science, University of California at Irvine, Irvine, CA 92697 USA (e-mail: jkrichma@uci.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCDS.2017.2655539

the path. In simulations, we were able to show that this algorithm was more computationally efficient and sensitive to cost than existing path planning algorithms.

Versions of the wavefront algorithm have been implemented on neuromorphic hardware that supports spiking neurons [14]–[16], including our own algorithm which was successfully implemented on the IBM TrueNorth chip [17]. In addition, it has been shown to plan efficient paths on mobile robots and robotic arms [11], [14], [18]. However, these algorithms are not flexible in dynamic environments or in situations where the context changes. Other neural inspired implementations of the wavefront or diffusion idea use learning rules to learn routes through environments [19]–[22]. These models rely on synaptic plasticity to learn and adapt to environments. However, most of the environments used in these experiments have been static and highly constrained.

In this paper, we expand upon this previous work, demonstrating the algorithm's effectiveness in complex natural environments. Rather than using a manually constructed map, we create maps from an outdoor park with an abundance of natural obstacles and varied terrain. These obstacles and varied terrain are reflected in the algorithm's cost function. We further show how this algorithm can be implemented on an autonomous robot navigating an outdoor environment. The mobile robot had to consider real-world costs and tradeoffs in the environment, such as smooth roads versus rough grass, as well as obstacles such as benches, bushes, and trees. The robot demonstrated context-dependent path planning through this environment and the spiking wavefront was efficient enough to run on an Android smartphone that was mounted on and controlled the robot.

II. Methods

A. Neuron Model and Connectivity

To demonstrate a spiking neuronal wave path planning algorithm, we constructed a simple spiking neuron. The neuron model contained a membrane potential (v), a recovery variable (u), and received current input (I) from synaptically connected neurons

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

$$u_i(t+1) = \min(u_i(t)+1, 0) \tag{2}$$

$$I_i(t+1) = \sum\nolimits_j \big(1 \text{ if } d_{ij}(t) = 1;\ 0\, \text{otherwise}\big) \tag{3}$$

$$d_{ij}(t+1) = \max\big(d_{ij}(t) - 1, 0\big). \tag{4}$$

$d_{ij}(t)$ is the axonal delay between when neuron $v_j(t)$ fires an action potential and neuron $v_i(t)$ receives the action potential. When $v_j(t)$ fires an action potential, $d_{ij}(t)$ is set to a delay value of $D_{i,j}(t)$, which was assigned according to variable costs in the environment. Note from (4) that d_{ij} has a null value of zero unless the presynaptic neuron fires an action potential.

Equations (1)–(4) calculate the membrane potential, recovery variable, synaptic input, and axonal delay for neuron i at time step t, which is connected to j presynaptic neurons. The neuron spiked when v in (1) was greater than zero, in which case, v was set to 1 to simulate a spike, u was set to -5 to simulate a refractory period, and the axonal delay buffer, d, was set to D. The recovery variable, u, changed each time step per (2). The delay buffer, d, changed each time step per (4). I in (3) was the summation of the j presynaptic neurons that delivered a spike to post-synaptic neuron i at time t. Because of the refractory and delay periods, most neurons will be inactive during each timestep, resulting in sparse activity. Although neurons have to check if they fired a spike or received a spike each timestep, most of the computation occurs when neurons emit or receive a spike.

The neural network consisted of a 20×20 grid of spiking neurons as described in (1)–(4). The 20×20 grid corresponded to grid locations used in the outdoor robot experiments. Each neuron corresponded to a location in the environment and was connected to its eight neighbors (i.e., *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, and *NW*). At initialization ($t = 0$), v and u were set to 0. All delays, D, were initially set to 5, but could vary depending on experience in the environment. D represented the time it takes to propagate a presynaptic spike to its post-synaptic target.

B. Axonal Delays and Plasticity

A spike wavefront proceeds by triggering a single spike at a neuron that corresponds to the start location. This neuron then sends a spike to its synaptically connected neighbors. The delivery of the spike to its post-synaptic targets depends on its current axonal delay. Each synapse has a delay buffer, which governs the speed of the spike wave

$$D_{i,j}(t+1) = D_{i,j}(t) + \delta\big(\text{map}_{x,y} - D_{i,j}(t)\big) \tag{5}$$

where the delay $D_{i,j}(t)$ represents the axonal delay at time t between neurons i and j, $\text{map}_{x,y}$ is the value of the environment at location (x, y), and δ is the learning rate. For the present experiments, δ was set to 1.0, which allows the system to instantaneously learn the values of locations. This allows us to use an *a priori* cost map to test the effectiveness of the planning algorithm when the map is known, without incremental learning. In Section IV, we describe how changing the learning rate can allow for map creation as the robot explores an environment. The learning is expressed through axonal delays. For example, if the spike wave agent encountered a major obstacle, with a high traversal cost (e.g., 9), the neuron at that location would schedule its spike to be delivered to its connected neurons nine time steps later, whereas, if the traversal cost of a location were 1, the spike would be delivered on the next time step. It should be noted that in this paper the delay buffers were reset before each route traversal.

C. Spike Wave Propagation and Path Readout

Fig. 1 shows the progression of a spike wave in a typical environment. The example shows how the algorithm is sensitive to different costs; the left columns of Fig. 1 show an environment where the best path is the most direct route, and the right columns show an environment where it is advantageous to take a longer route along a smooth road. The neuron at the starting location is triggered to emit a spike. The top panels of Fig. 1 show the start of the spike wave emanating from the start position. Note how the spike wave

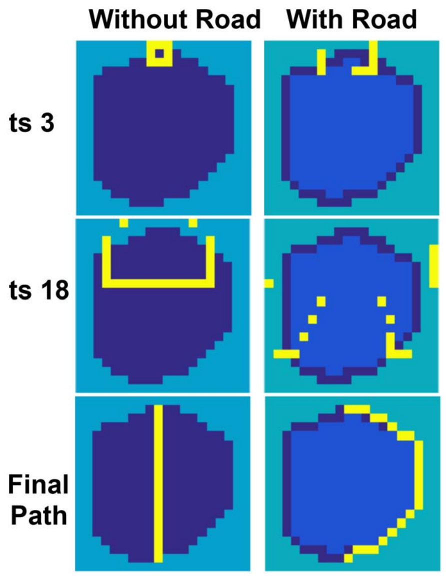


Fig. 1. Spike wave propagation in simulation. The top panels show the network activity at timestep 3, the middle panels show network activity at timestep 18, and the bottom panels show the resulting path. The left side shows the test area without a road. The light blue is the surrounding region, which has a cost of 9. The dark blue depicts the location of an open grass field, which has a cost of 3. The right side shows the test area that takes into consideration a road, which has a cost of 1 and is shown in dark blue. The spike wave propagation is shown in yellow. The starting location is at the top middle of the map, and the goal location is at the bottom middle of the map. Note how the spike wave propagates faster when there is a low cost road present.

is propagating faster on the regions that depict a smooth road [Fig. 1 (right column)]. This is because the road has a traversal cost of 1, whereas the grass field has a cost of 3. Each spike, which is shown in yellow in Fig. 1, is recorded in the AER table, with its neuron ID and time step.

To find the best path between the start and goal locations, we used the list of spikes held in the AER table. From the goal, the list was searched for the most recent spike from a neuron whose location was adjacent to the goal location. If more than one spike met this criterion, the neuron whose location corresponded to the lowest cost and was closest to the start location was chosen. This iteratively proceeded from the goal through other neuron locations until a spike at the start location was found. The bottom right image in Fig. 1 shows the found path.

In complex environments, there was the potential for multiple waves to occur and collide (Fig. 2). In this case, the AER table could contain more than one path. To find the best path, a second pass was made through algorithm with a temporary map that had a cost of 1 for the paths from the first pass, with the rest of the map set to 20. This second pass of the spike wave algorithm ensured that the resulting path was most efficient in terms of length and cost.

