ABSTRACT

Title of Dissertation:	THE CURVED OPENSPACE ALGORITHM AND NEUROMORPHIC MECHANISMS FOR SONAR-BASED OBSTACLE AVOIDANCE
	Chenxi Wen, Doctor of Philosophy, 2021
Dissertation directed by:	Professor Timothy K Horiuchi, Department of Electrical and Computer Engineering

Bats are known for their ability to pursue a goal while avoiding obstacles in a cluttered environment using ultrasonic echolocation. This dissertation explores two neuromorphic mechanisms involved in such a task: a conductance-based neuron circuit for azimuthal echolocation whose dynamic range can be expanded by power-law compression, and a sonar-based obstacle avoidance algorithm with its spike-latency model implementation. Bats and other mammals use interaural level differences (ILD) to estimate the direction of high-frequency sounds. To compute the ILD of a sound, independent of overall loudness, excitatory and inhibitory synaptic conductances (encoding the left and right amplitudes) are hypothesized to compete in the neurons of the lateral superior olive. Interestingly, this neural model can also accept power-law compressed amplitudes that can allow a much larger range of input signal levels, a common limitation in neural coding. This dissertation demonstrates the use of square-root and cube-root compression with a neuromorphic VLSI neuron to expand the range of distances over which ILD can be used to estimate echo direction in a sonar system based on echolocating bats. However, many questions remain regarding how to achieve the rapid control of a sonar-guided vehicle to pursue a goal while avoiding obstacles. Taking into account the limited field-of-view of practical sonar systems and vehicle kinematics, we propose an obstacle avoidance algorithm that maps the 2-D sensory space into a 1-D motor space and evaluates motor actions while combining obstacles and goal information. A winner-take-all (WTA) mechanism is used to select the final steering action. To avoid unnecessary scanning of the environment, an attentional system is proposed to control the directions of sonar pings for efficient, task-driven, sensory data collection. A mobile robot driven by the proposed algorithm was capable of navigating through a cluttered environment using a realistic sonar system. The algorithm was tested on a mobile robot, and it is implemented on a neural model using spike-timing representations, a spike-latency memory, and a "race-to-first-spike" WTA circuit. This dissertation also proposed a CMOS floating-gate circuit for artificial neural network synapse memories that can achieve a fixed rate of weight increase (adding vectors) or proportional decay (normalization) on the synaptic weight.

THE CURVED OPENSPACE ALGORITHM AND NEUROMORPHIC MECHANISMS FOR SONAR-BASED OBSTACLE AVOIDANCE

by

Chenxi Wen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Timothy K Horiuchi, Chair
Professor P. S. Krishnaprasad
Professor Pamela Abshire
Professor Nuno Martins
Professor Daniel Butts

Acknowledgments

I'm extremely grateful to my advisor, Dr. Horiuchi, for your guidance throughout my Ph.D. study. Thank you for leading me into the amazing world of neuromorphic engineering. Your love and passion for academic research have shaped my passion for research and drove me to keep pushing the frontiers of knowledge even after I graduate. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. Your hands-on experience in circuit design has helped me tremendously to become a better circuit designer. You have given me the opportunity to attend the Telluride workshop and join the amazing community of neuromorphic engineering. I have also become a much better writer and presenter under your guidance, which could only benefit me for the rest of my career.

I would like to extend my deepest gratitude to Dr. Krishnaprasad and Dr. Abshire, who were on my research proposal committee, for your advice and feedback on my work. I'm grateful to Dr. Krishnaprasad for giving me tremendous support on the robotic testing and allowing me to use the tracking system in your lab. I would like to thank Dr. Abshire for your support on layout and fabrication. I would like to extend my sincere thanks to Dr. Martins and Dr. Butts for serving on my dissertation defense committee.

I'm also grateful to Shashikant from the CSSL group for your expertise and support on the design and testing of my VLSI chip. I still vividly remember the long conversations that we had in the lab regarding circuits and research in general. I would like to extend my gratitude to Jacob for your help on 3-D printing and sonar systems.

The time we spent in Telluride together has been extremely fun. I would like to thank Xincheng Li for helping me use the tracking system for my mobile robot. I would like to thank Sheung Lu for your support in using the Cadence software. I would like to thank Dr. Yavuz Oruc for your guidance and support during my first year in Maryland. I would like to extend my thanks to Dr. Ankur Srivastava for your guidance and insights in academic research during my first year of Ph.D. study.

I would like to thank Maryland Industrial Partnerships (MIPS) Program for financial support. I would like to thank MOSIS for providing fabrication services for research and educational purposes.

I'm deeply indebted to my parents and my grandparents. Any milestones in my study and research would not have been possible without their support. I feel deeply sorry that my mother and my grandfathers are not able to witness this moment. I'm extremely grateful for what you have done in shaping the person that I am today, and I hope I can make you proud with my achievements.

I would like to express my deepest appreciation to my wife, Zhao Tong. Throughout my Ph.D. study, you have been my strongest emotional support and you have been the one person that I know I can always fall back on no matter what happens. I'm extremely grateful for your willingness to accompany and support me even if that means sacrificing your time, study, and career. One of my greatest achievements in life has been the unbreakable bond between us, especially during the pandemic where we did not get a chance to meet for over a year and a half. I'm extremely proud of you and our relationship, and I hope you are too.

Table of Contents

Chapter 1: Introduction

Traveling through an environment towards a goal without colliding with obstacles is one of many essential abilities for animals to survive. Animals are often able to detect obstacles through different types of sensors and quickly decide on the motions to avoid them. Echolocating bats are excellent examples of creatures that possess such capability. They can use ultrasonic echoes to perceive their surroundings and fly through dense forests in complete darkness with ease. During their hunt for flying insects or other prey, they can avoid obstacles while homing in on their prey at the same time. Big brown bats (*Eptesicus fuscus*), using sonar calls that last 2 to 3 milliseconds, can send out sonar pulses to detect obstacles at a rate of up to 90 Hz and can fly past obstacles at a flight speed of 2 m/s to 5 m/s even in an indoor environment [1]. How biological systems can deftly transform the storm of sensory information into motor actions to pursue the goal while avoiding obstacles has been a fascinating research topic for engineers and neuroscientists.

Echolocating bats have the ability to emit an ultrasonic sound pulse and estimate the range of objects by measuring the arrival time of the echoes. Combined with the more general ability of mammals to determine the direction of sound, Big brown bats (*Eptesicus fuscus*) are shown to be able to discriminate range differences as small as 1 cm [2] and angular differences of 6° [3], while other species such as the greater spear-nosed bat (*Phyllostomus hastatus*) can differentiate objects that are 4° apart. The ultrasonic sound pulses emitted by echolocating bats have frequencies ranging from 20kHz to 100kHz and have short wavelengths (~ 3mm to 17mm) that can produce detectable echoes from small insects [4]. Because of the high frequency and a small head size, echolocating bats have been shown to mainly rely on the interaural level difference (ILD) of an arriving sound created by its head and pinnae to localize the sound.

Neurons of the mammalian brainstem lateral superior olive (LSO) are well known to be sensitive to ILD and generate spikes when a sound arrives from an angle greater (from the midline) than its characteristic threshold angle (i.e., threshold ILD) and thus a population of neurons with different thresholds is needed to determine the specific angle. Although a "place code" could be derived from LSO neurons (as seen in birds), mammals appear to use the population for a "rate code" for encoding sound direction at both low frequencies [5] and high frequencies [6], where the summed spikes (or spike rate) from the whole population indicates the direction. To accurately estimate sound angle, accurate level comparisons between ears must be made.

Neuromorphic analog VLSI circuit models of LSO neurons for sound localization have been constructed previously [7]-[11] but these efforts modeled individual LSO neurons and were assumed to require a second stage of processing to estimate the sound angle. In the work presented in Chapter 2, to avoid constructing a large array of neurons with different ILD thresholds, a single, fast-spiking neuron is used to represent the whole LSO population on one side of the brain at a given frequency. This results in only two neurons for a given frequency, one to represent angles to the left of center and another to represent angles to the right of center. We report on testing of two VLSI LSO neurons receiving live echoes from a 40 kHz sonar system.

In the approaches to solve the problem of obstacle avoidance, there have historically been two extreme philosophical starting points in the world of robotics: rigorous path planning assuming accurate and extensive sensing [13] and fast reflexive behaviors based on minimal and unsophisticated sensing [14]. Clearly, there is an expansive world of algorithms lying between these two extremes. Path planning algorithms calculate routes between starting and goal points, requiring full knowledge of the environment. These are appropriate when a tremendous amount of relevant knowledge about the world is available. In contrast, reflexive algorithms simply steer the creature away from obstacles upon detection with very little latency. Although reflexive behaviors are well suited to a creature traveling quickly through an unknown or changing sparse environment, even mildly cluttered environments produce inappropriate movements. Philosophically, obstacles should not determine the direction in which a creature should move, rather it should simply indicate where the creature should *not* go. The question is then, "given the information about multiple sensed obstacles and the target location, how do we combine them to select a good path?" In the work presented in Chapter 3, a sonar-based obstacle avoidance algorithm, the Curved Openspace algorithm, is proposed and its implementation on a mobile robot is described. A biologically plausible neural model of the proposed algorithm using spike-timing representations is also presented.

Chapter 2: Power-Law Compression Expands the Dynamic Range of a Neuromorphic Echolocation System

2.1 Introduction

The biologically-realistic conductance model (Fig. 1) for the amplitude-independent detection of ILD introduced in [8] demonstrated that opposing synaptic conductances can respond quickly and correctly for short echolocation signals. Linear voltage encoding of amplitudes, however, limited the range of distances over which the system could operate because echo amplitudes can change up to 80dB over a change in distance from 0.3m to 3m. In that work, it was noted that the equations for the neuron membrane potential suggest that the input signals could be power-law compressed to significantly increase the input dynamic range without changing the operation. This chapter describes the successful implementation of both square-root compression and cube-root compression of the echo envelope amplitude to detect the angle of arrival of an echo from an object placed over a wide range of distances [9].

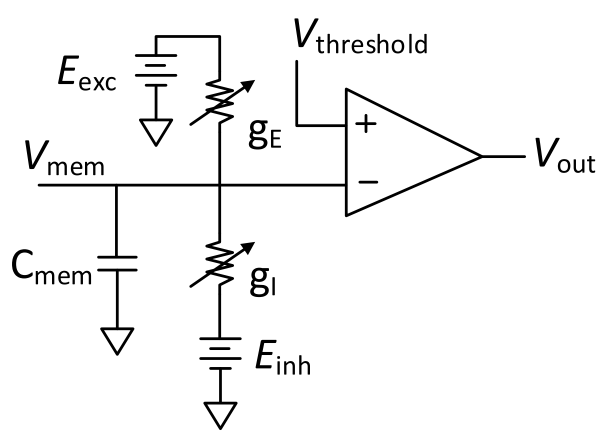


Fig. 1. The steady-state value of the membrane potential is determined by the ratio of the excitatory conductance (g_E) and inhibitory conductance (g_I).

2.2 Neuron and Compression Circuit

2.2.1 Conductance Neuron Circuit

The conductance synapse and neuron circuit (Fig. 2) is an improved version of the neuron circuit presented in [8]. The input transconductance amplifiers U_1 and U_2 receive voltages representing the compressed echo envelope amplitudes and linearly convert them to currents. These currents are then mirrored through M_1-M_2 and M_5-M_6 to the neuron circuit as I_e and I_i, which control the excitatory and inhibitory synaptic conductances, respectively. The steady-state value of V_x is decided by I_e/I_i and is transformed into a current I_m through M_7 that represents the membrane potential V_{mem} in Fig. 2.

According to the subthreshold pFET drain current equation in saturation for M_7 and M_3 in the neuron circuit (Fig. 2(a)),

$$I_\mathrm{m} = I_\mathrm{o1} \cdot \mathrm{e}^{\frac{\kappa(\mathrm{Vdd}-V_\mathrm{x})}{V_\mathrm{T}}} \text{ and } I_\mathrm{e} = I_\mathrm{o2} \cdot \mathrm{e}^{\frac{\kappa(V_\mathrm{b}-V_\mathrm{e})}{V_\mathrm{T}}} \qquad (1)(2)$$

The KCL equation at V_x is

$$I_\mathrm{e} + I_\mathrm{i} - I_\mathrm{o3} \cdot \mathrm{e}^{\frac{\kappa(V_\mathrm{x}-V_\mathrm{e})}{V_\mathrm{T}}} = \mathrm{C_m} \cdot \frac{\mathrm{d}V_\mathrm{x}}{\mathrm{d}t} \qquad (3)$$

By plugging in (1)(2), we obtain

$$\frac{-\kappa(I_\mathrm{e} + I_\mathrm{i})}{C_\mathrm{m} \cdot V_\mathrm{T}} I_\mathrm{m} - \frac{-\kappa \cdot I_\mathrm{o3} \cdot I_\mathrm{o1}}{I_\mathrm{o2} \cdot C_\mathrm{m} \cdot V_\mathrm{T}} I_\mathrm{e} \cdot \mathrm{e}^{\frac{\kappa(\mathrm{Vdd}-V_\mathrm{b})}{V_\mathrm{T}}} = \frac{\mathrm{d}I_\mathrm{m}}{\mathrm{d}t} \qquad (3B)$$

This is a first order differential equation in I_m. In this equation, the time-constant $\tau = \mathrm{C_m} \cdot V_\mathrm{T}/\kappa(I_\mathrm{e} + I_\mathrm{i})$ and the steady-state current occurs when the derivative is equal to zero

$$I_{\mathrm{m\_ss}} = \frac{I_{\mathrm{o3}} \cdot I_{\mathrm{o1}}}{I_{\mathrm{o2}}} \mathrm{e}^{\frac{\kappa(\mathrm{Vdd}-V_{\mathrm{b}})}{V_{\mathrm{T}}}} \left(\frac{I_{\mathrm{e}}/I_{\mathrm{i}}}{I_{\mathrm{e}}/I_{\mathrm{i}} + 1}\right) \tag{4}$$

This equation shows that the steady-state current is a function of the ratio I_e/I_i. The initial values of V_x and the amplitudes of I_e and I_i only affect the time (i.e. τ) for I_m to reach steady state. Equation (4) also indicates that a higher I_e/I_i ratio produces a higher steady-state current. To the extent that V_{exc} and V_{inh} are in the linear range of the input amplifiers U_1 and U_2, I_e and I_i will be proportional to V_{exc} and V_{inh}, respectively. Equation (4) can be rewritten as

$$I_{\mathrm{m\_ss}} = \frac{I_{\mathrm{o3}} \cdot I_{\mathrm{o1}}}{I_{\mathrm{o2}}} \cdot \mathrm{e}^{\frac{\kappa(\mathrm{Vdd}-V_{\mathrm{b}})}{V_{\mathrm{T}}}} \left(\frac{\beta}{\beta + 1}\right) \tag{5}$$

where $\beta = V_{\mathrm{exc}}/V_{\mathrm{inh}} = 10^{\frac{ILD}{20}}$. The neuron effectively operates as a lowpass-filtered resistor divider (i.e., synaptic conductances) where the output is a current (I_m). To generate spikes at the output, I_m is passed to a cascode transistor M_{12} and a switch M_8 that turns off I_m when I_e is too small ($V_{\mathrm{exc}} \lesssim$ 30mV) to compute the ratio accurately. When I_e is present and I_m is larger than the threshold current I_{thresh}, V_c is charged up quickly. A spike is generated when V_c is larger than the threshold voltage of the first inverter and the spike will reset V_c through M_{19}. A higher I_m increases the charging rate of V_c and thus increases the firing rate. When echoes are present, the neuron circuit fires a spike train with the firing rate as a monotonically increasing function of β.

2.2.2 Power-law Compression Circuit

Like biological neurons, the neuron circuit has a limited input dynamic range in part because the linear range of the input amplifiers is limited. As explained in the

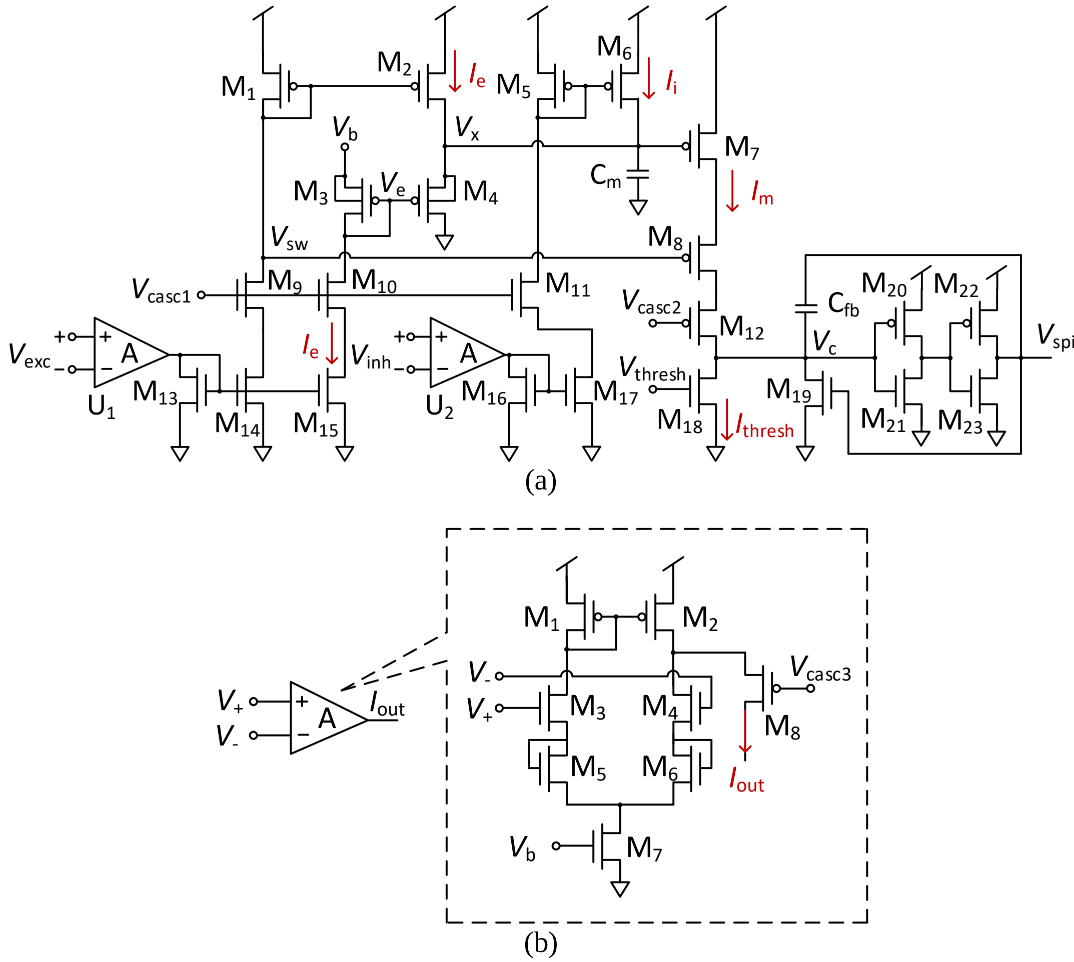


Fig. 2. (a) The conductance synapse and neuron circuit with the excitatory conductance controlled by V_{exc} and the inhibitory conductances controlled by V_{inh}. The firing rate of V_{spike} is only sensitive to the ratio of the two synaptic conductances and independent of their intensities. (b) Circuit schematic of the transconductance amplifiers U_1 and U_2 in Fig. 1(a) that transform the voltage inputs linearly to currents when the difference between the positive input V_+ and the negative input V_- is smaller than 180mV. The W/L ratios of the transistors and the values of the capacitors are shown in the Appendix.

introduction, however, power-law compression can be applied to the inputs to increase the dynamic range. The power-law compression circuit (Fig. 3) is currently implemented with discrete components and is capable of various orders of power-law compression. It has three stages: amplification, compression, and filtering.