D. A* Path Planner

For comparison purposes, we implemented the A* algorithm [5], which is commonly used in path planning. A* uses a best-first search and attempts to find a least-cost path from the start location to the goal location. The cost includes the Euclidean distance from the start, the Euclidean distance

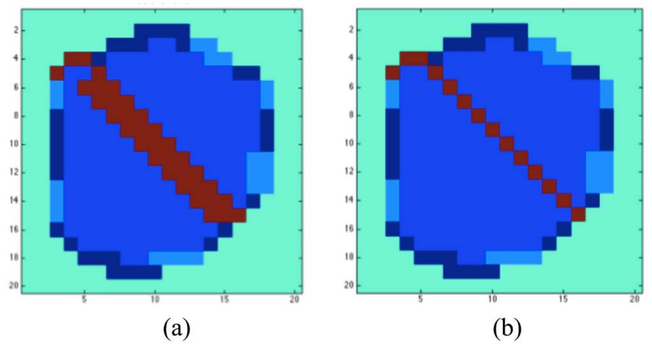


Fig. 2. In some instances, the collision of multiple spike waves generated inefficient paths (left). This was remedied by adding a second pass through the algorithm with a cost map containing just the paths from the first pass (right). (a) Path with wave collision. (b) Remedied path.

from the goal, and the cost of traversing the location. From the start location, adjacent locations are placed in a node list. Then the node list is searched for the node with the lowest cost. The location corresponding to this low-cost node is expanded by placing adjacent, unevaluated locations on the node list. The process is repeated until the goal location is reached. The A* algorithm can find the shortest path based on its cost function.

E. Map of Environment

To demonstrate the effectiveness of our spiking wavefront planner, we tested the algorithm in a real environment through a variety of terrains in Aldrich Park, a 19-acre botanical garden at the University of California at Irvine. Two sections of the park, an open area and a cluttered area (Fig. 3) were transformed into 20×20 grid maps encoding the costs of traveling and the global positioning system (GPS) coordinates. We generated the GPS coordinates by pacing off the area with a smartphone (Samsung Galaxy S5) and recording the GPS points with an Android application.

Two maps created from the sections of Aldrich Park consisted of a 20×20 grid of GPS coordinates and terrain costs (see Fig. 3). The first, referred to as map 1, was in an open grassy area of the park, which was surrounded by a paved sidewalk. In one variant, referred to as "without road" [Fig. 4(a)], the grassy area had a cost of 3, and all other areas had a cost of 9. In the "with road" variant [Fig. 4(b)], the paved sidewalk around the grassy area was given a cost of 1. In the "with road and obstacles" variant [Fig. 4(c)], benches, bushes, and trees were given a cost of 6. The second map, referred to as map 2 [Fig. 4(d)], had an outer region with a cost of 6, large trees and brush had a cost of 10, the paved road had a cost of 1, and the gravel road had a cost of 2. Map 2 was stretched horizontally such that the asphalt path location roughly matched the asphalt path of map 1, allowing for better route comparisons between maps. These maps were used in both simulations and in autonomous robot experiments.

F. Robot Hardware and Software Design

For the robot experiments, we used the Android-based robotic platform [23], a mobile ground robot constructed from off-the-shelf commodity parts and controlled through

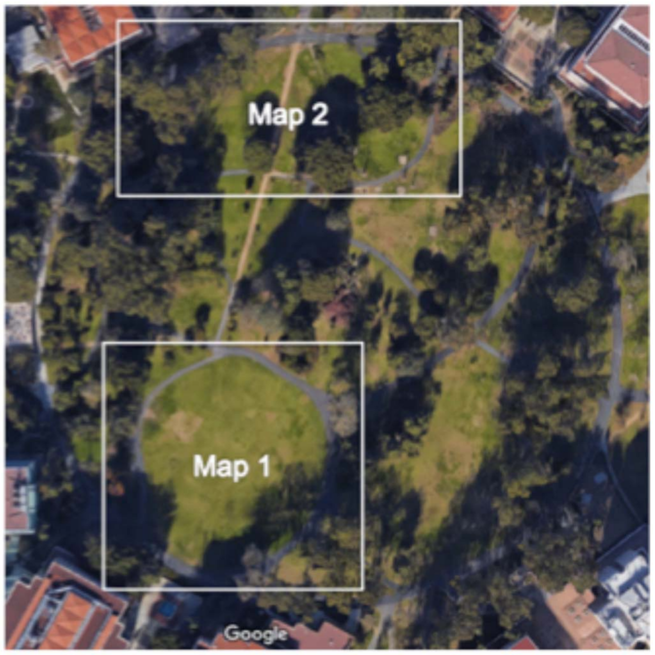


Fig. 3. Google satellite image of Aldrich Park at the University of California at Irvine. Two sections of the park (boxed) were transformed into cost maps (map 1 as bottom box and map 2 as top box) for the spiking wave planner. Imagery©2016 Google.

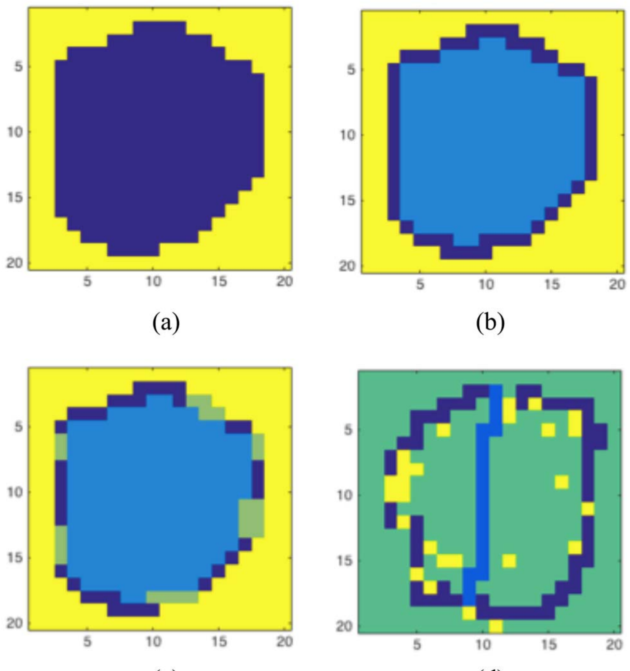


Fig. 4. 20×20 cost grids created from two areas of Aldrich Park. (a) Open area with uniform low cost in the traversable area, and high cost outside of this area. (b) Same area as (a) but with a lower cost for the surrounding road. (c) Same area as (b) but with obstacles denoting benches, bushes, and trees near the road. (d) Second area in Aldrich Park with high cost for trees, low cost for asphalt roads, and medium cost for dirt roads. (a) Map 1—without road. (b) Map 1—with road. (c) Map 1—with road and obstacles. (d) Map 2—with road and obstacles.

an Android smartphone (see Fig. 5). An IOIO-OTG (www.sparkfun.com/products/13613) microcontroller communicated with the Android smartphone via a Bluetooth connection and relayed motor commands to a separate RoboClaw motor controller (www.pololu.com/product/1499) for steering the Dagu Wild Thumper 6-Wheel Drive All-Terrain chassis (www.pololu.com/product/1563robot). The robot used a differential steering technique, moving the left and right sets of wheels at differing speeds to achieve different degrees of turning. Additional sensors and actuators were also connected to the robot through the IOIO-OTG, including several MaxBotix LV-MaxSonar sensors (http://www.maxbotix.com/Ultrasonic_Sensors/MB1000.htm) and an SPT200 pan and tilt unit (www.servocity.com/spt200) for controlling the view of the smartphone camera. Software for controlling the robot was created using the Android Software Development Kit. The software application was written in Java using Android Studio and deployed on a Samsung Galaxy S5 smartphone. The application utilized the phone's built-in accelerometer, gyroscope, compass, and GPS.