The first-order low-pass filter U_1 has a cut-off frequency of 42.3 kHz and the amplification stage has a total gain of 65.4 dB. In the compression stage, the amplified signal is logarithmically compressed using D_{01} and D_{02} followed by a voltage division using N diodes (between U_4 and U_5) and a conversion (exponential) back to current. This current is converted back to voltage by U_5. The output voltage is proportional to the power-law compressed input. At steady state, the AC output of the compression stage can be expressed as:

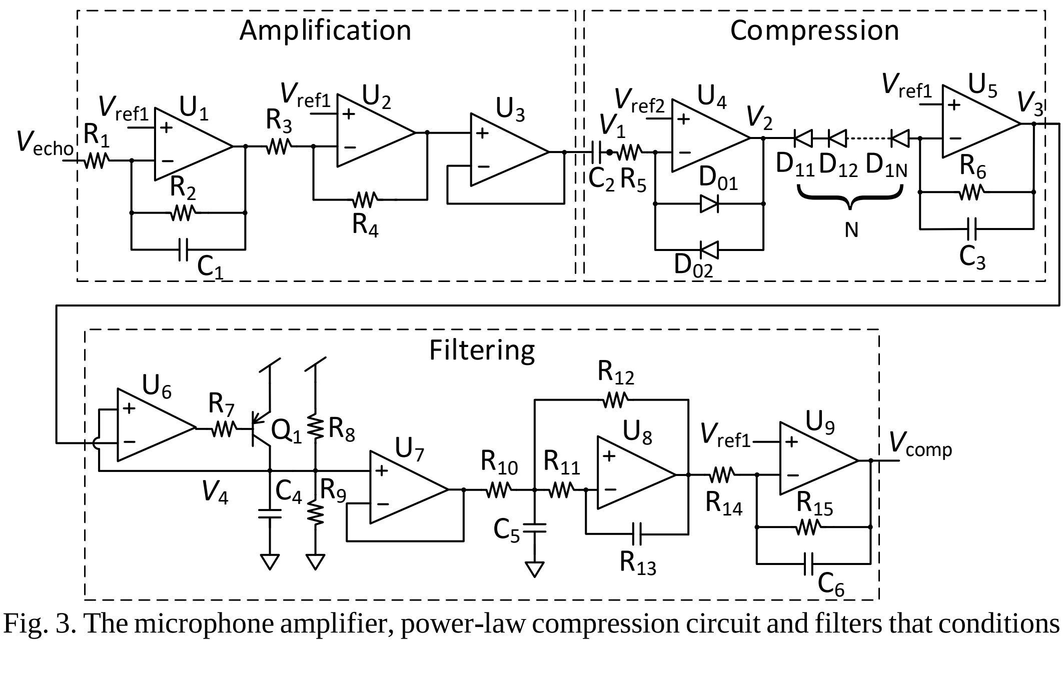


Fig. 3. The microphone amplifier, power-law compression circuit and filters that conditions the echo signals (V_{echo}) for input into the chip. The order of power-law compression can be easily changed by changing the number of diodes (N) between U_4 and U_5. The values of the parts are shown in the Appendix.

$$v_3 = I_S \cdot R_6 \left[e^{\frac{V_{ref1} - V_{ref2}}{N \cdot V_T}} \cdot \left(\frac{v_1}{I_S \cdot R_5} \right)^{1/N} - 1 \right] \tag{6}$$

where $v_3 = V_3 - V_{ref1}$ and $v_1 = V_1 - V_{ref2}$. By selecting V_{ref2} to be significantly lower than V_{ref1}, the output can be approximated as:

$$v_3 \approx I_S \cdot R_6 \cdot e^{\frac{V_{ref1} - V_{ref2}}{N \cdot V_T}} \cdot \left(\frac{v_1}{I_S \cdot R_5} \right)^{1/N} \tag{7}$$

As shown in equation (7), the output v_3 is proportional to the N-th root of the input voltage v_1. The gain in the compression stage can be adjusted by tuning V_{ref2}, R_5 or R_6.

Because the neuron circuit is designed to calculate the ratio of the echo amplitudes, this microphone interface circuit extracts the envelope of the echoes. The first block of the filtering stage is designed as a peak detector with decay. When the output V_4 is lower than the input V_3, the output of U_6 is pulled to ground, turning Q_1 on to charge V_4. The charging rate can be adjusted by adjusting R_7. When V_4 is charged to V_3, Q_1 is turned off and V_3 decays through a resistor divider R_8-R_9. Thus, V_4 follows the peak values of V_3 and decays with a fixed time-constant.

To obtain a smooth envelope, V_4 is connected through a buffer U_7 to the input of a multiple-feedback low-pass filter with a cut-off frequency of 12.1kHz and a unity gain. The first-order low-pass filter U_9, with the same cut-off frequency and a gain of -9.1dB, creates a smoother envelope and attenuates the output voltage V_{comp} into the input dynamic range of the neuron circuit.

2.3 Circuit Testing

2.3.1 Neuron Circuit Testing

The neuron circuit was fabricated in a commercially-available 3-metal, 2-poly, 0.5μm CMOS process. To demonstrate the *steady-state* performance of the neuron circuit, high-precision constant voltage inputs V_{exc} and V_{inh} were provided by two Keithley 230 voltage supplies. In the results shown in Fig. 4, each curve was tested by keeping β (defined in (5)) fixed while scaling the V_{exc} and V_{inh} levels. As expected, changing the scaling of V_{exc} and V_{inh} had little effect on the output firing rate and the neuron circuit is only sensitive to ILDs.

2.3.2 Compression Circuit Testing

To test the accuracy of the power-law compression, the ratio α is defined as $\alpha = v_{\text{comp}}/(v_{\text{echo}})^{1/\mathrm{N}}$, where N is the order of the power-law compression, $v_{\text{echo}} = V_{\text{echo}} - V_{\text{ref1}}$ and $v_{\text{comp}} = V_{\text{comp}} - V_{\text{ref1}}$. For accurate compression, α should be

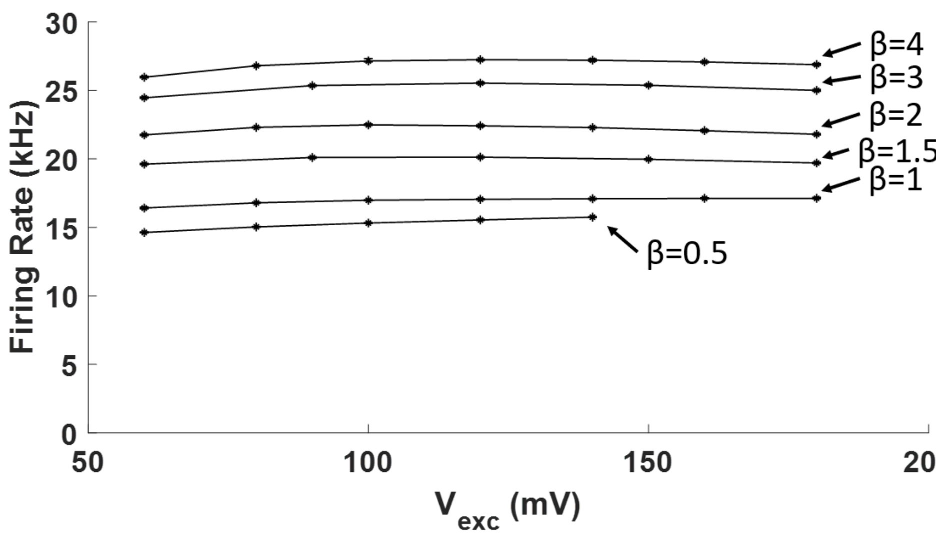
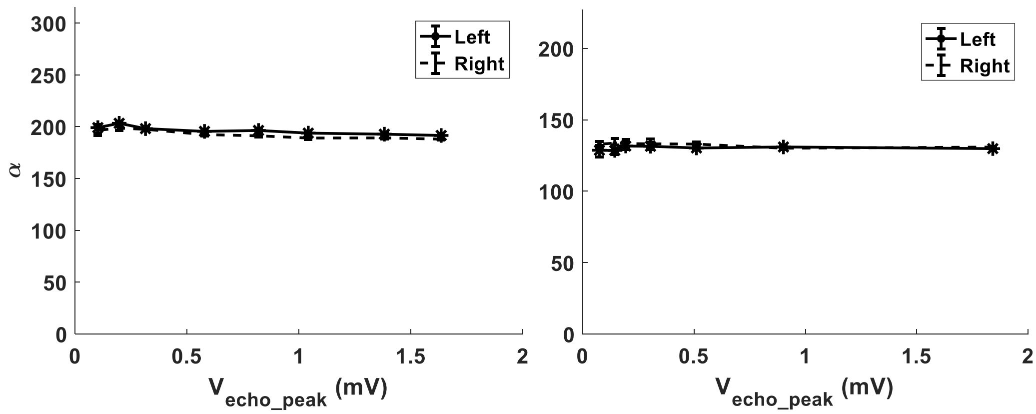


Fig. 4. Static-input testing of the neuron circuit shows that the neuron response is sensitive only to the ratio of excitatory and inhibitory inputs (β), and independent of their scaling.



(a) (b)

Fig. 6. The accuracy of the power-law compression circuit is shown by a constant α across a wide range of input amplitudes. (a) α for square-root (N=2) configuration. (b) α for cube-root (N=3) configuration.

constant across different v_{echo} values. In our testing with live echoes, only the square-root (N=2) and the cube-root (N=3) compression were tested and α is constant (within 3% deviation) across a wide range of v_{echo} for both configurations (Fig. 6).

Fig. 5. Photo of the servo-mounted, 3-D printed sonar head holding the transmitter (top-middle) and two receivers (bottom). The receivers are oriented 90 degrees apart.

2.4 Echolocation System Testing

To demonstrate the neurons with live echolocation signals, a bat-inspired narrowband sonar system was used (Fig. 7). A transmitter and two receivers were fitted into a 3-D printed mount on a servo motor (Fig. 5). The transmitter generates a 40 kHz ping of short duration (~0.5ms). The returning echoes are detected by the two 40 kHz receivers at the bottom and sent to the power-law compression circuits. The compressed envelope signal from the left excites the left neuron and the compressed envelope signal from the right inhibits the left neuron. The left neuron determines the direction for targets to the left of center. The right neuron has the opposite connectivity and determines directions for the right side. A measurement window that lasts for 225μs was created in the middle of an echo and the firing rate in this window was measured.

In the experiment, targets (5cm dia. PVC pipe) in different directions were simulated by rotating the *sonar* using the servo for repeatability. To demonstrate intensity-invariance, the target was placed at different distances while keeping the ILD fixed. The response at angles larger than 40 degrees on each side was inaccurate due to the inhibition going below the noise level and do not provide an accurate comparison.

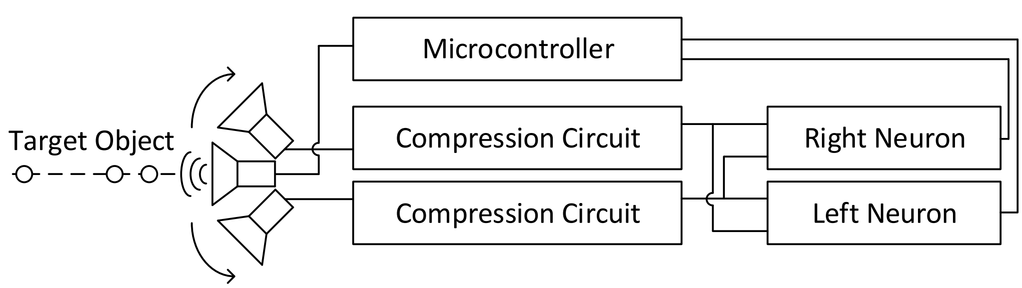


Fig. 7. The echolocation system receives the echoes from the target object and compresses in power-law. The ratios of left-to-right and right-to-left amplitudes are calculated in two synaptic conductance and neuron circuits, and their firing rates are measured in a microcontroller.

Test results of left and right neurons using different compression orders are shown in Fig. 8. The results of targets closer than 0.9m were not shown because the loud echoes saturate the amplification stage in the power-law compression circuit (Fig. 3) and produce inaccurate compression.

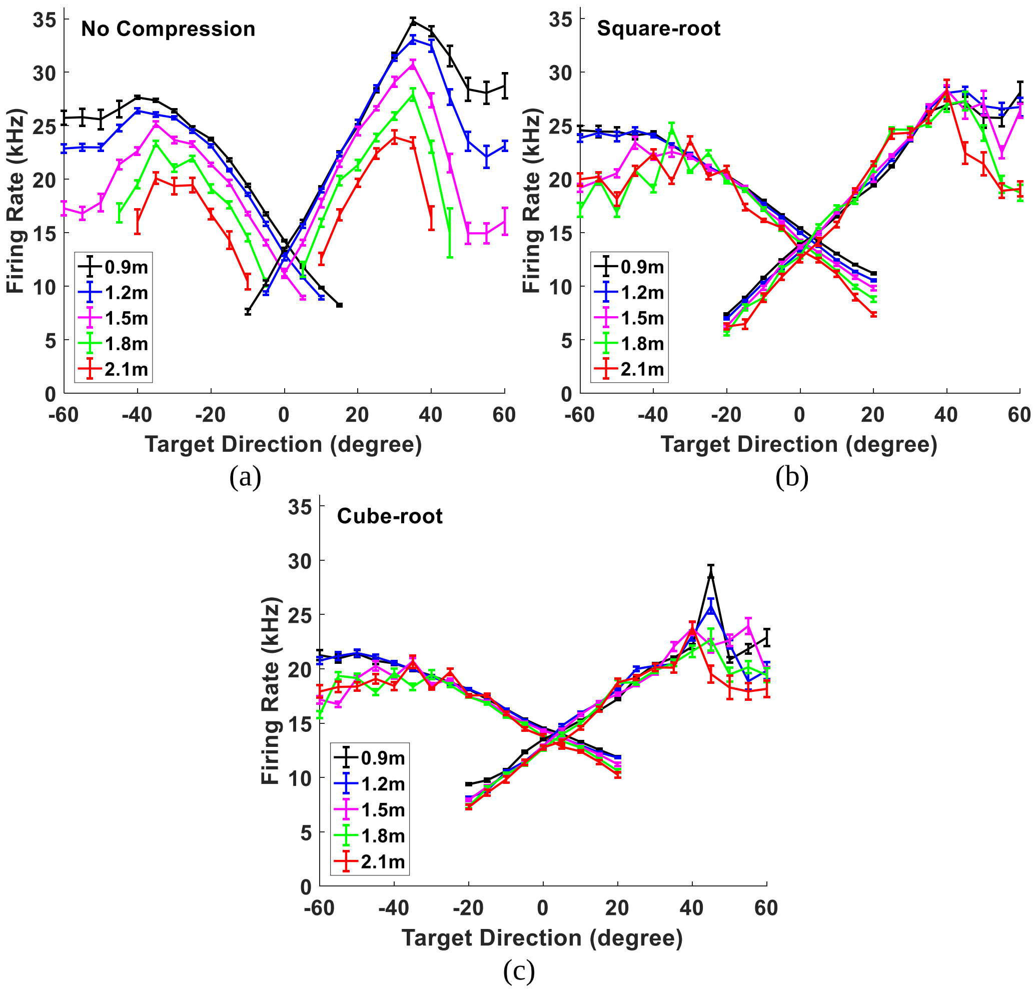


Fig. 8. The responses from the left and the right neuron show intensity-independence among certain target distances. Higher-order power-law compression shows better performance at large distances. The curves with their majority in negative degrees are from the left neuron and the ones in positive degrees are from the right neuron. (a) Responses without compression. (b) Responses with square-root compression. (c) Responses with cube-root compression.

To demonstrate how well the firing rate represents target direction using different compression orders, the average response (across distances of 0.9m and 1.2m) for different directions was used to create a mapping function to decode new test examples and estimate direction. The range of angles tested (-30 degrees to 30 degrees) was limited to where the echoes were strong enough in both receivers for accurate comparison. For target angles to the left of center, the left neuron was used and for target angles to the right of center, the right neuron was used. The decoding process was simply finding the firing rate with the mapping function and reading out the direction. Errors between the true directions and the decoded directions were calculated.

The results (Fig. 9) show that with no compression, only targets at close range produce echoes loud enough to compute the ratio accurately. Comparing the errors from the square-root and cube-root compression to the case with no compression shows that adding a power-law compression to the echoes significantly reduces the decoding errors at long distances. Overall, the echolocation system with power-law compression

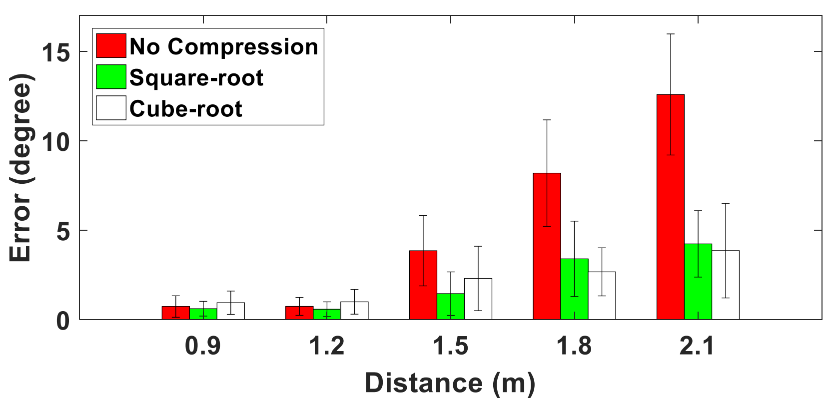


Fig. 9. The average errors between the decoded target directions and the true directions (-30 to 30 deg) with error bars showing the standard deviation. With power-law compression, the average errors are all below 4.2 degrees.

can produce good intensity-independent mapping between firing rates and directions from 0.9m to 2.1m. It is a significant improvement of the working range over prior work [8].

2.5 Discussion

The ubiquity of logarithmic compression in biological systems is widely understood to manage the wide dynamic range of sensory signals and to exploit the computational advantages that logarithms provide, particularly for comparisons by *subtraction* mechanisms. When *divisive* mechanisms are brought to bear, however, it seems that power-law compression can be more appropriate.

Although compression of incoming signals improves the input dynamic range of neural circuits, most sensory systems also have active mechanisms to increase the dynamic range further. In the echolocating bat, for example, the loudness of the outgoing pulse is modulated according to the distance of an attended object and the middle-ear muscle provides a time-dependent attenuation that can partially counteract the signal attenuation with range [12].

Chapter 3: The Curved Openspace Algorithm and a Spike-Latency Model for Sonar-Based Obstacle Avoidance

3.1 Introduction

Beyond echolocation, there are many proposed approaches to this local obstacle-avoidance problem for a robot or a vehicle with other sensing domains (e.g., vision, LIDAR, or radar.) Some of the classical approaches are vector summation based algorithms. In these approaches, obstacles create repulsive forces and the goals create attractive forces. The summation of these forces steers the vehicle into a safe path without colliding with obstacles. Some implementations are APF (Artificial Potential Fields) [15] and VVF (Vector Field Force) [16]. These algorithms are effective and interesting in their computational simplicity and mathematical elegance. They have problems such as the vehicle being trapped in local minima, although recent modifications have been proposed [17] to solve the local minima problem. As mentioned before, however, we believe that obstacles should not turn the vehicle in any particular direction but only indicate where it should not go.

Other approaches to navigation on a planar floor divide the surroundings of the vehicle into angular sectors and transform them into a polar histogram. In this histogram, the proximity of obstacles in each sector is represented and the next direction in which to steer is calculated based on the values of the histogram. These approaches include VFH (Vector Field Histogram) [18], its extension VFH+ (Vector Field Histogram Plus) [19] and the "Openspace" algorithm [20]. There are also velocity methods that map the cartesian space into the velocity space that represents the linear and angular velocities of the vehicle, then calculate the next movement of the vehicle in the velocity space. These methods are suitable for differential or holonomic vehicles since a point in the velocity space corresponds to a velocity that is directly executable to the vehicle. The algorithm calculates the next velocity in the velocity space according to an objective function that includes criteria such as speed, distances of the obstacles, or goal direction. Some implementations include CVM (Curvature Velocity Method) [21], DWA (Dynamic Window Approach) [22], and its recent extension DW4DO (Dynamic Window for Dynamic Obstacles) [23].

There are collision avoidance algorithms that rely on deliberate planning [24]. In these algorithms, an optimal or near-optimal path with collision-free routes is calculated based on an environmental map that the vehicle senses and updates. These algorithms typically assume extensive, accurate maps over which long trajectories are tested sequentially, requiring both significant computational resources and memory as

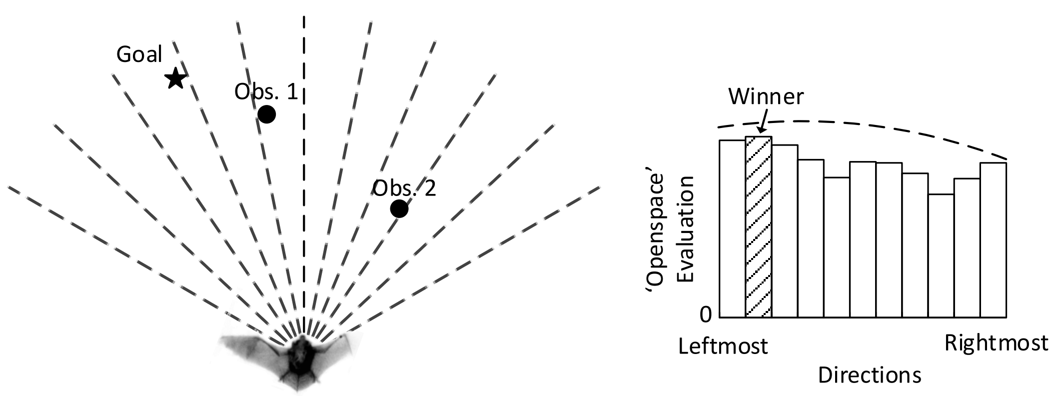


Fig. 10. The Openspace algorithm of [20]. (Left) An echolocating bat that is attempting to fly to the goal (filled star) while avoiding two obstacles (filled circle). (Right) The evaluation pattern consists of a constant plus a wide low-amplitude Gaussian (Goal input) with two dips created by the suppression from the two obstacles. A winner-take-all (WTA) function selects the direction with the highest evaluation (filled bin). The dashed line indicates the default evaluation with no obstacle present.

well as fast, accurate sensing. To address the high computational complexity of these algorithms, several optimization methods have been developed. In [25], a minimal time search algorithm with ant colony optimization is used to calculate the optimal path under communication-related constraints. The algorithm in [26] incrementally constructs a graph of trajectories while efficiently searching over candidate paths, resulting in a search tree in belief space that converges to the optimal path. Using this algorithm, aggressive flight of a fixed-wing air vehicle in an unstructured 3D environment was demonstrated [27].

The Openspace algorithm proposed in [20] provides a neuromorphic implementation of the histogram approach with a latency-based spiking neural network, giving insights into how a biological system can achieve sonar-based navigation. The Openspace algorithm seeks to find the most desirable straight-line direction of travel to avoid obstacles (Fig. 10) while a bat is traveling on a 2-D plane. It divides the area in front into an arbitrary number of steering directions, evaluates their desirability, and selects the winning direction with the maximum evaluation. It combines different inputs (a goal direction and detected obstacles) into a decision function to determine the steering decision. To physically travel precisely in the selected direction, however, a bat must be capable of making an extremely sharp turn. In practice, a flying bat will only be able to rotate gradually to the desired direction. This will produce an overshoot that will require ongoing corrections that lead to a mismatch between the selected path and the path the bat flies on. Instead of evaluating travel directions along sensory angles, our proposed algorithm, the Curved Openspace Algorithm, projects the sensory-based obstacle data into “motor coordinates” before comparing motor choices similar to the velocity approaches [21][22]. For a flying bat, this could mean selecting different turning radii that are viewed as circular arcs with different radii (Fig. 11).

Another limitation of the original Openspace algorithm is the assumption that the bat has a wide field-of-view (FOV) that covers all directions with the same effective range, thus localizing all obstacles in front of the animal with a single sonar ping. A practical sonar system, however, has a limited field-of-view whose detection range is angle-dependent due to the beam patterns of both the sonar transmitter and the

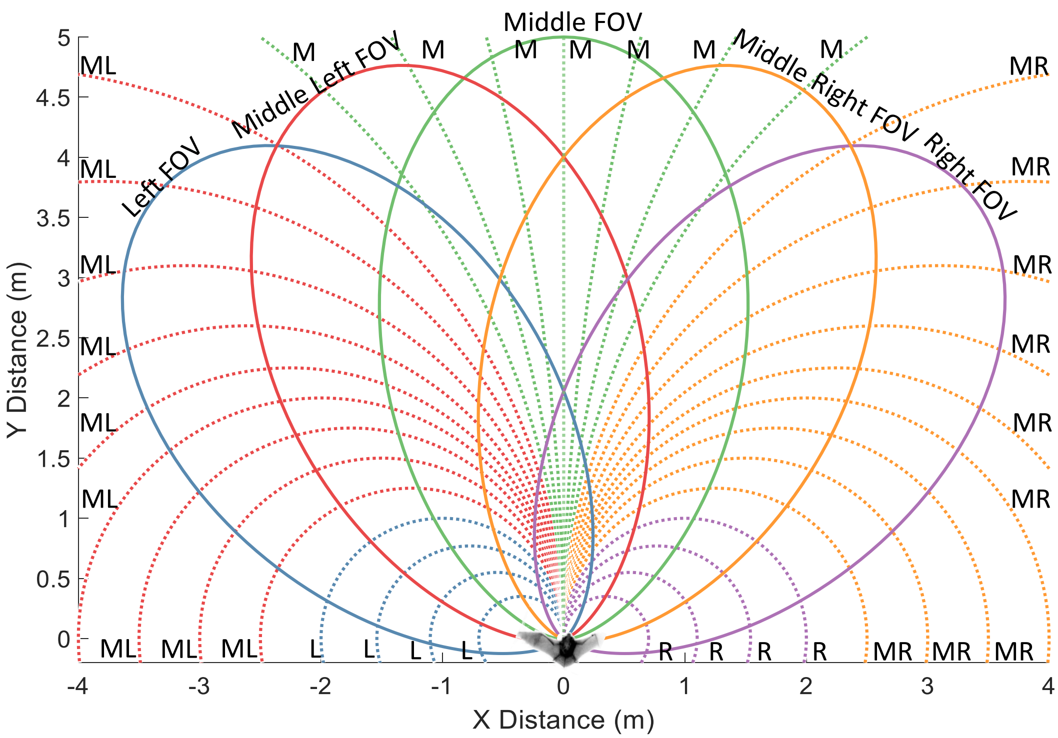


Fig. 11. An illustration of motor choices, sonar field of view (FOV) and groupings of the motor choices. Selecting different motor choices results in paths as circular arcs with different radii (dotted lines). The sonar FOV (solid lines) is ellipsoidal and the bat can turn its head to ping in different directions. 5 ping directions and 33 motor choices are shown. The colors along with the letters at the end of the paths indicate which groups (left, middle-left, middle, middle-right or right) the motor choices belong to.

receivers, resulting in an ellipsoidal FOV (Fig. 11). For big brown bats, the half-power beamwidth (the angular width of the beam pattern at the 3 dB cutoff points) of their emitted ultrasonic signal at 35-40 kHz is approximately 56 to 80 degrees [28][29]. In the simulations shown here, we used a Gaussian-shaped FOV with a standard deviation of 30 degrees, resulting in a half-maximum width of 70.7 degrees. This limited FOV means that a bat will need to scan its environment to make good steering choices, which is time-consuming and leads to choices based on old data. Although engineered systems frequently employ continuous side-to-side scanning, this is *not* observed in echolocating bats flying through a field of obstacles. For a task like steering towards a goal, if a clear path towards the goal is detected, no scanning is needed.

The Curved Openspace algorithm, like the original Openspace algorithm, combines echoes from a 2-D sonar to create an evaluation for each of the different motor actions under consideration. Due to the limited field-of-view of the simulated sonar, a memory is needed to hold these evaluation values as the sonar interrogates different directions. To avoid excessive scanning, we introduce an attentional system that integrates information about a goal direction and stored action evaluations to determine where (or if) to turn the sonar for the next ping. Instead of constructing and updating an expensive 2-D map about all the obstacles in a bat's memory, we can build a significantly simpler system by collapsing the 2-D sensory map into a 1-D evaluation memory among different motor action choices. Each motor action represents an arc of travel through the environment and its evaluation represents how risky it is. Having the evaluation memory to combine sensory inputs across head turns enables an action selection layer to select the most desirable path using a winner-take-all (WTA) function. Unlike the immediate motor response of the original Openspace algorithm with each sonar ping, the action selection (WTA) layer only provides preliminary decisions that are further processed before making a final decision. It should be noted that although the proposed algorithm is described with a sonar system, it can work with any type of input sensor as the input without changing the structure of the algorithm.

3.2 The Curved Openspace Algorithm

The purpose of the Curved Openspace algorithm is to generate a series of motor actions guided by a practical sonar system so that a bat-like agent can reach a goal location while avoiding obstacles on a 2-D plane.

The bat first localizes the obstacles inside its field of view and sends the information to the evaluation memory (Fig. 12). The evaluation memory then combines the new information with previous results to provide an evaluation of the *risk* along the trajectories of different motor choices. It suppresses the action selection layer with a Gaussian-shaped projection of inhibition. The action selection layer is also excited by the goal input and uses a WTA function to calculate the most desirable motor choice. To fine-tune the action selection, other inputs are added: winner hysteresis and ping-direction bias. Following the WTA, a "recency" condition is used to decide if the bat has recent information about the winning motor action (i.e., path). If so, the bat is confident that the path is clear and executes the winning motor action. If the sonar has not sampled the winning motor action's direction recently, the bat will not execute the winning motor choice until it has.

In this algorithm, it is assumed that the bat has a limited selection of motor choices, which includes a straight path and several curved paths (Fig. 11). The straight path represents moving straight forward, and the curved paths are circular arcs with different radii consistent with the bat flying with a fixed turning rate. The bat is only

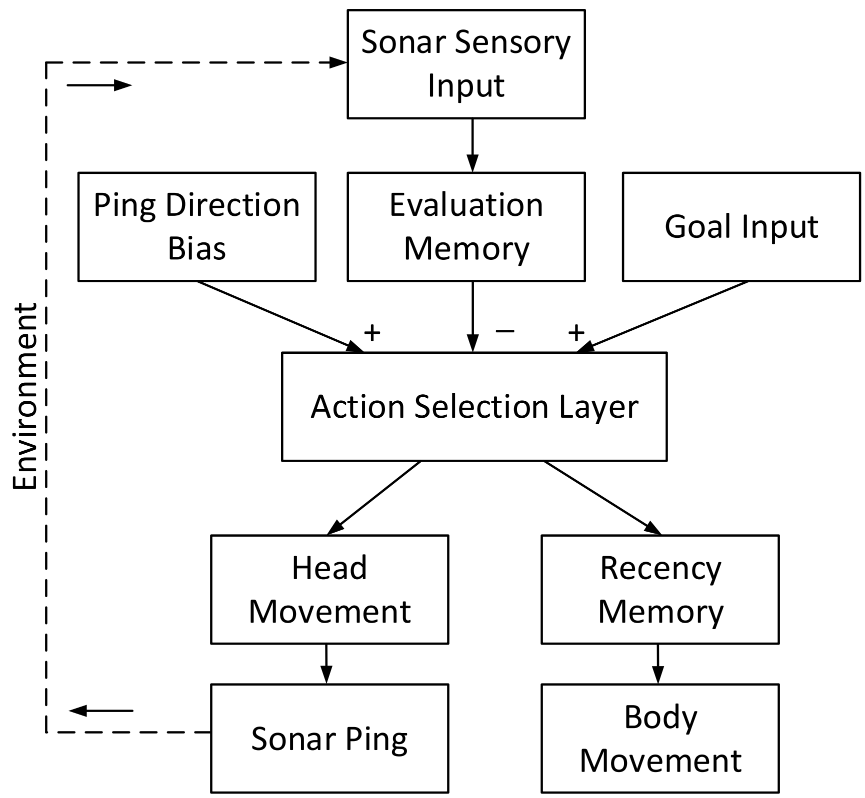


Fig. 12. The system structure of the proposed algorithm. After a sonar ping, the sensory input updates the evaluation memory. The evaluation memory, representing the risk of each path, inhibits the action selection layer. The direction of the current ping excites the action selection layer to encourage selecting paths with the most recent information. The goal input represents the direction of the goal and excites the action selection layer. The action selection layer then selects the winning path with a winner-take-all (WTA) function and the winning path drives the head movement of the bat to direct its next ping in the associated direction. In the meantime, the recency memory checks if the sensory information on the winning path is recent. If so, the winning path drives the bat to turn its body and fly on it.

allowed to ping in five fixed directions with respect to the body orientation (Fig. 11). Depending on the portion of a path that falls into the field of view of a ping direction, all possible paths are assigned to one of five view groups so that each path is associated with a single ping direction (Fig. 11).

We will refer to this algorithm as the “analog” version when comparing it to a different implementation in a later section.

3.2.1 Zone of Collision

We define the radius of the bat to be its maximum wingspan and we define the zone of collision to be a disk around the center of the bat. It has a radius of the bat plus the radius of the obstacle (Fig. 13(a)). For the purposes of simulation, the radius of the obstacle and the radius of the bat are combined into the zone of collision so that obstacles can be treated as points and the bat can be treated as a disk with a fixed radius. The bat is considered to collide with an obstacle if the obstacle passes into the zone of collision. If the minimum distance between an obstacle and a path is smaller than the radius of the zone of collision, the obstacle is defined to be a blocking obstacle since traveling on this path will eventually lead to a collision. The area where the obstacles are blocking the path is defined as the blocking area of the path. In this work, the bat is not modeled to have flapping wings and cannot perform agile maneuvers with its wings to avoid contact with obstacles as might occur in the real world.

3.2.2 Speed Control

The traveling speed of the bat is controlled according to a “speed profile” that assigns a certain constant speed to each trajectory. The speed along a path p is selected as

$$v_p = V_{\mathrm{MAX}} \cdot \gamma_p \tag{8}$$

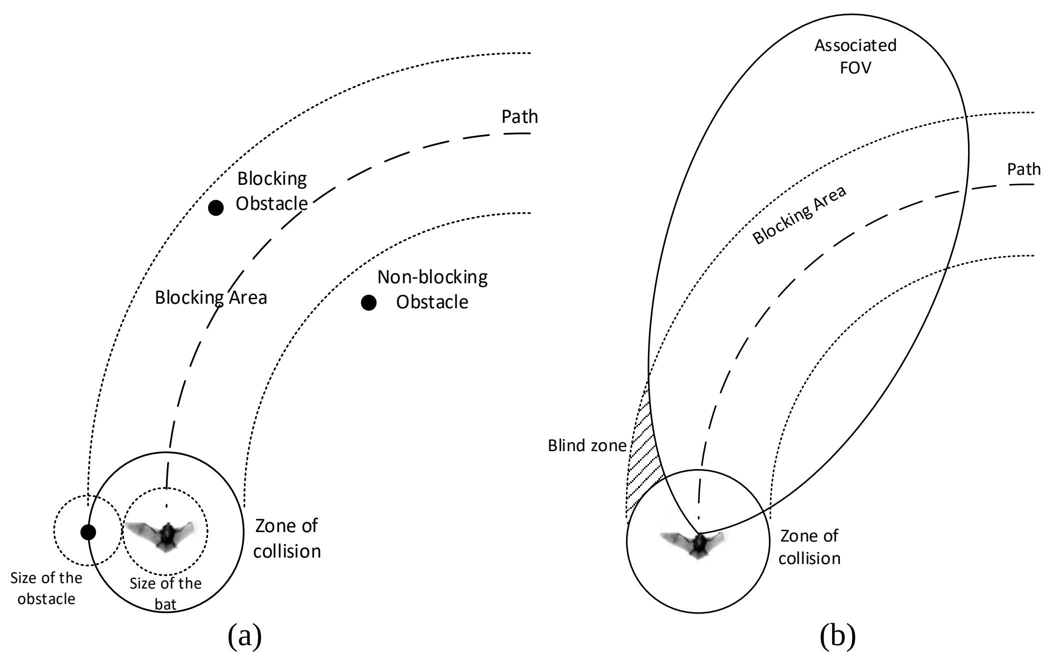


Fig. 13. (a) The definition of the zone of collision. The zone of collision is a disk around the bat with a radius of the radius of the bat combined with the radius of the obstacle. With this definition, we can treat obstacles as points and the bat as a disk. An obstacle in the blocking area (dotted line) of the path (dashed line) is defined as a blocking obstacle. (b) An example of the “blind zone” where the associated FOV cannot cover. Obstacles can disappear from the FOV and move into the blind zone while still blocking the path. Taking the maximum value between the memory and the most recent evaluation can help the bat remember the existence of the obstacle and steer away accordingly.

where V_{MAX} is a constant maximum speed and γ is the speed profile that varies from 0.0 to 1.0 for each path. Because the sonar observes different distances along each trajectory, the speed is adapted to normalize the risk for comparison across a fixed amount of time (i.e., the time between sonar pings). The straight forward path has the longest observability and thus supports the highest speed. Hence, the speed profile can be calculated as

$$\gamma_p = R_p / R_{\text{MAX}} \tag{9}$$

where R_p is the length of path p in its associated FOV and R_{MAX} is the maximum distance that the sonar system can detect. The speed profile aims to give the bat a constant reaction time on each path between a blocking obstacle becoming detectable and colliding with the bat. Intuitively, if the bat can detect obstacles further into the future on a path, it should be more comfortable with flying at a higher speed on that path. Sharp turns, however, usually require the bat to fly at a lower speed since the FOVs to the side only cover a short length of the path and any detectable obstacles in the blocking area are already close to the bat (Fig. 11). Implementing the "speed profile" for speed control reduces the chance of collision during sharp turns.

3.2.3 Evaluation Memory

A 1-D evaluation memory represents the collision risk of different paths based on new and old sensory information. Each bin of the evaluation memory corresponds to a motor action choice. The analog value of the memory represents "risk" that combines the immediacy of avoidance and the concept of the growing uncertainty of the locations of blocking obstacles due to the movement of the bat. The *immediacy* of avoiding a blocking obstacle *j* on a path *p* is represented as a function (*f*).

$$f_p(j) = E_{\mathrm{MAX}} \cdot \left[1 - \frac{r_p(j) - R_C}{R_{\mathrm{MAX}} \cdot \gamma_p}\right] \tag{10}$$

where E_{MAX} is a constant representing the maximum value of f_p. $r_p(j)$ is the distance from the bat to a blocking obstacle *j* along path *p* approximating the obstacle to be on the closest spot on path p. R_C is the radius of the zone of collision, and $r_p(j) - R_C$ represents the distance that the bat can travel along path *p* before colliding with obstacle *j*. R_{MAX} is the maximum distance that the sonar system can detect and γ_p is the speed profile for path *p*. According to Eqn. (8) and the relationship that $r_p(j) \le R_p$, the second term in Eqn. (10) is smaller than 1, resulting in a positive $f_p(j)$. The second term indicates how long the bat can travel on path *p* before colliding with a blocking obstacle *j*. A closer blocking obstacle on a path with a higher speed profile will result in a higher immediacy f_p.