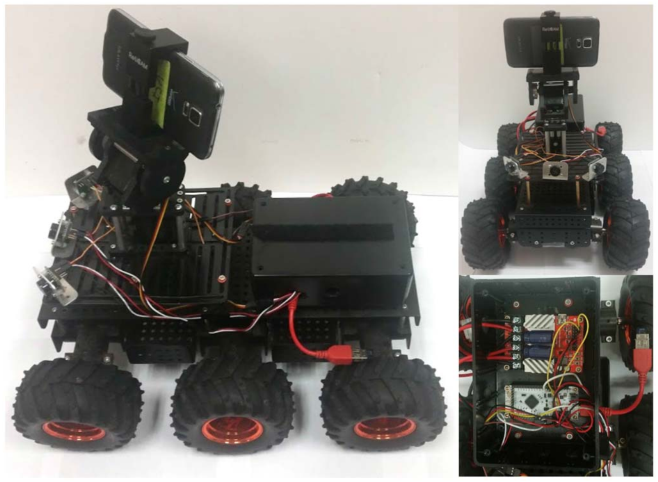


Fig. 5. Android-based robotic platform. Left: side view of ground robot. A flexible pan and tilt unit controls view of smartphone camera. Top right: front view of ground robot. Three LV-MaxSonar sensors are able to detect obstacles up to 254 inches away. Bottom right: top view of component enclosure. An IOIO-OTG microcontroller (below) serves as the central hub for communication, including sending motor commands to the motor controller (above).

G. Computation

Simulations and robot experiments were run to test the spike wave algorithm. The simulations of the spike wave and the A* algorithm were run in MATLAB. For robot experiments, the spike wavefront algorithm, robot I/O, and robot control software were implemented in Java using Android Studio, and run as an app on a Samsung Galaxy S5. Fig. 6 shows a screenshot of the Android application. A graphical user interface (GUI) on the phone allowed the user to input start and goal grid locations, as well as select a map. The app then generated a path using the spike wavefront planner described in Sections II-A–II-C. The phone then displayed the desired path on the GUI (see Fig. 6). Once the operator pressed the auto button on the GUI, the app generated a list of ordered GPS waypoints, from the start to the goal location, from the path grid locations. The robot then used a navigation strategy to visit each waypoint on the list in succession. The robot

TABLE I
COMPARISON BETWEEN SPIKE WAVEFRONT PLANNER AND A* PLANNER IN DIFFERENT ENVIRONMENTS

	Path Length		***Path Cost***		***Time(ms)***	
	Spike Wave	A*	Spike Wave	A*	Spike Wave	A*
Map 1 - Without Road	8.5	8.5	25.5	25.5	6.11	1.08
Map 1 - With Road	10	9	21	24	3.81	0.76
Map 1 - Road and Obstacles	9	9	24	24.5	4.45	0.88
Map 2 - Road and Obstacles	13	11	34.5	42.5	5.61	2.71

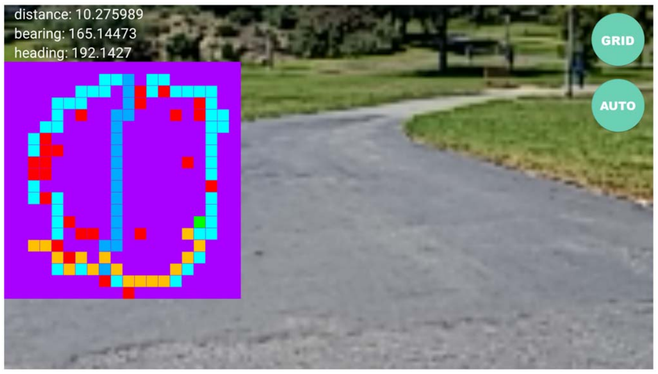


Fig. 6. Screenshot of app used for robot navigation. The screen displays a camera view overlaid with information about distance to the destination, bearing to the destination ranging $0°-360°$ from true north, heading direction (also ranging $0°-360°$ from true north), and the 2-D cost grid. Colors on the grid ranging from dark blue to red indicated the costs of the grid, with tree locations marked at highest cost in red. The planned path of the robot is indicated in yellow and the current location of the robot is marked in green. The "grid" button toggles the grid view on and off and the "auto" button switches the robot into and out of autonomous navigation mode.

stopped moving once the last waypoint, which represented the goal location, was reached.

For robot navigation, a GPS location was queried using the Google Play services location API. The bearing direction from the current GPS location of the robot to a desired waypoint was calculated using the Android API function bearingTo. A second value, the heading, was calculated by subtracting declination of the robot's location to the smartphone compass value, which was relative to magnetic north. This resulted in an azimuth direction relative to true north. The robot traveled forward and steered in attempt to minimize the difference between the bearing and heading. The steering direction was determined by deciding whether turning left or turning right would require the least amount of steering to match the bearing and heading. The navigation procedure continued until the distance between the robot's location and the current waypoint was less than 10 m, at which point the next waypoint in the path list was selected.

III. RESULTS

A. Path Planning Simulations

Table I shows path and cost metrics for simulated path planning that compared the spike wave algorithm with the A* path planner. Simulations were run with all four map variants: 1) map 1—without road; 2) map 1—with road; 3) map 1—with road and obstacles; and 4) map 2—with road and obstacles. One hundred start and goal locations were randomly chosen, in which the locations could not be out of bounds and the Euclidean distance between the start and goal was greater than five grid units.

The path lengths between the two algorithms were nearly identical (see Table I), but the spike wave algorithm found lower cost paths, especially when there were obstacles and roads present ($p < 0.001$; Wilcoxon Ranksum). This is because the spike wave algorithm depends primarily on cost, whereas our A* implementation uses the common and standard heuristic of Euclidean distance in addition to the cost of a node on the map. Although A* is proven to be optimal given an admissible heuristic [5], our heuristic is only admissible when calculating for shortest path. A varied and dynamically changing environment would make a cost-admissible heuristic more difficult to determine, whereas the spike wave algorithm inherently includes both distance and cost in its calculation by combining neighbor connectivity and axonal delay.

The A* algorithm ran faster than the spike wave algorithm when calculating the paths, as measured by the tic/toc functions in MATLAB (see Table I). Interestingly, this difference became smaller as the maps became more complex (see map 2 in Table I). This is due to the presence of low cost roads among high cost obstacles, which leads to the algorithm requiring less neural activity to calculate a path. In Section IV, we discuss how the spike wave algorithm can be made parallel, asynchronous, and implemented on neuromorphic hardware. This should allow for substantial speedups in processing, and reduction in power consumption, which can be quantifiably measured against the baseline run times reported here.

B. Robotic Experiments

Given that the spike wavefront planner showed possible advantages over a traditional approach in simulation, we aimed to test the plausibility of embedding the planning algorithm on an autonomous robot with limited power and computational resources. Robot experiments were conducted in Aldrich Park on the campus of the University of California at Irvine (see Fig. 3). For each map, we tested a set of six routes with the same start and end coordinates on the cost grids. See Figs. 7–10 for route start and end coordinates. The generated routes were different depending on whether roads and obstacles were taken into account. For each route in a given map, the robot ran four trials, following the route

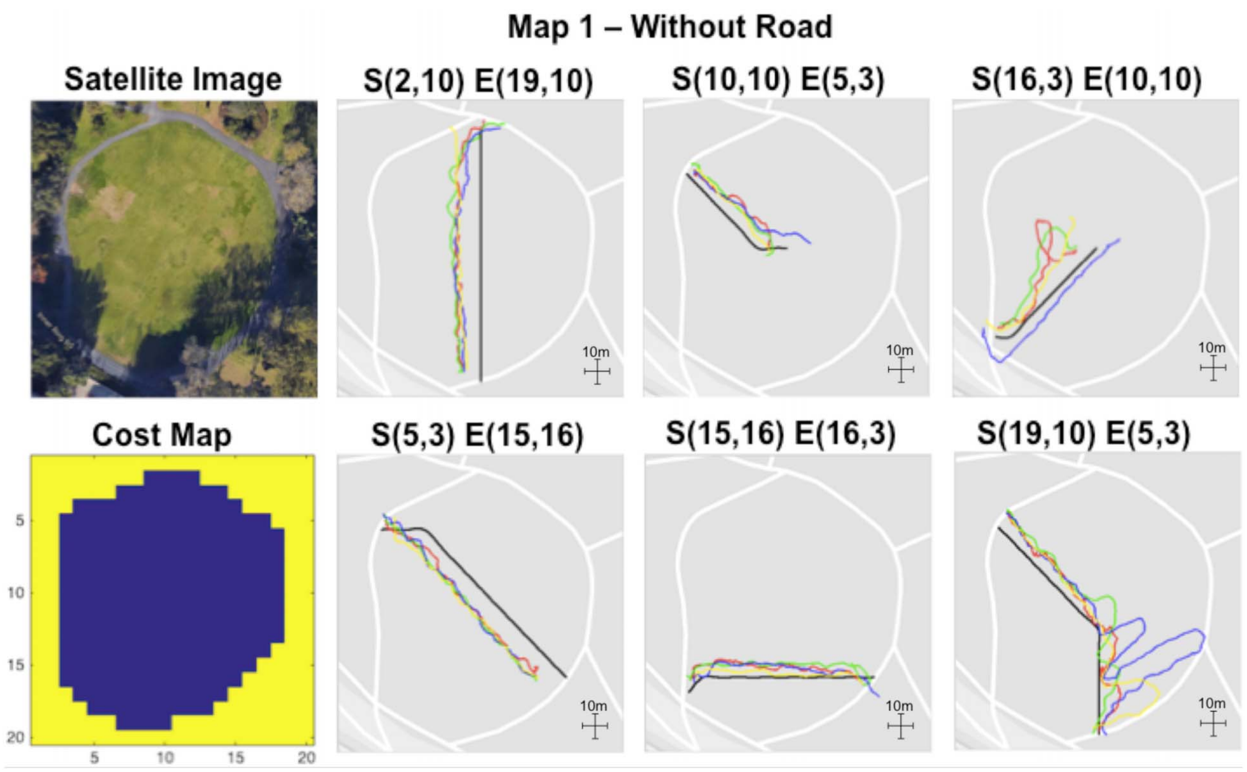


Fig. 7. Experimental results for the spiking path planning algorithm on map 1, with no unique costs encoded for the path. Start and end locations are noted by their row and column position on the cost map. Black lines indicate the planned route and the four colored lines indicate the actual route taken by the robot. Scale bars indicate the length of 10 m along latitudinal and longitudinal axes, indicating the size of error threshold of our navigation strategy. Imagery©2016 Google.

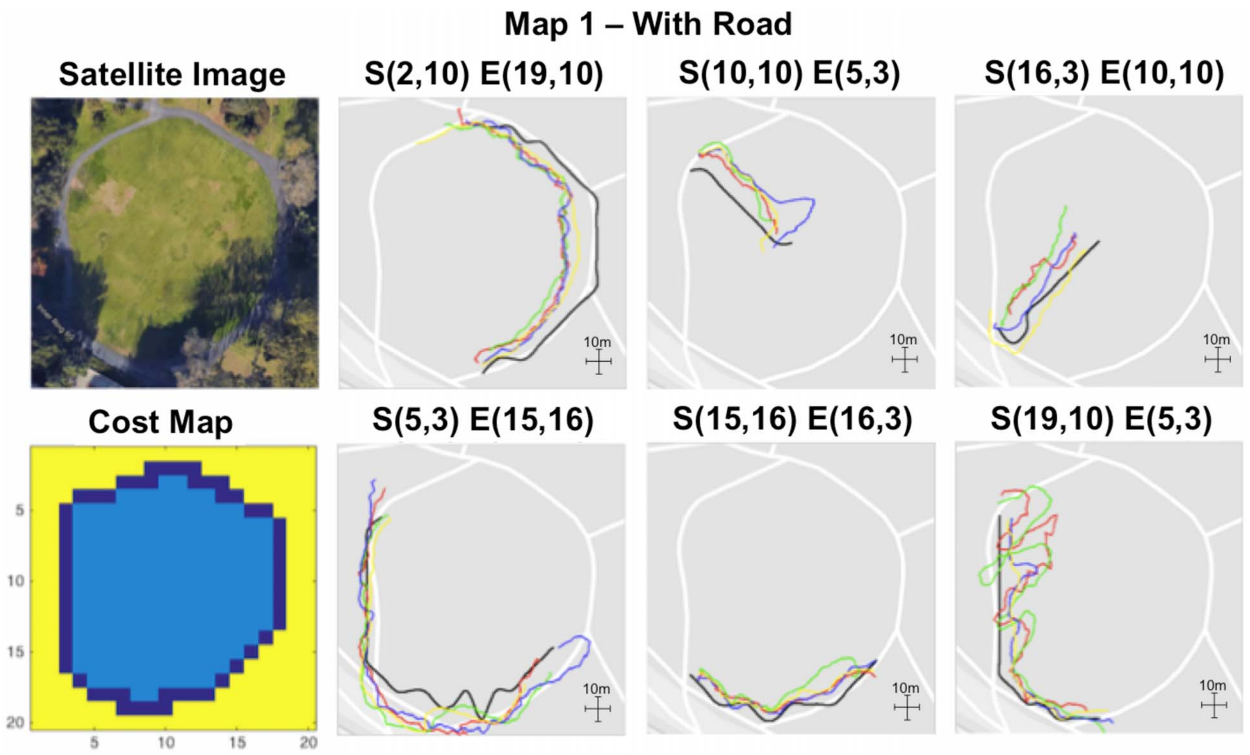


Fig. 8. Experimental results for the spiking path planning algorithm on map 1, with lower costs encoded for the path. Black lines indicate the planned route and the four colored lines indicate the actual route taken by the robot. Imagery©2016 Google.

produced by the spiking wavefront path planner. To account for changing satellite conditions and other environmental factors, we spread out the testing times to sample the variance of GPS signal quality. The first two runs were performed in the morning and the last two runs were performed in the afternoon. Since the robot only relied on GPS and compass to