To evaluate the risk of a given motor action choice *p* (a path), we use the following equation:

$$E_p(t) = \max\left[E_p(t-\Delta t) - \beta(t-\Delta t) \cdot I_d - \alpha(p) \cdot I_{inh}\,,\ \ \min\left(\sum_{j \in B_p} f_p(j)\,, E_{\mathrm{MAX}}\right)\right] \tag{11}$$

New value	Previous value	Passive decay	Directional inhibition	Summation of the immediacy of avoidance (saturating)

where

$$\alpha(p) = \begin{cases} 1, & \text{path } p \text{ is associated with the direction of the current ping} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$\beta(t) = \begin{cases} 0, & decay\ is\ halted \\ 1, & otherwise \end{cases}$$

where $E_p(t)$ is the evaluated risk for a path p, and the new risk is the maximum between a decayed memory and a saturating summation of the immediacy of avoiding blocking obstacles. $\beta(t)$ represents a passive decay whose strength I_d is a constant. The passive decay represents the increasing uncertainty of the location of obstacles with time and it happens at every time step except when the bat enters a "scanning mode" and halts the passive decay. $\alpha(p)$ is a binary value that becomes 1 when the bat sends a ping and path p is associated with the direction of the ping. It represents a directional inhibition where the strength of the inhibition I_{inh} is a constant. B_p is a set of the obstacles blocking path p and it defaults to an empty set when the bat does not send out a sonar ping at time t, resulting in a summation term of 0. This summation term represents "risk", where obstacles with higher immediacy of avoidance pose a higher risk and this risk accumulates with every blocking obstacle along the path. The risk is then saturated (using the minimum function) if it gets higher than the maximum evaluation value E_{MAX}. The final evaluation value after the update is the maximum value between the decayed old value and the new risk value computed from the objects currently being sensed. Because the FOV that a path is associated with cannot cover the whole blocking area of the path, a blocking obstacle can disappear from the current field of view while still being a threat (Fig. 13(b)). The max function is thus used as a more conservative assessment of risk between the memory and what the sensory system is detecting. Essentially, the evaluation memory is updated with the new information if a sonar ping is sent at time t, and the memory is kept with or without decay (depending on β) when there is no sonar ping.

The evaluation memory allows the bat to combine path evaluations gathered through several pings in different directions. Since the bat is still moving while gathering information, the stored memories can become outdated. By default, the values of the evaluation memory decay quickly over time to represent an increase in the ambiguity of obstacles (constant “passive decay”). When the bat changes its ping direction, the decay is temporarily halted until a motor choice has been selected and executed. In this “scanning mode” , the bat pings rapidly in the different directions of interest to minimize the distance traveled between pings and the loss of accuracy of the memories due to movement.

When a sonar ping occurs, the evaluation memories of the paths associated with the current ping direction are also inhibited. This “directional inhibition” is shown as the term $\alpha(p) \cdot I_{inh}$ in Eqn. (11). Since the maximum function will keep the previous evaluation memory if the bat did not detect any obstacles blocking the path, the path might be more open than what the memory suggests. The directional inhibition aims to reduce the risk of a path in this scenario.

3.2.4 Action Selection Layer

The evaluation memory inhibits an action selection layer of the same size, whose values indicate the *desirability* of possible motor choices. Having no obstacle on a path will give that path a low risk value, resulting in a high desirability in the WTA layer. Calculation of the desirability of each path occurs after every sonar ping and can be described by

$$D_p = D_0(p) + G \cdot e^{\frac{-(p-p_g)^2}{\sigma_g^2}} + P \cdot \alpha(p) + H \cdot e^{\frac{-(p-p_h)^2}{\sigma_h^2}} - W \cdot \sum_{p_{mem}=1}^{N} E_{p_{mem}} \cdot e^{\frac{-(p-p_{mem})^2}{\sigma_m^2}} \quad (13)$$

Constant Bias	Goal input	Ping Bias	Hysteresis	Inhibition from evaluation memory

The first term $D_0(p)$ is a positive constant bias that represents default desirability. This term allows the evaluation to remain positive even while being reduced by other terms. Besides, each path could have a different bias term to incorporate additional information about the desirability of individual paths due to actuation limits or energy considerations. This was used in the simulations described in the next section to discourage sharp left or right turns. The coefficient G is the amplitude of an additive Gaussian term with its center at p_g and a standard deviation of σ_g. This term provides a bias toward some motor actions over others due to externally provided information about a goal location. The magnitude of excitation from the goal input is much weaker in comparison to the inhibition from the evaluation memory since the goal input only serves as a bias towards the goal when different motor actions have similar risks levels The third term is an excitation term from the current ping direction that biases the bat to select the path with the most recent information when several paths have similar risks. $\alpha(p)$ is the binary value defined in Eqn. (12) and P is the amplitude of the excitation. The coefficient H is the amplitude of a Gaussian term that produces hysteretic behavior, where p_h is the path that the bat is currently traveling on and σ_h is a constant that controls the width of the Gaussian. This term prevents the bat from changing paths erratically due to noise in the measurements or occupancy calculations. The index p_{mem} refers to the bins of the evaluation memory that suppress the desirability with a subtractive Gaussian term scaled with their values

$E_{p_{mem}}$ and an inhibition weight of W. The width of the suppression is controlled by σ_m that is kept constant. After the evaluation, the action selection layer selects the path with the maximum desirability for head and body control.

3.2.5 Motor Control

The action selection layer determines the direction in which the head should be pointed (i.e., along the winning path). If a head movement is needed, it will be executed. There are two scenarios in which a path is selected. The first scenario is when the path has a low risk value after the bat pinged in its direction. In this case, the bat will likely travel along this path. The second scenario is that the bat has not pinged in the direction of the winning path, and its desirability is high because the default risk value in the evaluation memory is low. In this situation, pinging in an unknown direction can help the bat explore possible open paths.

For the bat to fly along a particular trajectory (i.e., *execute* a motor action,) the path must be chosen by the action selection layer ***and*** the sonar data evaluating that path must be fresh (a.k.a., "recent"). If the selected path is based on old data, a head movement (and a ping) is generated to obtain new data. Once both criteria are satisfied, the output of the action selection layer is allowed to drive a change in the bat trajectory. At the same time, the bat exits "scanning mode" and begins allowing its evaluation memory to decay. To keep track of data recency, each of the five ping directions has a countdown timer called recency memory that resets to a high value after a ping in its assigned direction. The recency memory of a ping direction must exceed a certain threshold for the bat to select paths associated with that direction. Note that the bat continues to travel along its prior trajectory until a new path (motor action) is selected for execution. An example of the behavior of the proposed algorithm is shown in Fig. 14.

When the desirability of the winning path is lower than a certain threshold (i.e., no acceptable paths were detected), an “emergency” is declared and the bat will do a 180-degree turn and travel in the opposite direction to the path it was traveling on before the turn. It will also direct its next sonar ping to the associated direction of the traveling path. This emergency “turnaround” maneuver is vital in the simulation for the bat to escape from the scenario where all paths are blocked by obstacles (a.k.a., a trap). Bats in the real world often perform this by using the vertical dimension to abruptly fly straight up, turn around and fly back down in the opposite direction.

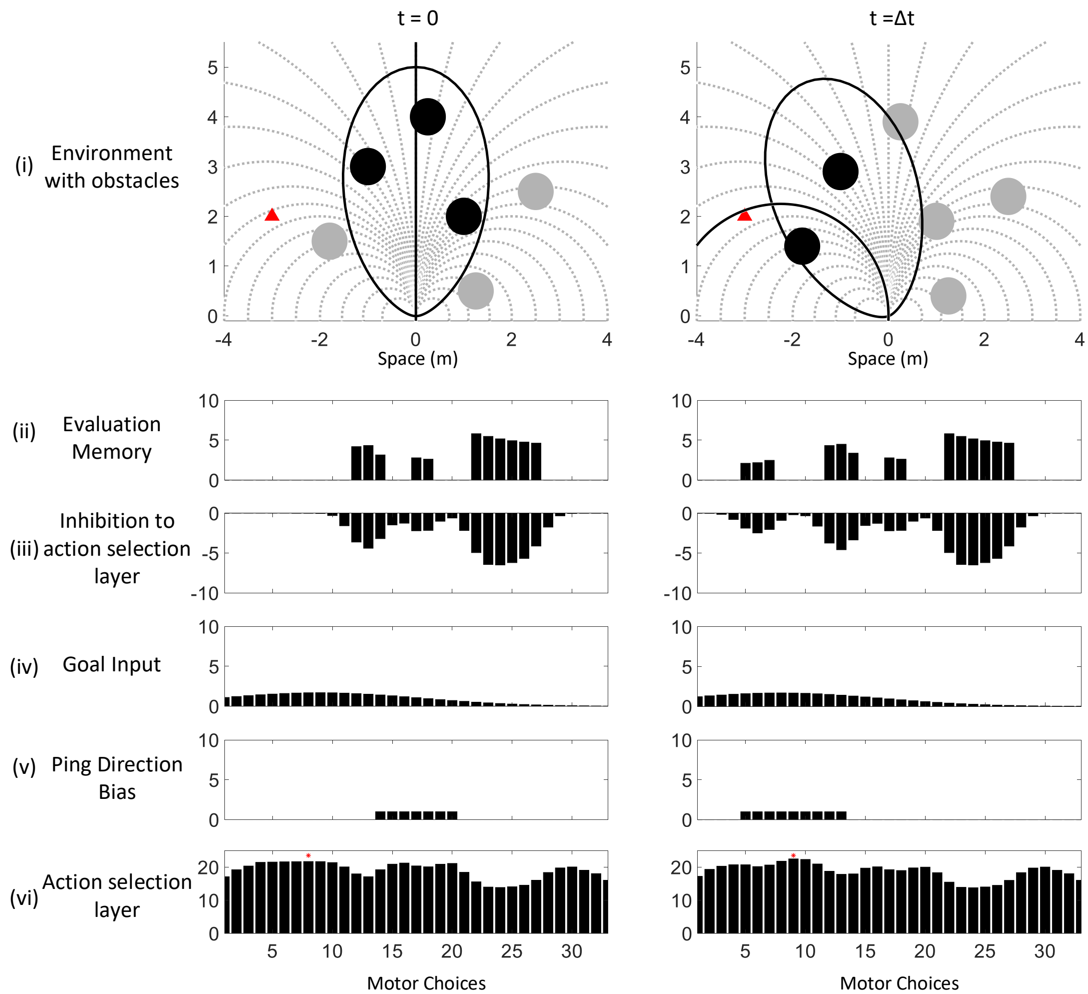


Fig. 14. An example of the proposed algorithm from sensory input to motor action selection. The environment, evaluation memory, values of the action selection layer, ping direction and motor actions in two timesteps from the simulation are shown. A bat was flying in a field with 6 obstacles (filled circle) and a desired destination (filled triangle in red). The circles of the obstacles are drawn with the size of the zone of collision to show the paths that they are blocking. The bat has a limited selection of motor choices (dotted line) and each of the motor actions corresponds to a bin in the bar graphs below. At t=0, it is assumed that the bat arrived in the environment (i) with no prior information. It is also assumed that it was flying along a straight path (solid straight line) and it directed its first sonar ping to the front (ellipsoidal FOV shown in solid line). (Continued on next page)

With the first ping at t=0, it detected three obstacles (filled circle in black) and updated its evaluation memory (ii). The evaluation memory units suppressed the action selection layer with Gaussian projection of inhibition (iii) while the goal input imposed a wide Gaussian excitation (iv). Because the bat sent out its sonar ping towards the front, the associated motor actions received excitations as shown in the ping direction bias (v). The action selection layer (vi) selected the winning path (indicated with a red dot on top of the bar), which is gated by the recency memory (not shown). Because the winning path is associated with the middle-left ping direction and the bat had not pinged in that direction recently, it turned its head to send the next sonar ping to the middle-left, did not execute the winning path and entered the "scanning mode" that halted the passive decay on the evaluation memory. At the next timestep $t=\Delta t$, the bat pinged to the middle left direction and updated the evaluation memory. The goal bias stayed the same while the ping direction bias changed to excite the paths associated with the middle left direction. The action selection layer combined the information in the same way and selected a path as the winner. Since the winning path belongs to the middle-left group and the bat just pinged in the same direction, the bat executed the winning path (solid line), exited the "scanning mode" and allowed the evaluation memory to decay. Notice that with the proposed algorithm, the bat only pinged in the directions of interest to find the most desirable path and did not need to do a full scan.

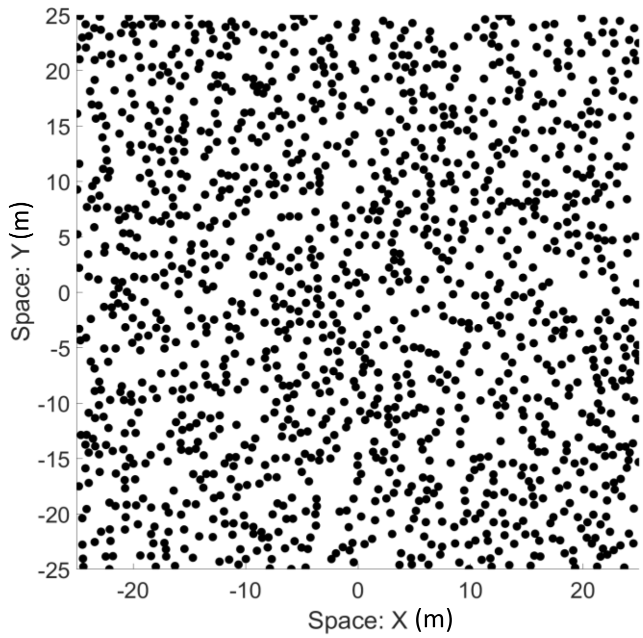
3.3 Simulation Results

We focus our simulation on the bat traveling in a field with densely placed obstacles to test the capabilities and characteristics of the proposed algorithm. 1400 identical obstacles are placed randomly in a 50m x 50m field with two-dimensional periodic boundary conditions (Fig. 15(a)). The space of the field can be mapped onto a torus to simulate a bat traveling on an infinite plane as the bat can sense and travel through boundaries. The objective for the bat is to keep traveling to the left without colliding with any obstacles. We provided a goal input to drive the bat to a goal destination at the left boundary (X=0) with the same vertical position (same Y coordinate) and updated it at each time step according to the position of the bat. When the bat crosses the left boundary, it will reappear at the right boundary with the same orientation and speed before the crossing. An example of the traveled path in a simulation is shown in Fig. 15(b) and it shows that the bat visited most of the viable paths instead of repeatedly traveling on a few paths.

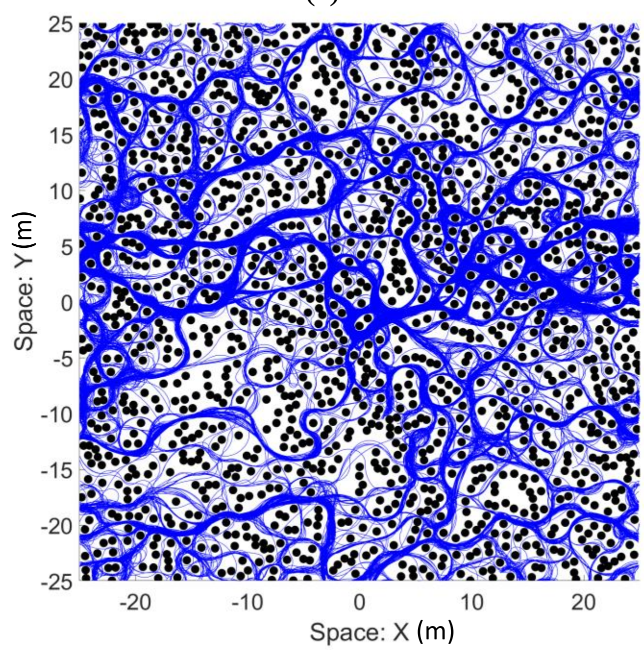
In the simulations, the bat was able to sense the accurate locations of the obstacles whose centers were in its current FOV. Occlusion of distant obstacles was not simulated. The FOV function (detectable distance vs. relative angle to the head direction) used in the simulation is a Gaussian distribution with $\sigma = 30°$ with a maximum distance $R_{\mathrm{MAX}} = 5\ m$, resulting in a half-maximum width of $70.7°$. For big brown bats, the half-power beamwidth (the angular width of the beam pattern at the 3 dB cutoff points) of their emitted ultrasonic signal at 35-40 kHz is $56°$ to $80°$ [28][29]. Although the FOV of a sonar system is not equivalent to the beampattern of the emitted sound, we approximated them to be similar and designed the FOV with a similar shape.

The bat emitted sonar pings at a maximum rate of 5 Hz in the simulation with the exception in section 3.4 where the maximum sensing rate is increased to test its influence on the performance. Big brown bats can send out sonar pulses to sense obstacles at a rate of up to 90 Hz [1] and turn their heads to ping in a different direction in less than 30 milliseconds [15]. We used a much lower maximum sensing rate in the simulation to push the limits of the algorithm and test its capabilities. It is assumed that there are no errors in the steering action and the bat was able to follow the winning path exactly.

To evaluate the performance of the algorithm, the number of unique obstacles that the bat detected but did not collide with is counted as the number of avoided obstacles. Only unique obstacles were counted because the bat could be trapped temporarily in a local area and the obstacles around the bat should not be counted more than once in this scenario. Whenever the bat crossed the left boundary, the bat entered a new environment and all the obstacles could be counted again. The simulation ends if one of the three scenarios happened: (1) the bat collided with an obstacle; (2) the bat has not crossed the left boundary in a long time (2500 seconds) suggesting that it is trapped; (3) the simulation has reached a time limit (250,000 seconds). In the second and third scenarios, the simulation restarts with a different random seed, and the number of avoided obstacles is accumulated. Whenever the bat collided with an obstacle, the accumulated number of avoided obstacles before the collision is recorded and is considered as a sample.



(a)



(b)

Fig. 15. (a) An example of the field with randomized locations of obstacles. 1400 obstacles (black circles) were placed in a 50m x 50m area. The size of the obstacles shown in the figure is the size of the zone of collision (combination of the sizes of the bat and the obstacle). (b) The paths that the bat traveled (blue line) on the field. Although open areas were traveled more often by the bat, most of the viable paths were visited.

3.3.1 Density of Obstacles

To test how the proposed algorithm performed with different densities of obstacles in the field, simulations were run with different numbers of obstacles ranging from 500 to 1700. Because the field with periodic boundary conditions is equivalent to an infinite space, changing the number of obstacles in the 50m x 50m field is equivalent to changing the density of obstacles. The simulation results are shown in Fig. 16. As expected, the performance increased as the density decreased.

3.3.2 Desirability Function

To test the effect of different terms in the desirability function (Eqn. (13)) under different maximum speed settings, we compared the performance between a control configuration of the algorithm across different speed settings (i.e. control group), a

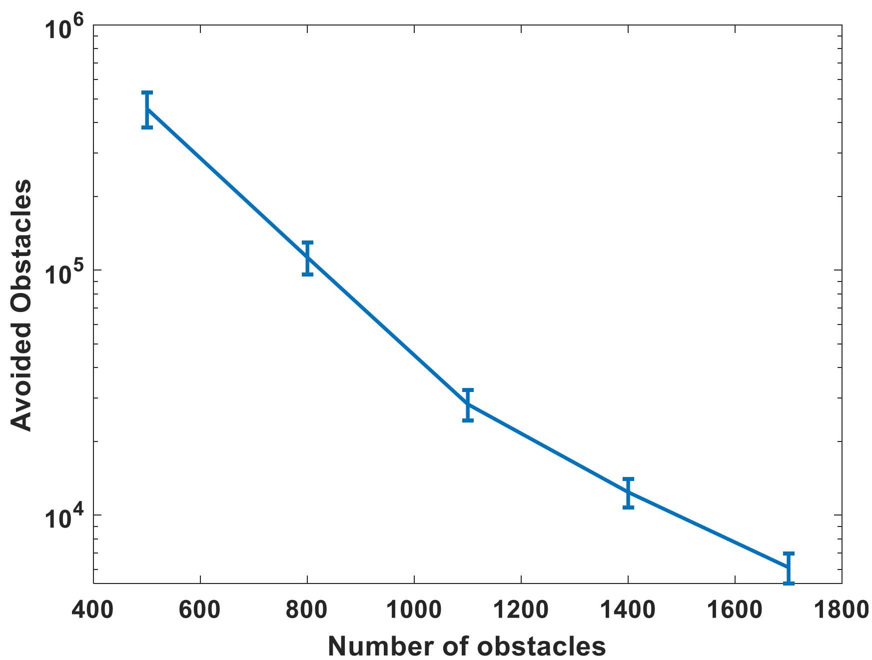


Fig. 16. The performance of the proposed algorithm as a function of the number of obstacles in the field. The number of avoided obstacles increased, approaching infinity, as the number of obstacles decreased.

configuration without WTA hysteresis and a configuration without ping direction bias (Fig. 17). The control simulations used all the terms of the desirability function described in Eqn. (13), while the group without WTA hysteresis excluded the hysteresis term ($H = 0$) and the last group excluded the ping direction bias term ($P = 0$). The average number of avoided obstacles is calculated based on 200 samples under different maximum speed (V_{MAX} in Eqn. (8)) settings and the 95% confidence interval for each configuration is shown as error bars in Fig. 17.

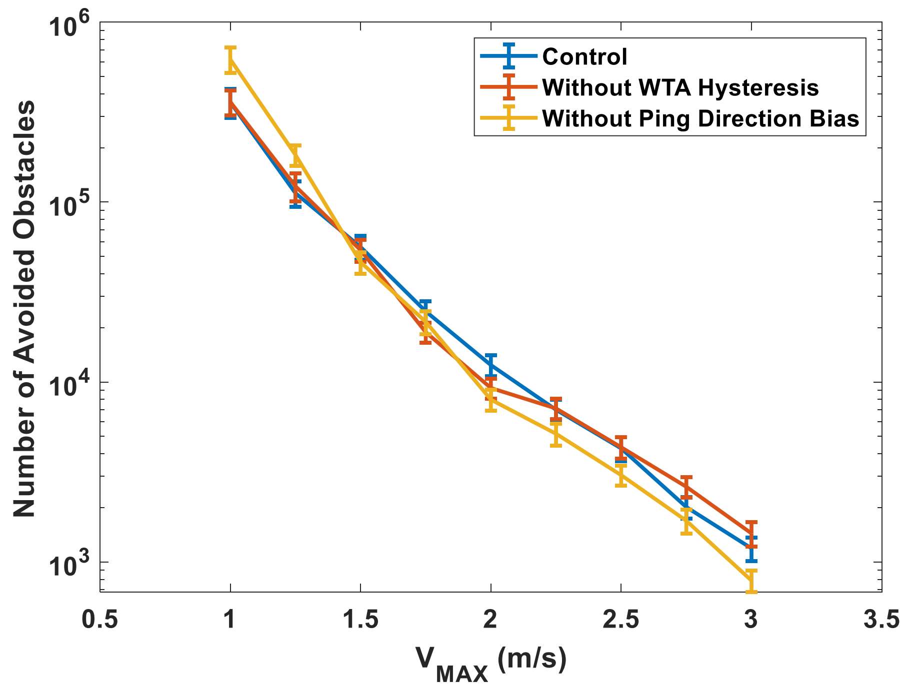
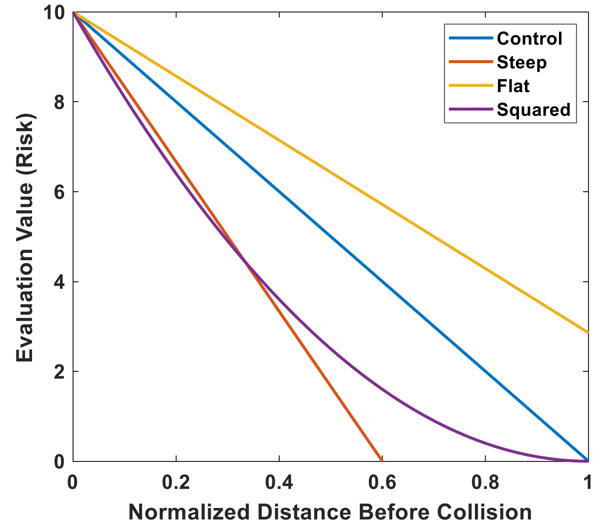


Fig. 17. The simulation results of the effect of different terms in the desirability function under different maximum speed (V_{MAX}) settings. The control configuration included all the terms described in Eqn. (13) while the configuration without WTA hysteresis had H=0 and the group without ping direction bias had P=0. The group without WTA hysteresis had similar performance at most of the speed settings compared to the control group, while the group without ping direction bias had significantly better performance at lower speeds ($V_{\mathrm{MAX}} \leq 1.25\, m/s$) and significantly worse performance at higher speeds ($V_{\mathrm{MAX}} \geq$

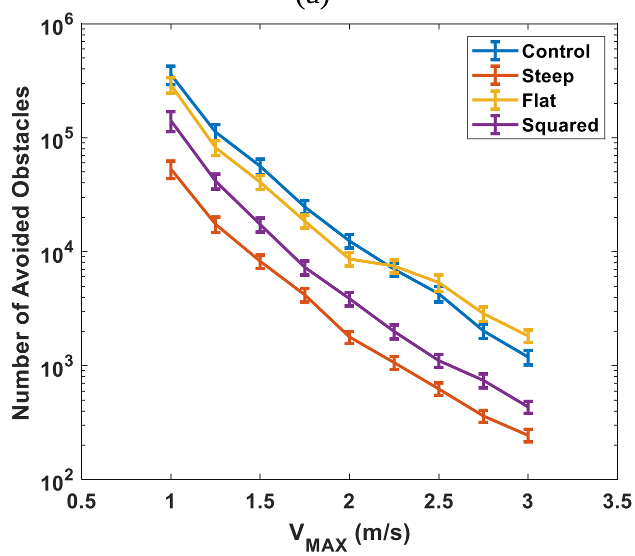
The group without WTA hysteresis had significantly worse performance ($p<0.05$) only at $V_{\text{MAX}} = 2\ m/s$ compared to the control group. Hysteresis is generally introduced in a WTA layer to prevent the winner from oscillating between a few similar candidates due to noise in the system. We believe that one of the main reasons why WTA hysteresis did not have a significant impact on the performance is because sensory or other noise is not included in the simulation.

A comparison of performance shows that the group without ping direction bias had significantly better performance ($p<0.05$) than the control group when $V_{\text{MAX}} \leq 1.25\ m/s$ and has significantly worse performance ($p<0.05$) when $V_{\text{MAX}} \geq 2\ m/s$. As mentioned in section 3.2.4, the function of the ping direction bias is to encourage the bat to select the path with the most recent information when several paths have similar risks. At low speeds, the evaluation memory of the paths from different ping directions is relatively accurate and adding the ping direction bias frequently prevents the bat from selecting the actual "best" path, thus dropping the performance. At high speeds, however, the evaluation memory may be wildly inaccurate and selecting a slightly better path according to the memory could be more dangerous than selecting a "good" path with recent and accurate information. The results suggest that features in the desirability function provide benefits in different situations and adding a speed-dependent controller on the weights of different features could help improve performance.

3.3.3 Immediacy Function



(a)



(b)

Fig. 18. (a) The shape of different immediacy functions tested in the experiment. (b) The performance of the algorithm with different immediacy functions. The results from the "Steep" group and the "Squared" group showed that reducing the significance of distant obstacles negatively impacted the performance across all speed settings, and thus indicated that distant obstacles were important to the decision-making process of the algorithm. The comparison between the control group and the "Flat" group showed that increasing the significance of distant obstacles could be beneficial at higher speed.

We tested and compared different functions representing the immediacy of avoiding an obstacle. As a reminder, the immediacy of avoiding a blocking obstacle *j* on a path *p* is represented as a function (*f*).

$$f_p(j) = E_{\mathrm{MAX}} \cdot \left[1 - \frac{r_p'(j)}{R_{\mathrm{MAX0}} \cdot \gamma_p}\right] \tag{14}$$

where E_{MAX} is a constant representing the maximum value of f_p and is set to 10 for all the different configurations in this experiment. $r_p'(j)$ is the distance that the bat can travel along path p before colliding with a blocking obstacle j. The distance is then normalized with the speed profile γ_p to represent the time before colliding with the obstacle.

The results from this experiment are shown in Fig. 18. The control group had the same function of Eqn. (14) with $R_{\mathrm{MAX0}} = R_{\mathrm{MAX}} = 5$ and it decreased linearly with the distance to the obstacle normalized by the speed profile γ_p. Because $R_{\mathrm{MAX0}} = R_{\mathrm{MAX}}$, the immediacy function decreased to 0 when the normalized distance $r_p'(j)/\gamma_p$ is 1, as described earlier. The "Steep" group had $R_{\mathrm{MAX0}} = 3$, thus creating a steeper linearly-decreasing function compared to the control group. The "Steep" function was rectified to 0 when it dropped below 0, which means that more distant obstacles were simply ignored. The "Flat" group had $R_{\mathrm{MAX0}} = 7$, resulting in a flatter function that included the contributions of a blocking obstacle even when it was distant. As a tradeoff, the differences between obstacles at different distances were reduced. The "Squared" group had a function as $f_p(j) = E_{\mathrm{MAX}} \cdot \left[1 - r_p'(j)/\left(R_{\mathrm{MAX}} \cdot \gamma_p\right)\right]^2$, which emphasizes close obstacles without completely ignoring distant obstacles.

As shown in Fig. 18(b), the number of avoided obstacles in the "Steep" group was significantly lower than all other groups across all speed settings, followed by the "Squared" group with the second-worst performance. It showed that throwing away or reducing the significance of the information about distant obstacles severely reduced performance, indicating that the proposed algorithm could make good use of this information in its decision-making process. The "Flat" performed significantly worse ($p<0.05$) when $1.5\ m/s \leq V_{\text{MAX}} \leq 2\ m/s$ but performed significantly better ($p<0.05$) at speeds higher than $2\ m/s$. Distant obstacles present a greater threat at higher speeds, thus the steering decisions could benefit from emphasizing them. The results suggest that adjusting the evaluation function accordingly at different speed settings could improve performance.

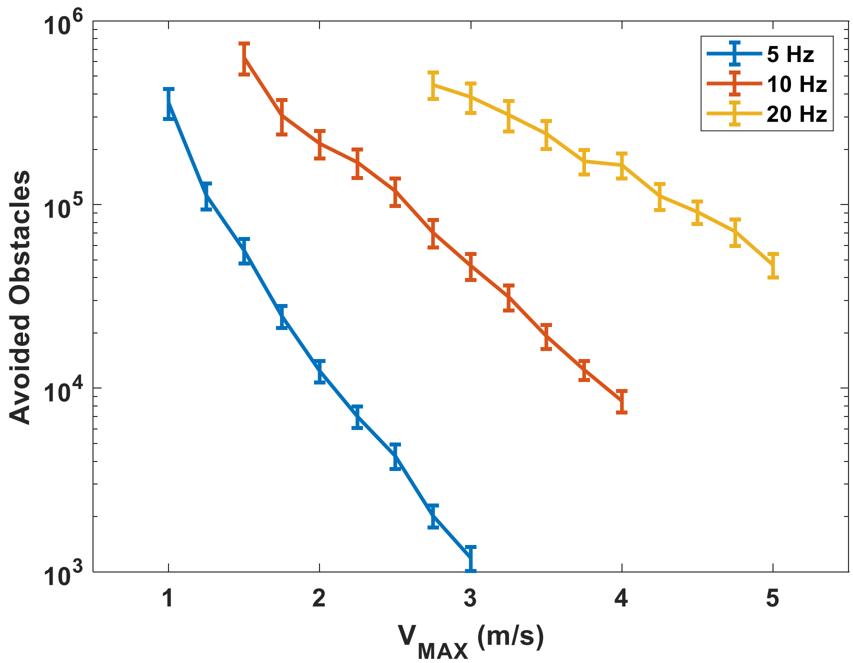


Fig. 19. The effect on the performance from changing the maximum rate of sonar pings. The performance not only was significantly increased with increased rate of pings but also declined more slowly with the increase in the maximum speed.

3.3.3 Maximum Ping Rate

In the previous experiments, the maximum ping rate is set to be 5 Hz, meaning that the bat could only detect obstacles five times per second. As mentioned before, echolocating bats can emit outgoing sonar pings up to a rate of 90 Hz [1]. In this experiment, we increased the maximum rate of sonar pings to 10 Hz and 20 Hz to test its effect on the performance.

As shown in Fig. 19, the performance improved significantly with increased ping rates. Increasing the ping rate also helped their performance decline more slowly with the increase in speed settings, indicated by a lower slope of 10 Hz and 20 Hz curves compared to that of the 5 Hz curve. A higher rate of pings allowed the bat to travel at a much higher speed while maintaining its performance.

3.4 Robot Implementation

We have implemented the analog version of the Curved Openspace algorithm on a mobile robot with a car-like steering mechanism (Fig. 20). The 3-D printed sonar head is mounted on a servo motor at the front of the vehicle and has three vertically stacked sonar transducers, each facing a different direction. The transducers are custom-modified MaxBotix® sonar transducers (piezo) that act as both a speaker and a microphone. They resonate at 40 kHz and will only detect signals near this frequency. The top transducer points 45° to the right (from the head's point of view), the middle transducer points straight forward, and the bottom transducer points 45° to the left. To collect sensory data, the middle transducer sends out a sonar ping and the echoes reflected from objects are measured by all three transducers. The transmitted beam has a half-power beamwidth of about 60°. The distances to objects are measured using the time of flight of their echoes, and the direction of each object is computed using the relative amplitudes of the sonar echo. Following an outgoing ping, the highly resonant transducers will produce a decaying signal that continues for some time, interfering with amplitude measurements of echoes from nearby objects. Although objects can be *detected* within this signal, amplitude measurements for localization are not practical. This is the near-field blind zone of the sonar system, and its range can be reduced by

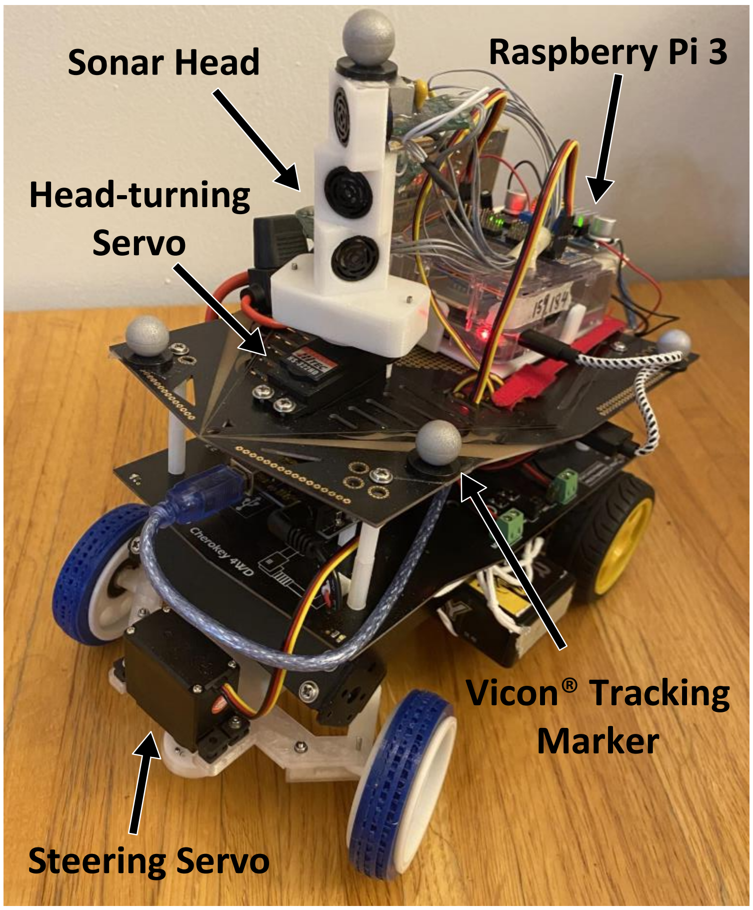


Fig. 20. The mobile robot where the analog version of the Curved Openspace algorithm is implemented. The robot consists of a sonar head with three sonar transducers mounted on a servo motor, a Raspberry Pi 3 running the algorithm, an Arduino Uno board controlling the rear wheel DC motors, a servo motor controlling the front wheels with an Ackermann steering geometry, an Adafruit 16-channel servo driver, and a lithium-ion polymer battery.

lowering the intensity of the outgoing ping when the robot is encountering close obstacles.

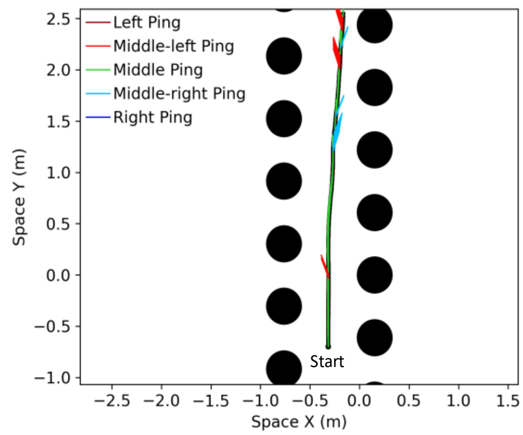
The custom sonar boards output a logarithmically-compressed envelope signal of echoes as an analog voltage. For each echo, the peak voltage on the three transducers is recorded by a microcontroller analog-to-digital converter (ADC). A radial basis function (RBF) network is then used to map the echoes to angles in the horizontal plane at 1-degree increments. The RBF network was trained using a dataset consisting of 910 echoes from 91 different angles and 10 intensities (to simulate different ranges).

The trained sonar RBF network and the analog version of the Curved Openspace model are implemented in Python on a Raspberry Pi 3 mounted on the back of the robot. The Raspberry Pi allows for wireless operation via WiFi and coordinates the commands to a servo controller board (for sonar head pointing and steering) and the powered rear drive wheels. The servo motor in the front of the vehicle controls the turning rate of the vehicle through an Ackermann steering geometry. A given steering angle thus translates to a specific turning radius of the vehicle and a lookup table is used during operation. Because of the limited resolution of the steering motor, the total number of motor actions was reduced from 33 to 15 in the robot implementation. The algorithm controls the speed of the vehicle by adjusting the power delivered to the rear wheel DC motors using pulse-width modulation.

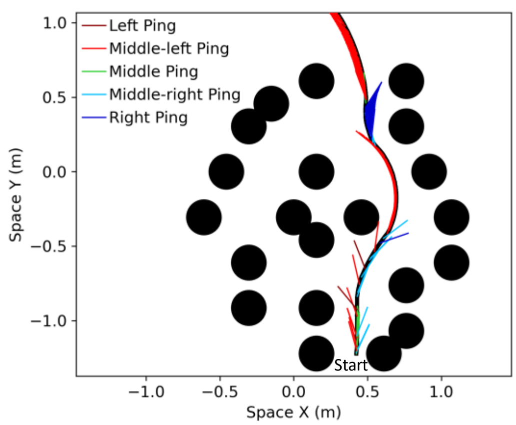
The program initially sends out a sonar ping in the middle direction, waits for a period of time to collect sonar echoes, and uses the RBF network to localize the detected obstacles. The goal input is provided in the straight-forward direction initially and it can change according to the movement of the robot so that it guides the robot towards the initial goal direction. The risk values and the desirability of 15 different paths are then calculated and a winning path along with the next ping direction is executed. After that, the program waits for the turning of the head to finish before sending out a new sonar ping and starts the cycle again. If no head movement is required, a sonar ping is sent in the current ping direction immediately. Because of the limited rotational speed of the head-turning servo, the ping frequency varies significantly depending on the delay due to the required head movements, ranging from 3Hz when the head needs to turn from one end to another, to 12Hz when no head movement is required.