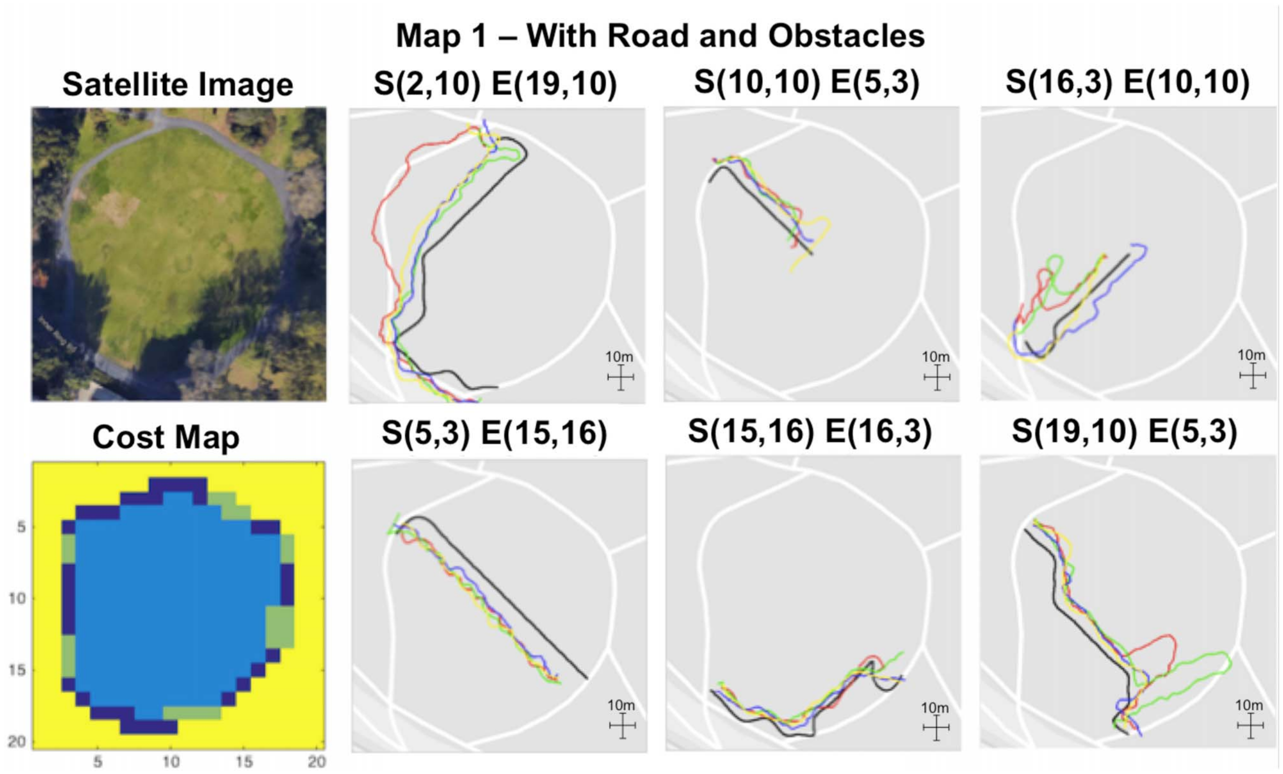


Fig. 9. Experimental results for the spiking path planning algorithm on map 1, with lower costs encoded for the road and higher costs for obstacles. Black lines indicate the planned route and the four colored lines indicate the actual route taken by the robot. Imagery©2016 Google.

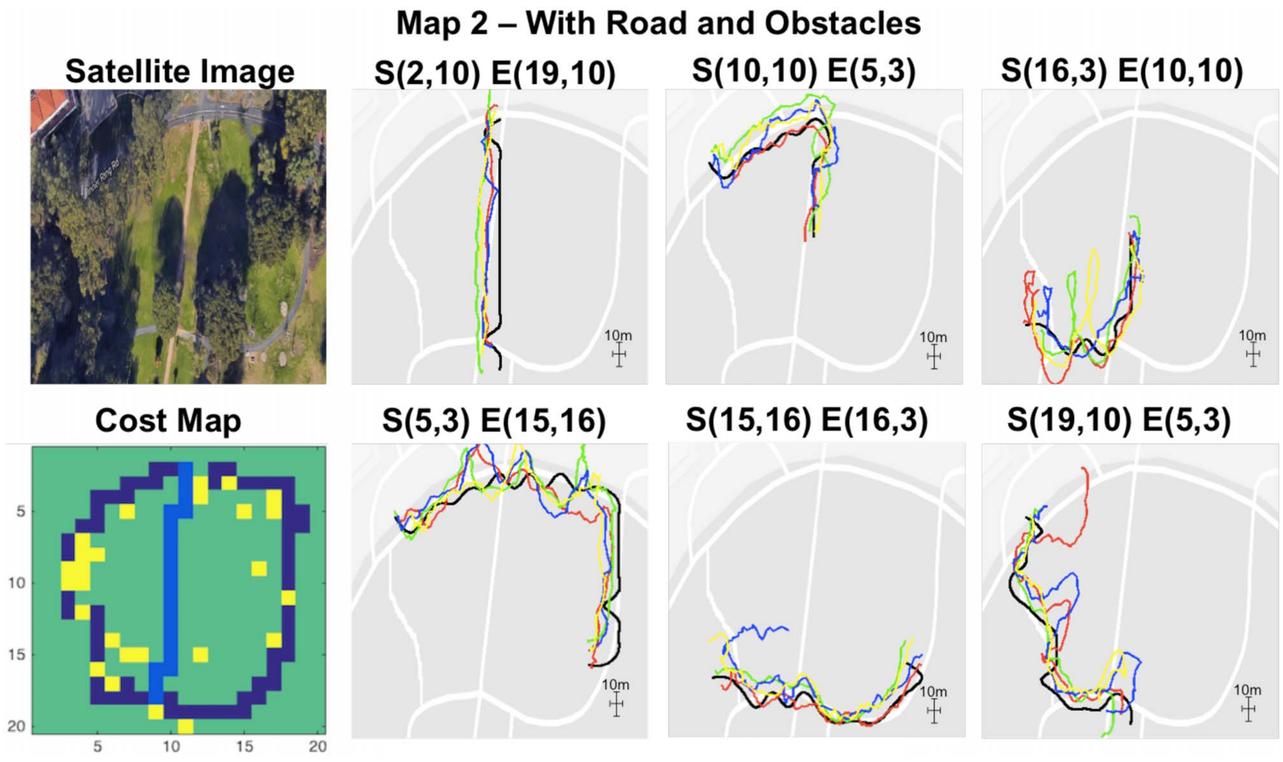


Fig. 10. Experimental results for the spiking path planning algorithm on map 2, with different costs encoded for the asphalt road, dirt path, and trees. Black lines indicate the planned route and the four colored lines indicate the actual route taken by the robot. Note that the 10-m scale bars indicate that the image has been compressed along the longitudinal axis. Imagery©2016 Google.

navigate, it was sometimes necessary to manually redirect the robot slightly away from undetected obstacles. This occurred very infrequently in map 1, and more frequently in map 2 due to the presence of dense vegetation and an abundance of obstacles. In Section IV, we discuss ways to mitigate these interventions.

TABLE II
FRÉCHET DISTANCES (IN METERS) BETWEEN PLANNED ROUTE AND ACTUAL ROUTE FOR SPIKE WAVEFRONT PLANNER

	Route					
	S(2,10) E(19,10)	S(10,10) E(5,3)	S(16,3) E(10,10)	S(5,3) E(15,16)	S(15,16) E(16,3)	S(19,10) E(5,3)
Map 1 - Without Road	12.54±1.33	9.28±1.33	18.20±6.15	13.05±0.34	8.39±1.99	23.70±15.98
Map 1 - With Road	19.44±6.37	14.28±4.29	13.04±5.08	16.56±2.29	8.73±3.88	18.15±7.24
Map 1 - Road and Obstacles	18.67±10.87	11.47±2.56	16.06±6.52	12.87±0.22	7.92±2.36	21.44±15.99
Map 2 - Road and Obstacles	11.69±2.23	10.69±2.75	18.46±5.64	14.78±0.74	20.11±17.41	22.98±8.48

Overall, the actual robot trajectories matched the desired trajectories calculated from the spike wavefront algorithm quite well. For each map condition, Figs. 7–10 show the satellite image of the area, the cost map used by the path planner, and the trajectories for the six routes from a starting grid location to a goal grid location. The trajectories were superimposed on the street view of Google Maps. The black line in these figures shows the desired path, and the four colored lines show a robot trajectory. Occasionally, the GPS signal became unreliable due to buildings, trees, and other environmental noise. This sometimes caused the robot to drive away from the desired destination, requiring the robot to backtrack and visit a missed waypoint.

We used the discrete Fréchet distance [24] as a metric for calculating the similarities of trajectories between the actual robot's movements and the intended route. Intuitively, Fréchet distance is the minimal leash length necessary to physically connect two agents as they walk along their two separate paths. The agents are allowed to pause at any time but not permitted to backtrack, and both must complete their respective paths from start to endpoint. Compared to other comparison techniques such as Hausdorff distance, Fréchet distance takes into account the specific ordering of points on the trajectory, ideal for our experimental conditions. Table II shows mean and standard deviation of Fréchet distances for each of the maps and routes, with the sample size of four trials for each condition.

As our navigation strategy defined reaching a waypoint as arriving within 10 m of the waypoint GPS location, any Fréchet distance near the 10-m threshold should be considered acceptable. We also see the scale at which our hardware and algorithm can operate accurately, which in this case may not be sufficient in some areas of the park but may be sufficient for a car on a commercial road.

IV. DISCUSSION

In prior work, we introduced a path planning algorithm that used spiking neurons and axonal delays to compute efficient routes [13]. The spike wavefront path planner could generate near optimal paths and was comparable to conventional path planning algorithms, such as the A* algorithm or a standard wavefront planner. We introduced a learning rule that represented the cost of traversal in axonal delays. Because the spike wavefront is a local algorithm (i.e., computations are independent and based on neighboring neurons), it is suitable for parallel implementation on neuromorphic hardware, as was shown recently with both grid-based and topological maps [17].

In this paper, we showed that this algorithm was efficient and accurate enough for autonomous robot path planning in complex, outdoor settings. In prior work, maps are idealized, virtual environments. In this paper, the axonal delays represented real-world costs, such as park benches, vegetation, bumpy grass terrain, and trees. Smooth roads were represented with short axonal delays, and this led to the robot choosing easier to traverse terrain, despite the longer overall path. The spiking algorithm, input/output handling, and robot control all ran on an off-the-shelf smartphone with an application written in Java. This demonstrated that the algorithm was lightweight and could support autonomous navigation in real time.

A. Neurobiological Inspiration for the Spike Wavefront Algorithm

The present algorithm was inspired by recent evidence suggesting that the myelin sheath, which wraps around and insulates axons, may undergo a form of activity-dependent plasticity [25], [26]. These studies have shown that the myelin sheath becomes thicker with learning motor skills and cognitive tasks. A thicker myelin sheath implies faster conduction velocities and improved synchrony between neurons.

Based on these findings, we developed a learning rule in which a path that traverses through an easy portion of the environment (e.g., via a road) would have shorter axonal delays than a path that travels through rough terrain. Although it is not known if such spatial navigation costs are represented in the brain in this way, and most likely they are not, this learning rule does investigate a rarely considered form of plasticity. Moreover, manipulating the delays, as was done in this paper, shows how a spiking neural network can solve a real-world problem using a purely temporal code. Other groups have investigated learning rules based on axonal delays. Wang and colleagues have implemented a spike timing delay dependent plasticity rule that can shorten or lengthen the axonal delay between two connected neurons [27], [28]. They showed that altering axonal delays had advantages in forming polychronous neuronal groups, which represent spatiotemporal memories [29], over altering synaptic weights via spike timing dependent plasticity (STDP).

The present algorithm is also inspired by the notion of neuronal waves in the brain. Wave propagation has broad empirical support in motor cortex, sensory cortex, and the hippocampus [30]–[36]. These waves have been suggested as a means to solve the credit assignment problem for associating a conditioned stimulus with the later arrival of an unconditioned stimulus [37]. In this paper, we have shown that neuronal wave dynamics in complex spiking neural network models can be used to associate visual stimuli with noisy tactile inputs in a physical robot [38]. Therefore, the idea of solving problems with spike timing generated by propagating waves of activity has biological and theoretical support.

Relevant to the present task, is the experimental observation of neural activity that represent potential paths through space. Sequences of place cell activity in the hippocampus prior can predict an animal's trajectory through the environment [39]–[43]. These so-called preplays may be a means to assess different possible paths prior to selecting a specific path plan. In a way, this is similar to how the spike wavefront planner operates. Sequences of place activity are generated, and the spike sequence that arrives first at the goal is the one selected for execution.

B. Parallel Implementation of Spike Wavefront Planner on Neuromorphic Hardware

The present path planner calculates paths based on the timing of spiking neurons. Because each neuron can calculate its state independently, the algorithm could realize impressive speedups through parallelization. Moreover, spiking neuron networks are inherently event-driven, that is, a new state is only calculated when an incoming spike has been received. This further reduces computational load. Lastly, by stopping as soon as the first spike is received at a goal node, the spike wavefront planner algorithm only calculates what is necessary. For example, when there were variable costs, such as in map 2, the amount of time to calculate a path with the spike wavefront planner was reduced relative to the A* path planner (see Table I). It should be noted that the A* path planner can be parallelized [44], but unlike this and other conventional algorithms, they cannot take advantage of neuromorphic hardware as can spiking neuron algorithms.

Neuromorphic hardware differs from the conventional Von Neumann computer architecture in that it is asynchronous and event-driven, with parallel computation [8], [9]. The artificial neurons do not take up computation cycles unless they receive a spike event from a connected neuron. Typical neuromorphic designs have the memory, in the form of synapses, co-located with the processing units, that is, the neurons. This allows computations to be local, independent, and parallel. These features allow neuromorphic chips to have low size, weight, and power [8], [45]. Nearly all these chips use spiking neuron elements and some form of AER.

As has been shown in prior implementations, the spike wavefront algorithm is compatible with neuromorphic hardware [14]–[18]. These implementations show the feasibility and parallelization of the wavefront planner. Moreover, they show how this neuromorphic algorithm can generate optimal paths. In addition, IBM's TrueNorth neuromorphic chip was recently embedded on the robot used in the present experiments in an autonomous self-driving application [46]. Considering that the present spike wavefront algorithm has been implemented on TrueNorth [17] a complete neuromorphic path planning system is now feasible on our robot.

This paper builds on these implementations by adding a learning rule to make the planner more flexible and to consider the relative costs of traversing an environment. Axonal delays have been introduced in large-scale spiking neural network simulations [47], [48], but are not typical for neuromorphic hardware. However, some neuromorphic designs include axonal delays [27], [28], [49], [50]. To implement the present algorithm in neuromorphic hardware, all that would be needed is a delay buffer, delay line, or a means to schedule spikes at specific times in the future. Because a synaptic based learning rule, such as STDP, is not needed for the present algorithm, the circuitry to support the spike wavefront planner could be simplified.

In the present algorithm, the AER representation is used to read out the path, which may be a limitation since it requires saving the AER list for each planned route. It also requires a planning calculation and readout for every route. A more natural implementation might use the rank order of the spike wave in a similar way to that proposed by Thorpe *et al.* [51] and VanRullen *et al.* [52]. Such alternative readout implementations will be explored in the future.

C. Comparison to Other Neurally Inspired Path Planning Approaches

The result of our algorithm has complementary parallels to past work in bioinspired algorithms for mobile robot control [53]. For instance, Ni and Yang [54] proposed a neural network for multirobot cooperative hunting in unknown environments, representing space in a 2-D neuron grid and using a shunting model to represent attraction and repulsion agents on the field. Similarly, our algorithm could draw upon these principles, representing not only environmental costs but costs of interacting with other dynamic agents cooperatively and competitively. It also opens the possibility of neuromorphic solutions for the complex tasks of swarm coordination in mobile robots.

Aside from neural navigation models inspired by hippocampal activity and cognitive map representation, cerebellar models of motor control using the delayed eligibility trace learning rule have also been used for spatial motion planning [55], with further developments increasing its efficacy in real environments such as urban expressways and tracks [56]. Perhaps a model of predictive motor control combined with a larger cognitive map representation could be implemented in neuromorphic hardware to form an effective multiscale motion planning system.

D. Simultaneous Path Planning and Mapping

The present algorithm could be modified to build a map as the robot explores its environment. It would need additional sensors to measure the cost of traversal or some other cost function related to navigation. Rather than assuming that the environment is known and static, the robot could update the map with each path it generates. This would require setting the learning rate in (5) to be less than one. In addition, if the spike wavefront planner had a learning rate between 0 and 1, the uncertainty of the cost at a location would be represented. Similar to [57], this would result in the spike wavefront planner predicting the cost of traversing locations in an environment. Moreover, the planner could utilize an exploration/exploitation tradeoff to decide whether to explore unknown regions, or exploit previously navigated regions. Such tradeoffs have been implemented in neurobiologically inspired algorithms [58]–[60]. Such a planner could respond flexibly and fluidly to dynamic environments, or the changing needs of the robot. For example, if the robot needed to get to a location as fast as possible, it might take a direct, but more risky route from the start to goal location. However, if the robot wanted to conserve energy, it might take a longer, but easier path. The different trajectories taken by the robot in Figs. 6–9 demonstrate this capability. Context is represented in the map itself. In a future implementation, one could change the cost values of the map based on the robot's needs, thus changing the robot's behavior.

Additionally, building a map incrementally opens up many possibilities of representing the map besides a 2-D grid configuration. For example, a more flexible arrangement such as a topological map is compatible with our spike wave propagation algorithm, and in fact has recently been achieved with large-scale maps [17]. For increased resolution of map representation, a system of multiscale place recognition [61] may also be considered. Further, a hierarchical spiking neural network [62] could be involved in forming multiscale representation compatible with neuromorphic hardware. Any of these suggested implementations would ease the computational load of 2-D grid implementations of the A* and the spike wavefront propagation algorithm, both of which increase in complexity with the grid resolution.