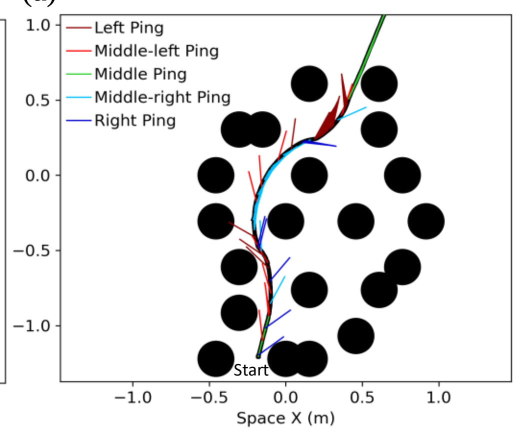
The mobile robot was tested in different forests of obstacles and its positions were recorded in 3-dimension using a Vicon® marker-based, visual tracking system. The robot path and ping directions of the robot in three different example forests are shown in Fig. 21. The first configuration (Fig. 21(a)) is a corridor formed by two lines of obstacles. When the robot did not detect any obstacles near the middle path, it did not need to turn its head to other directions because the goal direction is in the middle and the middle path was the winner. As a result, most of the time the robot kept pinging down the center. The sonar pings to the side were caused by errors in the sonar RBF network that sometimes incorrectly localizes side obstacles to the center.

The second forest was composed of 21 obstacles placed in a 1.8m x 1.8m area. The area around the starting point of the robot is more cluttered, so the robot pinged around and explored different paths more often. In the second half of the run, the robot entered a more open area where it mostly pinged in the direction that it was traveling in. A similar result can be observed in a denser forest where 21 obstacles were placed in a 1.4m x 1.8m area (Fig. 21(c)).



(a)





(b) (c)

Fig. 21. The path and ping directions of the robot during runs in three different forests. The obstacles (5 cm diameter PVC poles) are shown as black circles and the size of the circle is the size of the zone of collision (size of the robot plus the size of the PVC poles). The path the robot took is shown as a solid black line and the ping directions are shown as solid colored lines pointing at the corresponding ping directions. (a) Robot running down a corridor formed by two lines of evenly spaced obstacles. The ping directions in this run were mostly down the center (green solid lines) (b)(c) Robot navigating in two forests with different densities. The robot explored other paths more often by pinging in different directions in more cluttered areas, whereas it mainly pinged in the direction of its current path in the more open areas.

3.5 Spike-Latency Neural Model

Although the algorithm presented above can be implemented using large populations of spiking neurons to simulate (noisy) analog signal representations, spike-timing-based signal representations often suggest very different neural implementations with far fewer neurons. In this section, a spiking neural model of the Curved Openspace algorithm using spike-timing to represent signal values is described. As shown in Fig. 22, its structure consists of four main layers: a sensory layer that encodes the 2-D locations of obstacles, a memory layer that integrates and stores the sensory information associated with different paths ("Evaluation Memory"), an "Action Selection" layer that uses a "race-to-first-spike" winner-take-all (WTA) mechanism to select a path, and a "Motor" layer that implements the body steering decision and head movements. Inputs to the neural model also include ping directions (head direction), a global reset signal, goal direction input, and a ping onset signal.

The time at which a neuron fires a spike following an outgoing echolocation pulse is affected by many variables. Echolocation is foundationally based on the time-of-flight of sound to determine the distance to objects, with the closest objects generating echo signals first, producing a natural temporal coding scheme. Additionally, echoes from a given object are louder if it is closer. Interestingly, neurons commonly exhibit shorter latency responses to larger magnitude signals. In this model, following each sonar ping, sensory neurons will fire spikes with latencies that reflect the immediacy of avoiding any detected obstacles. The evaluation memory units integrate the sensory information on different paths and store the integrated information as spike latencies using a delay line and an array of latency memory units, which will be described in detail in section 3.5.2. Because the stored signals are spike latencies, during readout, the evaluation memory units must be able to re-generate output spikes with the same latencies without sensory input. The evaluation memory sends out spikes with the stored latencies to the action selection layer, where the net desirability of

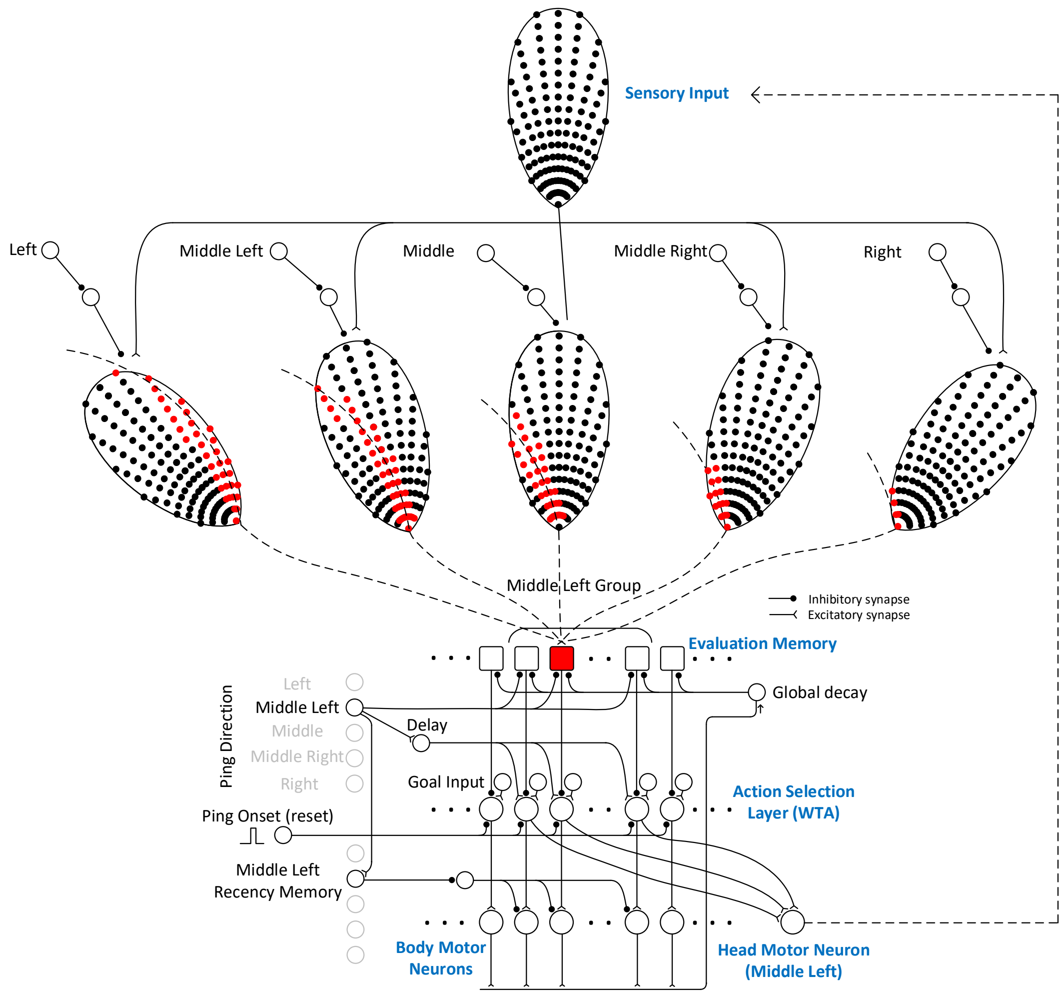


Fig. 22. The neural model of the Curved Openspace algorithm. It mainly consists of four layers: a 2-D sensory layer encoding the locations of obstacles, a memory layer storing the "risk" of different paths, an action selection layer that uses a "race-to-first-spike" WTA mechanism to select the winner and a motor layer that controls the body and head movements. Only a portion of the connections in the group of middle-left ping direction are shown in this example for clarity.

different paths is compared, and a "winner" is selected. The neuron that fires the first spike in the action selection layer has the highest desirability and inhibits all other neurons in the same layer to prevent them from firing. The spike from the winning neuron is then sent to the motor neurons that will orient the sonar head for the appropriate ping direction (head motor neurons). The spike is also sent to the body motor neurons that produce the turn rate needed for the corresponding path. The body motor neurons, however, will only be activated if the recency memory of the associated ping direction is active. The gating from the recency memory is implemented with a disinhibition mechanism. The following sections describe each layer in detail.

3.5.1 Sensory Layer

In this neural implementation, the sensory layer (Fig. 23) uses a 2-D head map to represent the locations of obstacles and converts the information from the head frame to the body frame with several body maps. The neurons in the head map make strong one-to-one excitatory connections to the neurons with the same positions in all the body maps. The number of body maps is the same as the number of possible ping directions and there is an inhibitory interneuron for each body map that strongly inhibits all the neurons in the map when it fires a spike. In this model, we define the spike latency as the time between the outgoing sonar ping and the first spike from a neuron. When the bat pings and detects an obstacle, the neuron at the corresponding location in the head map fires a spike with a latency close to the latency of the echo. Depending on the direction of the sonar ping, only one body map will be selected as active by inhibiting all other body maps using the inhibitory interneurons. A spike from the head map excites the corresponding neuron in the active body map and causes it to fire a spike immediately. The spikes from the active body map represent obstacles at certain locations in the bat's body frame. Because Curved Openspace algorithm uses the distance along a curved path to evaluate the risk value (Eqn. (10)), the latencies of the

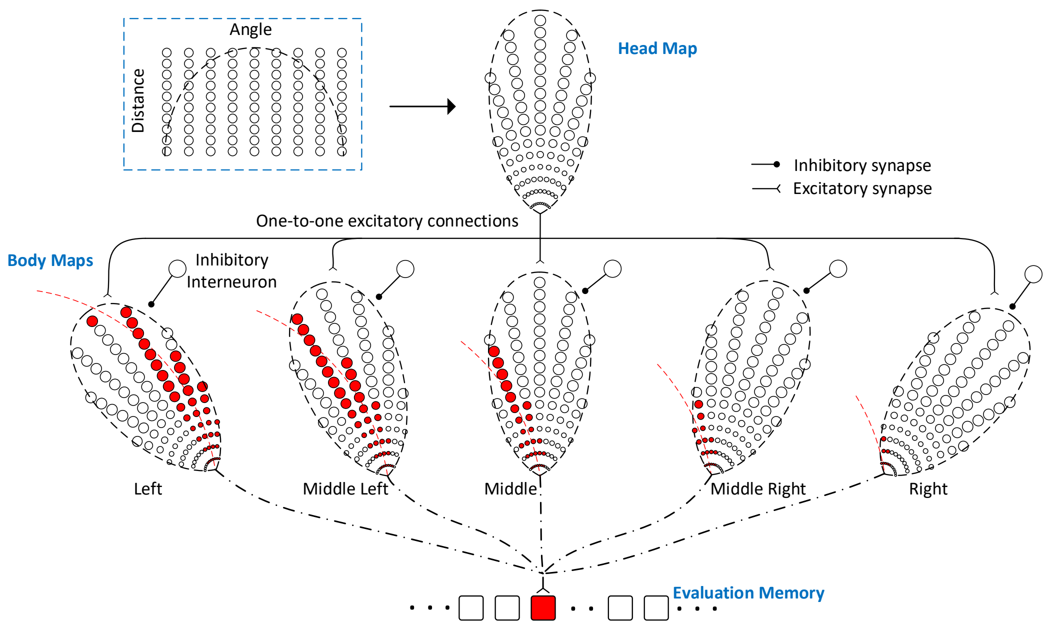


Fig. 23. The sensory layer consists of a 2-D sensory map in head frame (head map) and five sensory maps in body frame (body maps). The elliptical dashed lines around the maps represent the field of view of the bat and the circles represent sensory neurons which fire when obstacles are detected in their locations. The number of neurons in each map is reduced for clearer illustration. Each body map has an inhibitory neuron that inhibits all the neurons in the map when it fires a spike. The neurons in the head map strongly excite the neurons in the same positions in each body map but only one of the body maps will not be inhibited by the inhibitory interneuron after a sonar ping, depending on the ping direction. The neurons in the body maps make fixed excitatory connections (dash-dotted line) to the evaluation memory units if the represented obstacles in their positions are blocking the paths. One example of the connections between the body maps and an evaluation memory unit is shown in red.

spikes from the active body map are adjusted by adding extra latencies before the spikes are sent to the evaluation memory. Whether or not an evaluation memory unit takes synaptic inputs from neurons in body maps is determined by whether obstacles in their positions are blocking the path that the evaluation memory unit represents. If they are blocking the path, the spikes from the corresponding neurons will be sent to the integrating neuron in the evaluation memory, which will be described in detail in Section 3.5.2. The synaptic connections between the sensory layer and the evaluation memory are fixed if the paths are fixed.

The Openspace algorithm in [20] made clever use of the natural latency of echoes as a representation of the straight-line (i.e., radial) distance to obstacles. In the Curved Openspace algorithm, however, the immediacy of avoidance uses the distance along a curved path (Eqn. (10)). To adjust the spike latency to reflect the immediacy correctly, each neuron in the body map connects to a delay neuron (not shown in Fig. 23) that adds a constant latency specific to each position before connecting to the evaluation memory. The added latency for a connection between a sensory neuron j to the evaluation memory of path p can be calculated as

$$\Delta t(j,p) = T_{\mathrm{MAX}} \cdot \frac{r_p(j) - R_C}{R_{\mathrm{MAX}} \cdot \gamma_p} - t_{\mathrm{echo}}(j) + T_C \tag{15}$$

where $\Delta t(j,p)$ is the added latency and $t_{\mathrm{echo}}(j)$ is the echo latency from an obstacle represented by neuron j. $t_{\mathrm{echo}}(j)$ is constant for each sensory neuron since each neuron represents a fixed location on the body map. T_{MAX} is the echo delay from an obstacle at the maximum sensing distance R_{MAX} and is the maximum echo delay the sonar system can receive. The term $\left(r_p(j) - R_C\right)/\left(R_{\mathrm{MAX}} \cdot \gamma_p\right)$ is the same term used in the immediacy function (Eqn. (10)) that represents the time before the bat collides with an obstacle at the location of neuron j. $r_p(j)$ is the distance from the bat to a blocking obstacle j along path p, R_C is the radius of the zone of collision, R_{MAX} is the maximum distance that the sonar system can detect and γ_p is the speed profile for path p. The term has a value from 0 to 1, which makes the first term in Eqn. (15) have a value between 0 and T_{MAX}. T_C is a small and constant delay to keep $\Delta t(j,p)$ positive. With the added latency, the spike latency that arrives at the evaluation memory of path p from sensory neuron j (if an obstacle is sensed) can be written as

$$t_{\mathrm{spk}}(j,p) = \Delta t(j,p) + t_{\mathrm{echo}}(j) = T_{\mathrm{MAX}} \cdot \frac{r_p(j) - R_C}{R_{\mathrm{MAX}} \cdot \gamma_p} + T_C \tag{16}$$

A closer blocking obstacle on a path with a higher speed profile will result in a shorter spike latency, indicating a higher immediacy of avoidance.

3.5.2 Evaluation Memory

The role of the evaluation memory (as described in earlier sections) is to hold the spatially integrated value of immediacy along each path even when the sonar is interrogating a different direction and does not receive new sonar information for a given path. Given the spike latency representation, the output of the evaluation memory unit is a spike with a latency (following the sonar ping) that matches the previously observed latency (when the sonar was receiving new data). Each evaluation memory unit receives spikes with different latencies from the sensory neurons along a path and integrates them into a spike latency with the integrating neuron. An array of memory units along with a delay line detects and stores the occurrence of a spike at a particular latency and is then able to regenerate the spike upon later activation. The neural circuit of each evaluation memory unit is shown in Fig. 24.

An example of the mechanism of the integrating neuron is shown in Fig. 25. The integrating neurons are integrate-and-fire neurons and their membrane potentials simultaneously reset when the bat pings (t=0). Whenever the integrating neuron receives a spike, a step-excitation current is turned on for a duration, causing its

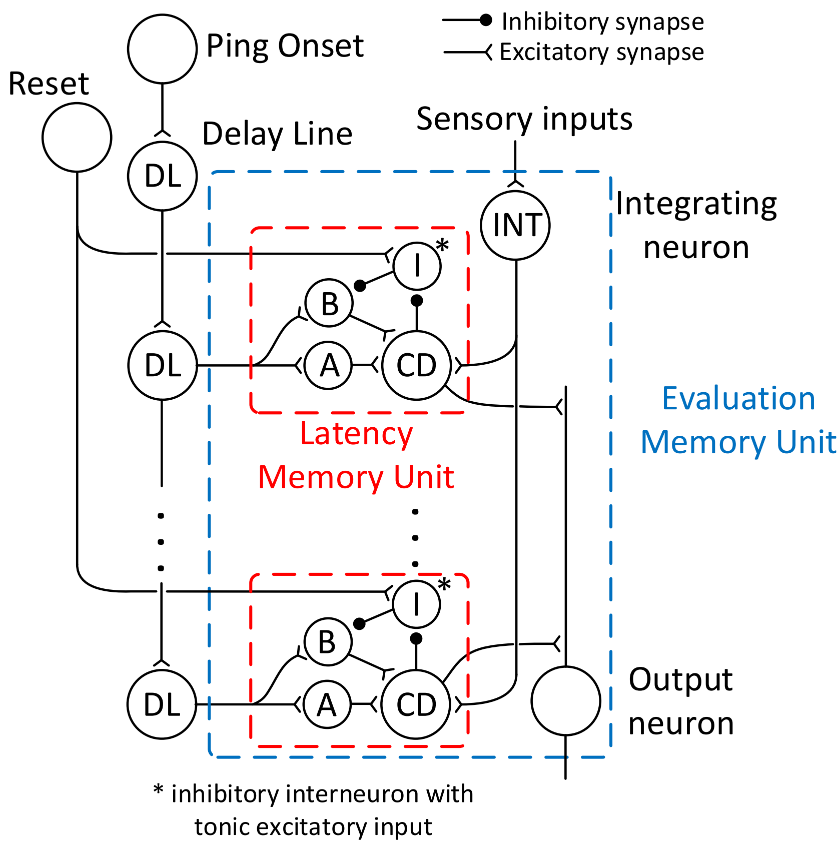


Fig. 24. The structure of an evaluation memory unit. The integrating neuron combines the spike train from sensory neurons into a single spike with a latency representing the risk of the corresponding path. The spike latency is then stored in an array of latency memory units. The memory can be reset by exciting the reset neuron. The output neuron combines the spikes from latency memory units into a single spike train and sends it to the action selection layer.

membrane potential to rise. If the membrane potential reaches a threshold, the integrating neuron fires a spike. The spike latency is shorter when excitatory spikes arrive earlier. The excitatory currents from different spikes are summated and can further reduce the latency of the spike.