V. Conclusion

In summary, we have shown that a spike-based wavefront planner can successfully be used on an autonomous robot to navigate natural environments. Developing from the existing literature on spiking path planning algorithms, we showed that the algorithm, which implemented a form of activity-dependent axonal delay plasticity, was sufficient to plan paths based on real costs of traversing an outdoor environment. We further demonstrated that this algorithm could be implemented on a standard smartphone with consumer-grade GPS and compass sensors, suggesting that this may be efficient enough for other autonomous vehicles that do not have access to high performance computing. Because the algorithm relies on spiking neurons and asynchronous, event-driven computation, it can be implemented on neuromorphic hardware, making it power efficient enough for many embedded applications.

Acknowledgment

The authors would like to thank participants in the Telluride Neuromorphic Cognition workshop where the initial implementation and testing of this algorithm occurred.

References

[1] J. O'Keefe and L. Nadel, *The Hippocampus as a Cognitive Map*. Oxford, U.K.: Oxford Univ. Press, 1978.

[2] C. R. Gallistel, *The Organization of Learning*. Cambridge, MA, USA: MIT Press, 1993.

[3] E. C. Tolman, "Cognitive maps in rats and men," *Psychol. Rev.*, vol. 55, no. 4, pp. 189–208, Jul. 1948.

[4] S. M. LaValle, "Motion planning," *IEEE Robot. Autom. Mag.*, vol. 18, no. 1, pp. 79–89, Mar. 2011.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[6] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," *Int. J. Robot. Autom. Syst.*, vol. 10, no. 3, pp. 89–100, 1995.

[7] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[8] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Front. Neurosci.*, vol. 5, May 2011, Art. no. 73.

[9] J. L. Krichmar, P. Coussy, and N. Dutt, "Large-scale spiking neural networks using neuromorphic hardware compatible models," *J. Emerg. Technol. Comput. Syst.*, vol. 11, no. 4, pp. 1–18, 2015.

[10] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 22, no. 2, pp. 224–241, Mar./Apr. 1992.

[11] M. Soulignac, "Feasible and optimal path planning in strong current fields," *IEEE Trans. Robot.*, vol. 27, no. 1, pp. 89–98, Feb. 2011.

[12] R. Bellman, "On a routing problem," *Quart. Appl. Math.*, vol. 16, no. 1, pp. 87–90, 1958.

[13] J. L. Krichmar, "Path planning using a spiking neuron algorithm with axonal delays," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 1219–1226.

[14] S. Koul and T. K. Horiuchi, "Path planning by spike propagation," in *Proc. IEEE Syst. Conf. Biomed. Circuits (BioCAS)*, Atlanta, GA, USA, 2015, pp. 1–4.

[15] S. Koziol, S. Brink, and J. Hasler, "Path planning using a neuron array integrated circuit," in *Proc. IEEE Glob. Conf. Signal Inf. Process. (GlobalSIP)*, Austin, TX, USA, 2013, pp. 663–666.

[16] S. Koziol, S. Brink, and J. Hasler, "A neuromorphic approach to path planning using a reconfigurable neuron array IC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 12, pp. 2724–2737, Dec. 2014.

[17] K. D. Fischl, K. Fair, W.-Y. Tsai, J. Sampson, and A. Andreou, "Large-scale path planning on spiking neuromorphic hardware," in *Proc. IEEE Int. Symp. Circuits Syst.*, Montreal, QC, Canada, 2016.

[18] S. Koziol, P. Hasler, and M. Stilman, "Robot path planning using field programmable analog arrays," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, St. Paul, MN, USA, 2012, pp. 1747–1752.

[19] P. Gaussier, A. Revel, J. P. Banquet, and V. Babeau, "From view cells and place cells to cognitive map learning: Processing stages of the hippocampal system," *Biol. Cybern.*, vol. 86, no. 1, pp. 15–28, Jan. 2002.

[20] F. Ponulak and J. J. Hopfield, "Rapid, parallel path planning by propagating wavefronts of spiking neural activity," *Front. Comput. Neurosci.*, vol. 7, p. 98, Jul. 2013.

[21] M. Quoy, P. Laroque, and P. Gaussier, "Learning and motivational couplings promote smarter behaviors of an animat in an unknown world," *Robot. Auton. Syst.*, vol. 38, nos. 3–7, pp. 149–156, Mar. 2002.

[22] A. V. Samsonovich and G. A. Ascoli, "A simple neural network model of the hippocampus suggesting its pathfinding role in episodic memory retrieval," *Learn Memory*, vol. 12, no. 2, pp. 193–208, Mar./Apr. 2005.

[23] N. Oros and J. L. Krichmar, "Smartphone based robotics: Powerful, flexible and inexpensive robots for hobbyists, educators, students and researchers," Center Embedded Comput. Syst., Univ. California at Irvine, Irvine, CA, USA, CECS Tech. Rep. 13-16, pp. 1–11, 2013.

[24] T. Eiter and H. Mannila, "Computing discrete Fréchet distance," Christian Doppler Lab. Expert Syst., Vienna, Austria, Tech. Rep. CD-TR 94/64, Apr. 1994.

[25] R. D. Fields, "White matter in learning, cognition and psychiatric disorders," *Trends Neurosci.*, vol. 31, no. 7, pp. 361–370, Jul. 2008.

[26] R. D. Fields, "A new mechanism of nervous system plasticity: Activity-dependent myelination," *Nat. Rev. Neurosci.*, vol. 16, no. 12, pp. 756–767, Dec. 2015.

[27] R. Wang *et al.*, "An FPGA implementation of a polychronous spiking neural network with delay adaptation," *Front. Neurosci.*, vol. 7, p. 14, Feb. 2013.

[28] R. M. Wang, T. J. Hamilton, J. C. Tapson, and A. van Schaik, "A mixed-signal implementation of a polychronous spiking neural network with delay adaptation," *Front. Neurosci.*, vol. 8, p. 51, Mar. 2014.

[29] E. M. Izhikevich, "Polychronization: Computation with spikes," *Neural Comput.*, vol. 18, no. 2, pp. 245–282, 2006.

[30] D. Rubino, K. A. Robbins, and N. G. Hatsopoulos, "Propagating waves mediate information transfer in the motor cortex," *Nat. Neurosci.*, vol. 9, no. 12, pp. 1549–1557, Nov. 2006.

[31] A. Benucci, R. A. Frazor, and M. Carandini, "Standing waves and traveling waves distinguish two circuits in visual cortex," *Neuron*, vol. 55, no. 1, pp. 103–117, Jul. 2007.

[32] F. Han, N. Caporale, and Y. Dan, "Reverberation of recent visual experience in spontaneous cortical waves," *Neuron*, vol. 60, no. 2, pp. 321–327, Oct. 2008.

[33] J. Y. Wu, X. Huang, and C. Zhang, "Propagating waves of activity in the neocortex: What they are, what they do," *Neuroscience*, vol. 14, no. 5, pp. 487–502, Oct. 2008.

[34] E. V. Lubenov and A. G. Siapas, "Hippocampal theta oscillations are travelling waves," *Nature*, vol. 459, pp. 534–539, May 2009.

[35] T. K. Sato, I. Nauhaus, and M. Carandini, "Traveling waves in visual cortex," *Neuron*, vol. 75, no. 2, pp. 218–229, Jul. 2012.

[36] T. P. Zanos, P. J. Mineault, K. T. Nasiotis, D. Guitton, and C. C. Pack, "A sensorimotor role for traveling waves in primate visual cortex," *Neuron*, vol. 85, no. 3, pp. 615–627, Feb. 2015.

[37] J. H. Palmer and P. Gong, "Associative learning of classical conditioning as an emergent property of spatially extended spiking neural circuits with synaptic plasticity," *Front. Comput. Neurosci.*, vol. 8, p. 79, Jul. 2014.

[38] T.-S. Chou, L. D. Bucci, and J. L. Krichmar, "Learning touch preferences with a tactile robot using dopamine modulated STDP in a model of insular cortex," *Front. Neurorobot.*, vol. 9, p. 6, Jul. 2015.

[39] G. Dragoi and S. Tonegawa, "Preplay of future place cell sequences by hippocampal cellular assemblies," *Nature*, vol. 469, no. 7330, pp. 397–401, Jan. 2011.

[40] G. Dragoi and S. Tonegawa, "Distinct preplay of multiple novel spatial experiences in the rat," *Proc. Nat. Acad. Sci. USA*, vol. 110, no. 22, pp. 9100–9105, May 2013.

[41] B. E. Pfeiffer and D. J. Foster, "Hippocampal place-cell sequences depict future paths to remembered goals," *Nature*, vol. 497, no. 7447, pp. 74–79, May 2013.

[42] B. E. Pfeiffer and D. J. Foster, "PLACE CELLS. Autoassociative dynamics in the generation of sequences of hippocampal place cells," *Science*, vol. 349, no. 6244, pp. 180–183, Jul. 2015.

[43] D. Silva, T. Feng, and D. J. Foster, "Trajectory events across hippocampal place cells require previous experience," *Nat. Neurosci.*, vol. 18, no. 12, pp. 1772–1779, Dec. 2015.

[44] Y. Zhou and J. Zeng, "Massively parallel A* search on a GPU," in *Proc. 29th AAAI Conf. AI*, Austin, TX, USA, 2015, pp. 1248–1255.

[45] N. Srinivasa and J. Cruz-Albrecht, "Neuromorphic adaptive plastic scalable electronics: Analog learning systems," *IEEE Pulse*, vol. 3, no. 1, pp. 51–56, Jan. 2012.

[46] T. Hwu, J. Isbell, N. Oros, and J. Krichmar, "A self-driving robot using deep convolutional neural networks on neuromorphic hardware," *arXiv preprint arXiv:1611.01235v1*, Nov. 2016. [Online]. Available: http://arxiv.org/abs/1611.01235

[47] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems," *Nat. Acad. Sci. USA*, vol. 105, no. 9, pp. 3593–3598, Mar. 2008.

[48] E. M. Izhikevich, J. A. Gally, and G. M. Edelman, "Spike-timing dynamics of neuronal groups," *Cereb. Cortex*, vol. 14, no. 8, pp. 933–944, Aug. 2004.

[49] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and STDP integrated circuits," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 3, pp. 246–256, Jun. 2012.

[50] R. M. Wang, T. J. Hamilton, J. C. Tapson, and A. V. Schaik, "A neuromorphic implementation of multiple spike-timing synaptic plasticity rules for large-scale neural networks," *Front. Neurosci.*, vol. 9, p. 180, May 2015.

[51] S. Thorpe, A. Delorme, and R. VanRullen, "Spike-based strategies for rapid processing," *Neural Netw.*, vol. 14, nos. 6–7, pp. 715–725, Jul./Sep. 2001.

[52] R. VanRullen, R. Guyonneau, and S. J. Thorpe, "Spike times make sense," *Trends Neurosci.*, vol. 28, no. 1, pp. 1–4, Jan. 2005.

[53] J. Ni, L. Wu, X. Fan, and S. X. Yang, "Bioinspired intelligent algorithm and its applications for mobile robot control: A survey," *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–16, Jan. 2016.

[54] J. Ni and S. X. Yang, "Bioinspired neural network for real-time cooperative hunting by multirobots in unknown environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2062–2077, Dec. 2011.

[55] J. L. McKinstry, G. M. Edelman, and J. L. Krichmar, "A cerebellar model for predictive motor control tested in a brain-based device," *Nat. Acad. Sci. USA*, vol. 103, no. 9, pp. 3387–3392, 2006.

[56] V. A. Shim, C. S. N. Ranjit, B. Tian, M. Yuan, and H. Tang, "A simplified cerebellar model with priority-based delayed eligibility trace learning for motor control," *IEEE Trans. Auton. Mental Develop.*, vol. 7, no. 1, pp. 26–38, Mar. 2015.

[57] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, no. 5306, pp. 1593–1599, Mar. 1997.

[58] B. R. Cox and J. L. Krichmar, "Neuromodulation as a robot controller," *IEEE Robot. Autom. Mag.*, vol. 16, no. 3, pp. 72–80, Sep. 2009.

[59] J. L. Krichmar, "The neuromodulatory system a framework for survival and adaptive behavior in a challenging world," *Adapt. Behav.*, vol. 16, no. 6, pp. 385–399, 2008.

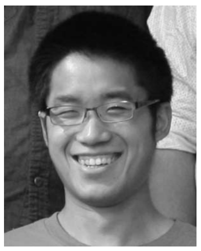
[60] G. Aston-Jones and J. D. Cohen, "An integrative theory of locus coeruleus-norepinephrine function: Adaptive gain and optimal performance," *Annu. Rev. Neurosci.*, vol. 28, no. 1, pp. 403–450, 2005.

[61] Z. Chen, A. Jacobson, U. M. Erdem, M. E. Hasselmo, and M. Milford, "Multi-scale bio-inspired place recognition," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, 2014, pp. 1895–1901.

[62] M. Beyeler, N. D. Dutt, and J. L. Krichmar, "Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule," *Neural Netw.*, vol. 48, pp. 109–124, Dec. 2013.

Tiffany Hwu received the B.A. degree in computer science and cognitive science from the University of California at Berkeley, Berkeley, CA, USA, in 2014, and the M.S. degree in cognitive neuroscience from the University of California at Irvine, Irvine, CA, USA, in 2016, where she is currently pursuing the Ph.D. degree in psychology.

She is a Systems Technical Intern of Basic Research with Northrop Grumman, Redondo Beach, CA, USA. Her current research interests include biologically inspired spatial navigation, transfer learning, neurorobotics, and decision making.



Alexander Y. Wang is currently pursuing the B.S. degree in mechanical engineering with the University of California at Irvine, Irvine, CA, USA.

He has over four years of engineering internship experience, researching for industry leaders in aerospace, additive manufacturing, and microelectro-mechanical systems. He is currently building a diverse array of robots at the Cognitive Anteater Robotics Laboratory, University of California at Irvine, and is part of a broad outreach program in Orange County, mentoring and inspiring high school students to pursue science, technology, engineering, and mathematics fields via building rescue robots.

Nicolas Oros received the B.Sc. degree in computer science and the M.Sc. and Ph.D. degrees in artificial intelligence from the University of Hertsfordshire, Hatfield, U.K., in 2005, 2006, and 2010, respectively.

He was a Post-Doctoral Research Scholar with the Cognitive Anteater Robotics Laboratory, University of California at Irvine, Irvine, CA, USA. He is currently a Senior Research Scientist with BrainChip Inc., Aliso Viejo, CA, USA. His current research interests include neural computation, evolutionary computation, neuromodulatory systems, artificial life, and robotics.

Jeffrey L. Krichmar received the B.S. degree in computer science from the University of Massachusetts at Amherst, Amherst, MA, USA, in 1983, the M.S. degree in computer science from George Washington University, Washington, DC, USA, in 1991, and the Ph.D. degree in computational sciences and informatics from George Mason University, Fairfax, VA, USA, in 1997.

He spent 15 years as a Software Engineer on projects ranging from the PATRIOT Missile System at the Raytheon Corporation, Waltham, MA, USA, to Air Traffic Control for the Federal Systems Division of IBM, Rockville, MD, USA. In 1997, he became an Assistant Professor with the Krasnow Institute for Advanced Study, George Mason University. From 1999 to 2007, he was a Senior Fellow of Theoretical Neurobiology with Neurosciences Institute, San Diego, CA, USA. He is currently a Professor with the Department of Cognitive Sciences and the Department of Computer Science, University of California at Irvine, Irvine, CA, USA. His current research interests include neurorobotics, embodied cognition, biologically plausible models of learning and memory, and the effect of neural architecture on neural function.