For an integrate-and-fire neuron with a membrane capacitance C_{mem0}, a spike threshold V_{th0}, and n ($where\ n \geq 1$) step-excitation currents each with an amplitude of E_0 activated at time $t_1, t_2, .., t_n$, the latency of the spike T_{eval} is given by

$$T_{\mathrm{eval}} = \frac{C_{\mathrm{mem0}} V_{\mathrm{th0}} + E_0 \cdot \sum_{i=1}^{n} t_i}{n \cdot E_0} \tag{17}$$

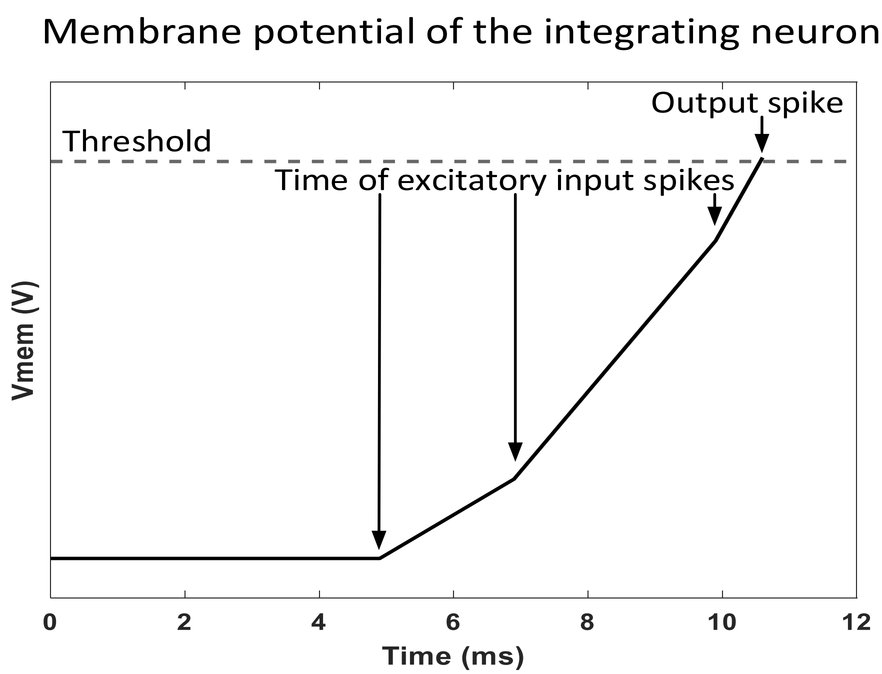


Fig. 25. The mechanism of the integrating neuron. Each input spike triggers a step-excitation current that charges up the membrane potential V_{mem} and the currents from different sensory neurons accumulate. A spike is fired when V_{mem} crosses a threshold. An input spike that arrives earlier will result in a shorter spike latency. Input spikes after the first one also reduces the spike latency by increasing the charge speed.

assuming that $t_1 \le t_2 \le \cdots \le t_n < T_{\text{eval}}$. Each input spike that arrives before the output spike reduces the spike latency, but the latency cannot be reduced lower than the latency of the first input spike. The spike latency T_{eval} represents the integration of immediacy of avoidance with a saturation limit similar to the second term in Eqn. (11), although the integration is different from simple addition. If the integrating neuron receives one or more spikes from the sensory layer, it will fire a spike with a latency of T_{eval} which resembles the evaluated risk of a path. Different from the risk calculation in the analog model of the algorithm (Eqn. (11)), here a smaller latency means a larger risk value. The latency of the spike then needs to be stored in the evaluation memory.

An array of latency memory units and a delay line are used to store the latency of the spike from the integrating neuron and later generate a spike with the same latency when needed without the sensory inputs. As is shown in Fig. 24, each evaluation memory unit has an array of latency memory units while the delay line is shared among all the evaluation memory units. The delay line is triggered by the onset of the sonar ping, and it generates spikes with different latencies which are sent to different latency memory units in different evaluation memory units. In this neural model, the delay line is implemented as neurons connected in serial with excitatory synapses and each spike from the previous neuron causes the next neuron to fire with a certain delay. The delays between neurons in the delay line affect the resolution of the latency memory.

As shown in Fig. 24, a latency memory unit consists of four neurons: two excitatory interneurons (A and B), a coincidence detector (CD), and an inhibitory interneuron (I) with tonic excitatory input. Neuron A and B both take excitatory input from a neuron in the delay line, but neuron B also takes inhibitory input from neuron I.

Besides the tonic excitatory input, neuron I is strongly inhibited by the CD neuron and strongly excited by a reset neuron. Without the input from the CD and the reset neuron, neuron I fires tonically and keeps neuron B inhibited. The CD takes excitatory inputs from neuron A, neuron B, and the integrating neuron. For a CD to fire a spike, two spikes need to arrive at around the same time, meaning that two out of the three neurons exciting the CD need to fire simultaneously.

At an idle state before a sonar ping, neuron B is inhibited by neuron I. When the bat pings, the ping onset starts the spike propagation in the delay line, and a spike with a certain latency will be sent to neurons A and B. Neuron A will be excited by the input spike and send a spike to the CD, whereas neuron B will not fire because it is strongly inhibited by neuron I. At this point, only a spike from the integrating neuron with the same latency as the spike from the delay line will be able to trigger a spike from the CD. If this is the case, the CD will send a spike to neuron I and keep it from firing again for the duration of the inhibitory synaptic input (around 300 ms). If the bat pings again while neuron I is still inhibited, neurons A and B will both fire and the CD will fire again even without the spike from the integrating neuron. Since the CD fires again, neuron I is kept inhibited for a new duration, allowing the next sonar ping to cause neuron B to fire. Unless neuron I is reset by the reset neuron or the bat doesn't ping for a long time, the memory of the spike latency from the integrating neuron is maintained and a spike with the same latency is reproduced after every sonar ping without any further sensory inputs.

The CD also sends an excitatory spike to the output neuron when it fires. The output neuron in each evaluation memory unit fires a spike with very little delay whenever it receives spikes from any of the latency memory units. Like an OR function, it combines all the spikes from the delay-tuned neurons into a spike train. Due to the mechanism of the action selection layer, as will be described later in section 3.5.4, only the latency of the first spike in the output spike train affects the calculation of the desirability. It means that in the scenario where an evaluation memory unit with a stored spike latency receives new sensory input, only the spike with the shorter latency is meaningful to the next layer. Since a shorter spike latency represents a higher risk value, this behavior is consistent with the description earlier in section 3.2.3 that the final risk value is the maximum value between the decayed old risk and the new risk computed from the objects currently being sensed.

3.5.3 Action Selection Layer

The action selection layer consists of two WTA layers (Fig. 26) with a similar structure, and both of the layers use a "race-to-first-spike" WTA mechanism to select the winning motor actions.

WTA layer 1 is similar to the temporal WTA circuit proposed in [20]. It compares the desirability of different paths using the latency of the spikes from the evaluation memory layer and selects the most desirable path with a "race-to-first-spike" WTA mechanism. WTA layer 1 consists of action selection neurons with an integrate-and-fire mechanism, a recurrent inhibitory interneuron (RIN), and a group of goal neurons (Fig. 26). The numbers of action selection neurons and the goal neurons are the same as the number of motor actions. The action selection neurons in WTA layer 1 receive weak excitatory input from the goal neurons that indicate the location of the goal. Each goal neuron connects to the field of action selection neurons with a

Gaussian-shaped pattern of synaptic strengths (not all the connections are shown in Fig. 26 for clarity). Only one of the goal inputs will fire a spike at a time to indicate the path that leads to the goal. This excitatory connection corresponds to the Gaussian-shaped goal input in Eqn. (12) described in section 5.2.4. The action selection neurons also receive weak excitatory input from the corresponding ping direction neuron through a delay neuron ("Delay" in Fig. 22). Upon receiving a spike from the ping direction neuron, the delay neuron fires a spike after some delay to the action selection neurons

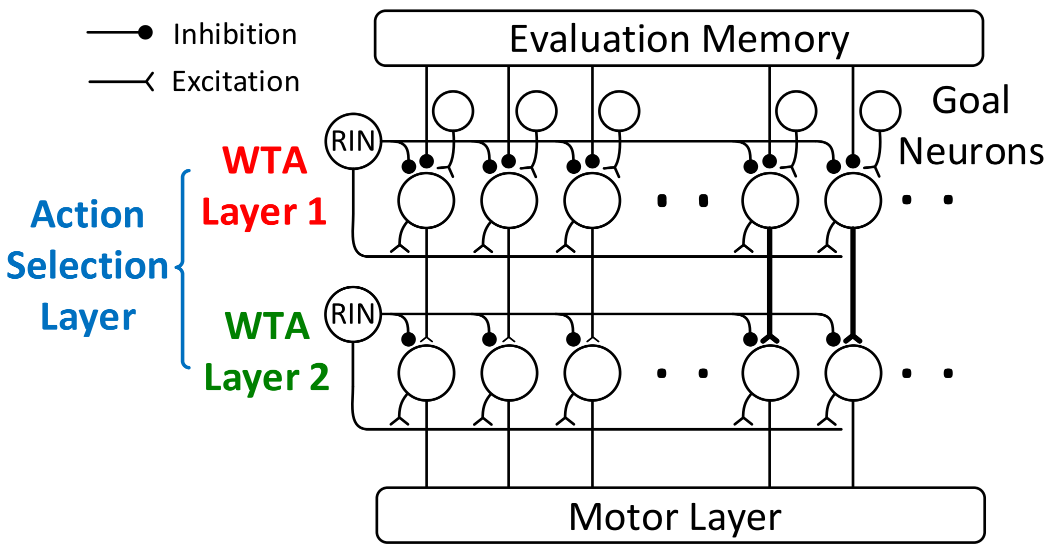


Fig. 26. The action selection layer is composed of two WTA layers: WTA layer 1 and WTA layer 2. Both layers use a "race-to-first-spike" WTA mechanism, where the first neuron to spike excites a recurrent inhibitory neuron (RIN) to fire that suppresses other neurons within the layer from firing. Because WTA layer 1 can produce multiple winners that represent significantly different turning, WTA layer 2 is added to only allow the winners from the same ping direction group to go to the motor layer. In WTA layer 2, neurons towards the center (straighter paths) have a stronger synaptic connection from WTA layer 1 (represented by thicker synapses in the figure), giving those paths priority over paths towards the side (sharper turns).

associated with the same ping direction. This excitatory connection corresponds to the ping direction bias term in Eqn. (13). Besides the connections from goal and ping direction neurons, all the action selection neurons receive passive excitatory currents that reflect the baseline desirability of different paths, corresponding to the "Constant Bias" term in Eqn. (13). This excitatory current could be either from a neuron firing tonically or intrinsic membrane currents [34].

When a sonar ping is emitted, the ping onset neuron simultaneously resets (i.e., strongly inhibit and then release) all the action selection neurons. The passive excitatory currents can then be inversely expressed in the spike latency across the field of neurons (Fig. 27(a)). In the absence of other inputs, the neuron that receives the strongest excitatory current will integrate to the threshold first and is the winner, meaning that the motor action with the largest constant bias will win. The excitatory spikes from the goal and ping direction neurons increase the membrane potential, thus decreasing the amount of charge that the IF neurons need to reach the firing threshold and making them more likely to win (Fig. 27(b)).

Each evaluation memory unit is connected to all the action selection neurons through inhibitory synapses that activates a long-lasting step-inhibition current if a spike arrives (not all connections are shown in Fig. 26 for clarity). The synapses have a Gaussian distribution of synaptic strengths with the peak centered on the synapse connecting the neurons representing the same path. Different synaptic weights mean different amplitudes of the activated inhibitory current. This way of connecting the evaluation memory and the action selection layer corresponds to the Gaussian inhibition term in Eqn. (13). The standard deviation of the Gaussian distribution, however, is small ($\sigma_c = 1$ in the simulation) and in practice synapses $3\sigma_c$ away from the center can be pruned without affecting the performance. The inhibitory current from a synapse saturates when the synapse receives a spike and any following spikes to the same synapse will not increase the amplitude of the inhibitory current. The saturating

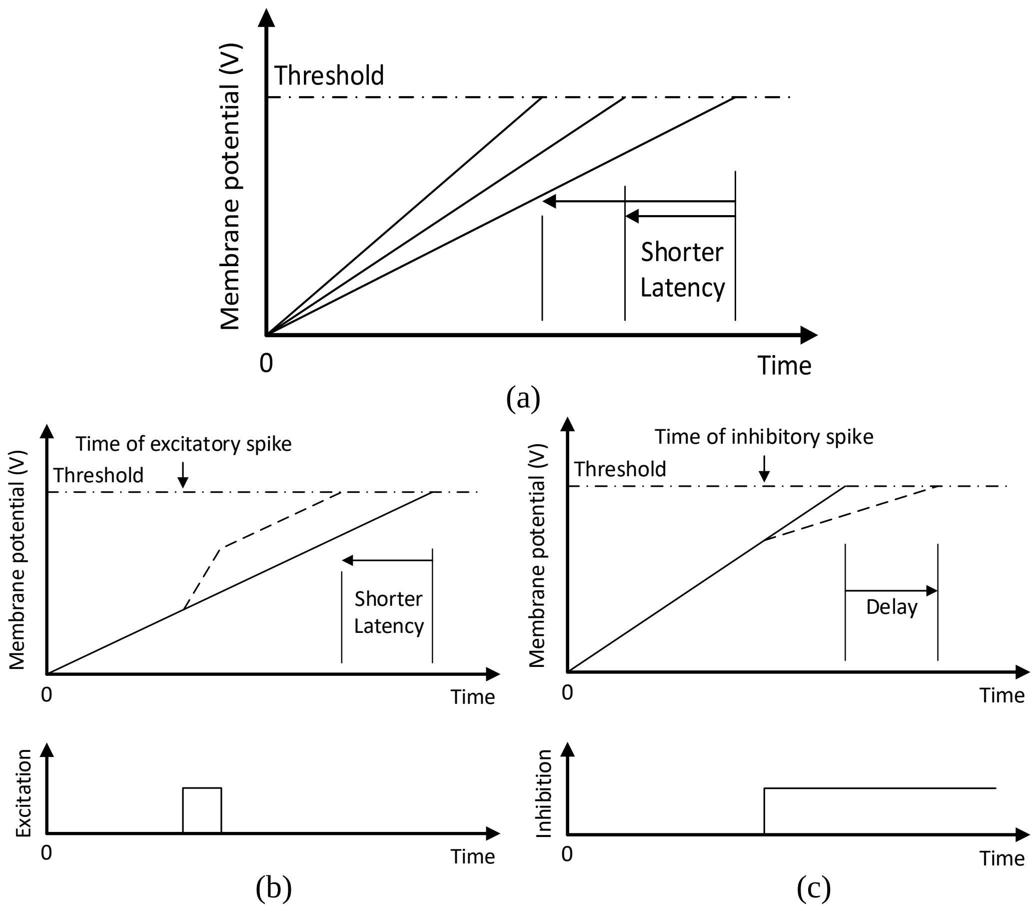


Fig. 27. The temporal WTA mechanism with an integrate-and-fire neuron. The neuron fires a spike when its membrane potential reaches the threshold (dash-dotted line) (a) Following a reset, increasing the passive excitatory currents increases the charge speed of the membrane potential and shorten the latency of the spike. (b) A pulse of excitatory current increases the membrane potential, shortening the spike latency (dashed line). (c) A long-lasting step-inhibition current increases the latency of the spike (dashed line) or prevents firing altogether. An inhibitory spike that arrives earlier produces a longer delay in firing.

current from a synapse is the reason why only the latency of the first spike from an evaluation memory unit affects the computation of desirability.

The accumulated inhibitory currents from different synapses slow the rate of charging of the action selection neuron and the time to spike will increase as inhibitory inputs start earlier (Fig. 27(c)). Because the spike latency from the evaluation memory is inversely related to the risk of different paths, a path with a higher risk will result in an earlier activation of inhibitory currents and thus increase the spike latency of the action selection neurons. In combination with passive currents and excitatory spikes from the goal and ping direction neurons, the action selection neuron that fires the first spike indicates the most desirable path.

For an action selection neuron with a membrane capacitance C_{mem}, a spike threshold V_{th}, a passive excitatory current I_{exc}, a sum of injected charge from goal neurons Q_{goal} and ping direction neurons Q_{dir}, and *n* step-inhibitory currents activated at time $t_1, t_2, \ldots, t_n$, the latency of the spike T_{WTA} is given by

$$T_{\text{WTA}} = \frac{C_{\text{mem}}V_{\text{th}} - Q_{\text{goal}} - Q_{\text{dir}} - \sum_{i=1}^{n} I_i \cdot t_i}{I_{\text{exc}} - \sum_{i=1}^{n} I_i} \tag{18}$$

where I_i is the amplitude of the inhibitory current activated at time t_i. It is assumed that $t_1 \leq t_2 \leq \cdots \leq t_n < T_{\text{WTA}}$, meaning the spikes that arrive after the neuron fires are ignored. It is also assumed that $I_{\text{exc}} - \sum_{i=1}^{n} I_i > 0$, or else no spike is generated. As is shown in Eqn. (18), a larger passive excitatory current decreases the spike latency by increasing the denominator while stronger excitatory inputs from goal and ping direction neurons decrease the latency by reducing the numerator. Earlier arrival of inhibitory spikes (smaller t_i) increases the latency by increasing the numerator.

Although the amplitudes of the inhibitory currents I_i affect both the denominator and the numerator, stronger inhibitions that arrive at the same time still increase the spike latency. In a cluttered environment where obstacles are blocking many paths, the sum of the inhibitory currents could exceed the sum of excitatory currents and prevent the neuron from firing a spike.

The first spike from the action selection neurons then causes a recurrent inhibitory neuron (RIN) to fire which recurrently inhibits every action selection neuron and prevents them from firing. Due to the delay of the RIN neuron and the slow activation of inhibitory synapses, however, all the spikes that are fired before the inhibition will go through the WTA mechanism and multiple winners can appear. Having multiple winners at the output of the WTA layer could cause problems in both head motor layer and body motor layer. In our model of the motor layers, we assume that multiple activation in the motor neurons would cause averaging of the motor actions. Multiple winners that encode nearby head directions and motor actions do not necessarily cause problems, but winners far away from each other will. To solve the problem with multiple winners, a second WTA layer ("WTA Layer 2" in Fig. 26) is added to ensure that the winners sent to the motor layer are in the same ping direction group by eliminating some of the winners.

The structure of WTA layer 2 is the same as WTA layer 1 except for the input synapses. The action selection neurons in WTA layer 2 take one-to-one excitatory connections from the neurons in WTA layer 1. All of these connections are strong enough to fire the excitatory neurons with a single spike, and stronger synaptic connections produce spikes with shorter delays. The synaptic strengths are the same for the neurons in the each of the 5 ping direction groups while they differ from group to group to create 5 different delays. The values of the delays are designed so that the minimal difference between delays is longer than the delay of the recurrent inhibition. As a result, although WTA layer 2 can still produce multiple winners, it is guaranteed that the winners are from the same ping direction group. The values of the synaptic strengths decide the priority of each ping direction group. It is beneficial in terms of safety and energy cost to favor slow turns over sharp turns, so in this model, the synaptic strengths in the middle group are set to be the highest, followed by the middle-left group, the middle-right group, the left group and then the right group.

3.5.4 Motor Layer

The function of the motor layer is to produce the motor actions needed to orient the head of the bat (head motor layer) or to turn the body (body motor layer) to follow a selected path.

Five head motor neurons representing five head (i.e., ping) directions receive excitatory input from the neurons in the action selection layer that are associated with the same ping direction (Fig. 22). Upon receiving a spike, a head motor neuron becomes active and orients the head of the bat in the associated ping direction for the next sonar ping, consistent with the proposed attentional system described in section 3.2.5. Because only the neurons in the same ping direction group can win in the action selection layer, as described in section 3.5.3, no more than one head motor neuron will be active at the same time.

Each body motor neuron is associated with a path, and when it fires, it produces the correct body/wing changes to execute the turn rate needed for its associated path.

When multiple body neurons are active, the bat is modeled to produce the averaged turn rate and execute the averaged path, similar to the population coding hypothesis seen in other animals [35]. The body motor neuron receives excitatory input from the action selection neuron associated with the same path and inhibitory input from an interneuron inhibited by the recency memory. Each recency memory is associated with a ping direction, and it is a decaying memory that stores the recency of its associated ping direction. The recency memory becomes active when a ping direction neuron fires, and it inhibits the interneuron for some duration until decayed back to its inactive state. The duration of the memory is decided by the duration of the excitation from the ping direction neurons. In this scenario, the inhibited interneuron can no longer inhibit body motor neurons and they are allowed to fire upon receiving spike input from the action selection layer. Through this disinhibition mechanism, a bat can turn its body to follow a winning path only when it has recent information in its associated ping direction.

In the situation where none of the action selection neurons fire a spike in a certain time window after the outgoing ping, all paths are undesirable for the bat to follow and the emergency "turnaround" maneuver will be executed as described in section 3.2.5. It is achieved by adding an "emergency" neuron in the action selection layers with a fixed spike latency and connecting it to a motor neuron responsible for the emergency maneuver. The emergency neuron will be inhibited if any other action selection neurons fire a spike before it does, creating a time window where a path is good enough to follow.

3.6 Spiking Neural Model Simulation

The proposed spiking neural model was implemented in Python using a combination of Hodgkin–Huxley neurons [36] and leaky integrate-and-fire (LIF) neurons. The Hodgkin-Huxley model was applied to as many neurons as possible to show the biological plausibility of the proposed spiking neural network, and LIF neurons were used to implement sensory neurons in the sensory layer, integrating neurons in the evaluation memory and WTA neurons in the action selection layer. The sensory neurons were implemented in LIF neurons to speed up the simulation because of their large number and their simple function (fire a spike upon receiving a spike). Implementing them with the LIF neurons does not change their function significantly compared to the implementation with Hodgkin-Huxley neurons. The integrating neurons and the WTA neurons used LIF models because their function includes integrating input spikes over a period. The numbers of neurons in different layers are shown in Table 1.

Table 1. Number of Neurons in the Proposed Spiking Neural Model

Type of Neurons	**LIF**	**Hodgkin–Huxley**	**Total**
Sensory Layer	2831	0	2831
Evaluation Memory	33	2028	2061
Action Selection Layer	67	2	69
Motor Layer	0	48	48
All Layers	2931	2078	5009

The synaptic connections to the Hodgkin-Huxley neurons were implemented with a time-dependent conductivity function $g_{\mathrm{syn}}(t)$ that changes when a pre-synaptic spike arrives. The synaptic current is calculated based on the difference between its reversal potential E_{syn} and the membrane potential of the neuron $v(t)$.

$$I_{\mathrm{syn}}(t) = g_{\mathrm{syn}}(t) \cdot [v(t) - E_{\mathrm{syn}}]$$

The reversal potential E_{syn} and the function $g_{\mathrm{syn}}(t)$ can be used to describe different types of synapses. For excitatory synapses, E_{syn} is set to 10 mV whereas for inhibitory synapses, E_{syn} is set to -70 mV. An alpha function was used to describe the synaptic conductance $g_{\mathrm{syn}}(t)$ as:

$$g_{\mathrm{syn}}(t) = \bar{g}_{\mathrm{syn}} \cdot (t - t_0)(\mathrm{e}/\tau)\mathrm{e}^{-(t-t_0)/\tau} \cdot \Theta(t - t_0)$$

where τ is the time constant, $\bar{g}_{\mathrm{syn}}$ is the maximum conductance, t_0 is the arrival time of a pre-synaptic spike and $\Theta(t)$ is the Heaviside step function.

The synaptic connections to LIF neurons are simplified as current injections, $I_{\mathrm{syn\_LIF}}(t)$, which are described by a boxcar function as

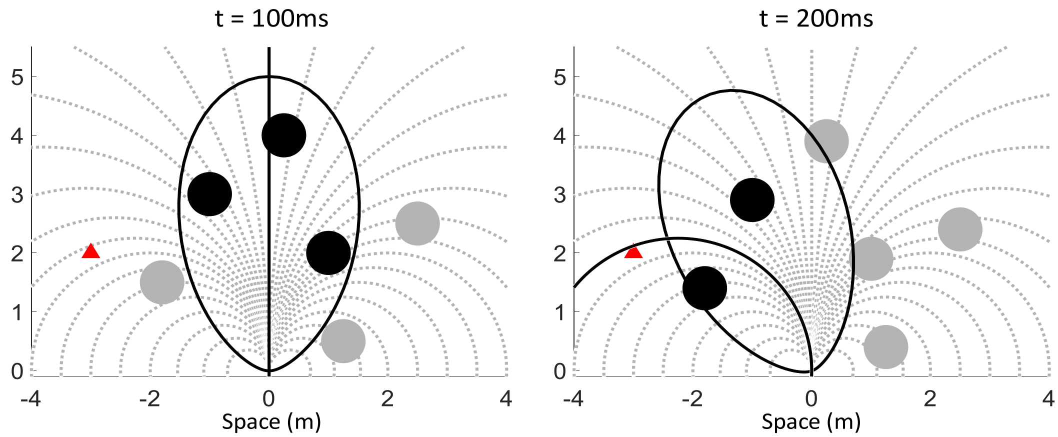
$$I_{\mathrm{syn\_LIF}}(t) = \bar{I}_{\mathrm{syn}} \cdot [\Theta(t - t_0) - \Theta(t - t_0 - T_{\mathrm{syn}})]$$

where $\bar{I}_{\mathrm{syn}}$ is the amplitude of the synaptic current and T_{syn} is the duration of the synaptic input.

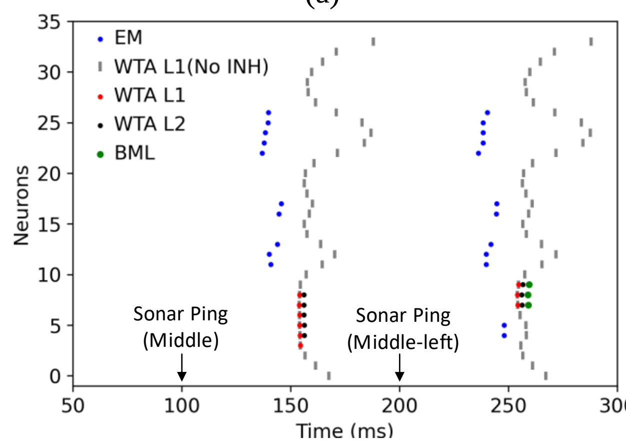
Each sensory map in the sensory layer consists of 471 neurons representing locations of up to 25 ranges at 0.2-meter increments (from 0.2m to 5m) and 31 angles at 4-degree increments (from -60° to 60°). Big brown bats (*Eptesicus fuscus*) are shown to be able to discriminate range differences as small as 1 cm [37] and angular differences of 6° [38], while other species such as the greater spear-nosed bat

(*Phyllostomus hastatus*) can differentiate objects that are 4° apart. We used a relatively small number of neurons to represent the range because a higher range resolution significantly increased the total number of neurons and slowed the simulation while not significantly improving the performance of obstacle avoidance. The neurons that would represent locations outside of the field-of-view of the sonar were removed, thus the number of neurons in each map is smaller than the total number of possible combinations of ranges and angles. During the simulation, if an obstacle is within the sonar field of view when it pings, the obstacle will cause the sensory neuron with the closest receptive field in the head map to fire a spike.

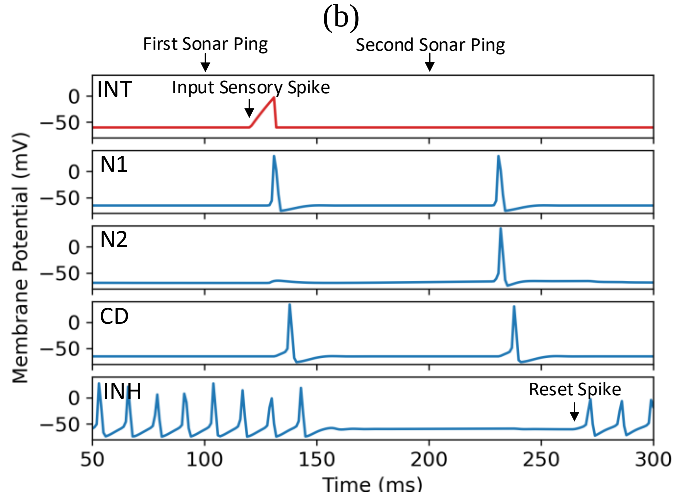
An example of the simulation from the neural model is shown in Fig. 28. The environment is the same as the example given in Fig. 15 for a clearer comparison between the analog version of the algorithm and the neural model. The spikes from the evaluation memory, action selection layer and body motor layer are plotted in Fig. 28(b) to demonstrate the flow of activity in the network. After the first sonar ping, the EM combined information from the sensory layer and produced three groups of spikes (blue dots) whose latencies represent the risk values of the paths that were blocked by the three detected obstacles. Smaller latencies from the EM represent larger risk values. The EM spikes with smaller latencies caused larger delays in the firing of the WTA neurons (grey vertical line), making those paths less desirable and less likely to win in the temporal WTA. Because several neurons in WTA layer 1 fired simultaneously, there were multiple winners (red dots). The biased WTA (layer 2) selected the winners that were closer to the center and in the same ping direction group (black dots).



(a)



(b)



(c)

Fig. 28. (a) The bat arrived in an environment with 6 obstacles (filled circle) and a desired destination (filled triangle in red) without prior information of the environment. It is assumed that it was flying along a straight path (solid straight line) and it directed its first sonar ping to the front (ellipsoidal FOV shown in solid line). (Continued on next page)

The circles of the obstacles are drawn with the size of the zone of collision to show the paths that they are blocking. The bat has a limited selection of motor choices (dotted line, 33 in total) and each of the motor actions corresponds to a neuron in the spike raster plot below. The bat sent the first sonar ping at t=100ms and a second one at t=200ms. The detected obstacles (filled circle in black) from each of the sonar ping caused the corresponding sensory neurons to fire. (b) The spikes from the output neurons in the evaluation memory (EM), the WTA neurons in the action selection layer and the neurons in the body motor layer (BML) were shown. The spike latency of each EM neuron represents the 'risk' value of a path, and the spike latency of each WTA neuron represents the desirability of a path. The neurons with smaller indices represent paths to the left and the neurons with larger indices represent paths to the right. The spikes from an imaginary WTA layer 1 without recurrent inhibition are also plotted (grey vertical line) for a better demonstration of the impact that the spikes from the EM had on the action selection layer. (c) The membrane potentials of the neurons in the 9th latency memory unit (LMU) and the integrating neuron (INT) for path 24 are shown to demonstrate the working mechanism of the evaluation memory. The integrating neuron was implemented as a LIF neuron (red curve) and the neurons in the LMU (neurons N1, N2, CD and INH) were implemented using Hodgkin-Huxley model (blue curve).

The spikes from the winners, which belong to the middle-left group, caused the middle-left head motor neuron to fire a spike (not shown). As a result, the bat turned its head to the middle-left direction for the next sonar ping (right panel in Fig. 28(a)). Because the bat did not have recent sensory information in the middle-left direction, no body motor neurons in the BML fired and the bat kept its original trajectory. The bat also maintained the memory of the spike latencies from the detected obstacles in the EM.

After the second sonar ping at 200ms, the output neurons in the EM fired spikes in response to the two currently detected obstacles (EM neurons 4-5 and 11-13). At the same time, the neurons reproduced the spike latencies from the previously detected obstacles that were no longer visible (EM neurons 16-17 and 22-26). Because the three winners from the action selection layer were all from the middle-left direction and the bat had recent information in this direction, the BML neurons fired spikes (green dots) and the bat executed the path averaged from the fired BML neurons (right panel in Fig. 28(a)).

The membrane potentials of the neurons in an activated latency memory unit in this example are shown in Fig. 28(c) to demonstrate the mechanism of the evaluation memory. The N1 neuron was excited by a delay line neuron (not shown) and fired a spike with a fixed latency after every sonar ping. Although the N2 neuron was excited by the same delay line neuron, it did not fire after the first sonar ping because it was inhibited by the tonically firing inhibitory interneuron (INH). Because a blocking obstacle was detected after the first sonar ping, the integrating neuron (INT) received a spike from the sensory layer and produced a spike with a latency representing the immediacy of avoiding that obstacle. When the spike from the INT neuron and the spike from the N1 neuron arrived at the coincidence detector (CD) neuron within 3ms of each other, the CD neuron fired a spike which excited the output neuron of the evaluation memory unit. The spike from the CD neuron also inhibited the INH neuron and suppressed its firing for a long duration (~150ms), releasing neuron N2 from its inhibition.

After the second sonar ping, the N2 neuron fired a spike at the same time as the N1 neuron because they were excited by the same delay line neuron. The coincidence of the spikes from N1 and N2 neurons caused the CD neuron to fire again and thus the same spike latency was reproduced. The spike from the CD neuron suppressed the INH neuron again and "extended" the latency memory. In this example, because the bat made a motor decision after the second sonar ping, all the latency memories were reset by an excitatory spike to the INH neuron ("reset spike" in the bottom panel in Fig. 28(c)) that was strong enough to overcome the previous inhibition from the CD neuron and caused the INH neuron to start firing again. Once the INH neuron is firing, the N2 neuron is suppressed, and the CD neuron will no longer fire without the input from the INT neuron. Hence, the latency memory is "erased".

The neural network was also simulated in the same environments used in the simulation of the analog algorithm described in section 3.3. The parameters of the bat were kept the same including the radii of the paths, shape of the field-of-view, ping direction and speed profile. The maximum ping rate is set to 10 Hz. The bat was traveling at a maximum speed of 2m/s in a 50m x 50m field with 1400 randomly placed obstacles (Fig. 29). The field has two-dimensional periodic boundary conditions to allow the bat to travel as if in an infinitely large space. A goal input is provided to drive the bat to a goal destination at the left boundary (X=0) with the same vertical position (same Y coordinate). One of the simulation results is shown in Fig. 29 where the bat successfully reached the left boundary 4 times without colliding with any obstacles.

Due to the complexity of the Hodgkin-Huxley model, simulating the proposed spiking neural network on a CPU (Intel Xeon Cascade Lake) is extremely slow (~ 10^5

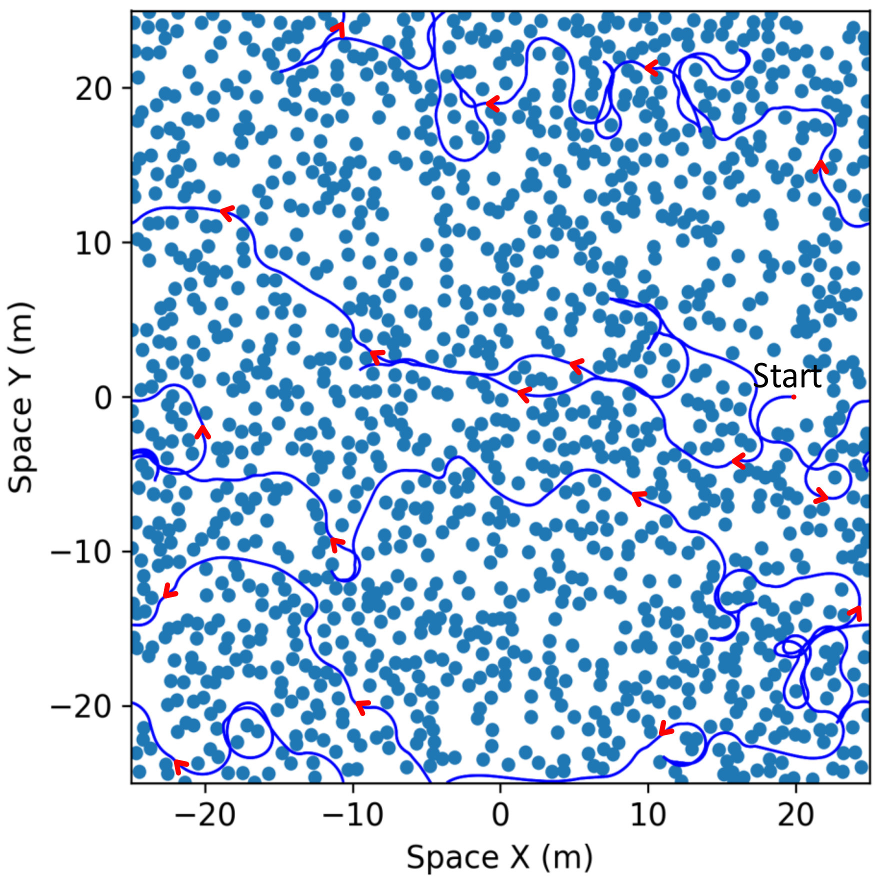


Fig. 29. An example of the bat traveling in a dense forest using the proposed neural network with a spike-timing representation. 1400 obstacles (filled circles in blue) were placed in a 50m x 50m area. The size of the obstacles shown in the figure is the size of the zone of collision (combination of the sizes of the bat and the obstacle). The paths that the bat traveled on the field are shown as blue curves. The bat successfully reached the goal (left boundary) 4 times without colliding with any obstacles.

times slower than simulating the analog version), and collecting enough data to do performance analysis like in section 3 is impractical with the current implementation. The short simulation run shown in Fig. 29 took more than 5 days to finish. A neuromorphic VLSI chip, however, can vastly improve the running speed of the proposed spiking neural network and can allow it to run in real-time on a mobile platform.

3.7 Discussion

This chapter presents an obstacle avoidance algorithm that provides steering decisions to a sonar-guided "bat" to pursue a goal while avoiding static obstacles. To achieve better matching between the desired path and a path that the bat can practically follow, the Curved Openspace algorithm projects the sensory-based obstacle data into "motor space" before comparing motor choices. Although the idea of selecting motor actions instead of sensory directions has been proposed before [21][22], they used sensors with a wide field-of-view and did not need to consider the question of how to gather information efficiently over time. Taking into account the limited field-of-view of an echolocating bat or a practical sonar system, an attentional system is proposed to control the direction of sonar pings in a smarter way where the bat will ping in the most beneficial direction to guide its motor action selection. The fact that the bat is moving while gathering and integrating information introduces problems such as the inaccuracy of old data. To alleviate the problem of inaccurate memory, we designed an evaluation memory that decays old data and a desirability function that incorporates both the information about the goal and obstacles as well as the recency of the sensory data. The presented simulations showed the effectiveness of different configurations under different situations. Although the proposed algorithm is described as an algorithm for sonar-guided creatures or vehicles, it can be implemented with other sensors such as cameras or laser rangefinders.

The algorithm was implemented in real-time on a car-like robot with an active sonar system. The robot was tested in multiple scenarios and the results showed that the analog version of the algorithm works with real sonar sensory data. During these runs, the robot would only turn its head and collect sensory data in non-traveling directions when necessary, showing the efficiency of the proposed attentional system. A neural model of the proposed algorithm using spike-timing representations is also described in this chapter. Instead of using a large population of neurons to represent analog signals with spike rates, we used spike-latency to represent signal values for computation. Sonar systems already have a timing code where low echo latency represents a close object and high echo latency represents a distant one. Only a small adjustment on the latency is required to represent the immediacy of avoidance as a spike latency. Moreover, a sonar system will receive echoes from close obstacles earlier than distant obstacles. It means that a creature implemented with the proposed neural model has a choice of reducing the time to make a decision if it wants to focus more on closer obstacles in a cluttered environment. With the “race-to-first-spike” WTA mechanism, one can force an early decision by increasing the passive excitatory input to further reduce the time spent on decision-making if most of the computation happens early. The presented simulations showed that the bat running on the proposed neural network is able to navigate through dense forests. The slow simulation speed of the neural network, however, prevents the current simulation setup from running extensive performance analyses and advocates an implementation on neuromorphic hardware such as a VLSI chip. Overall, the use of input spike timing to modulate the efficacy of a synaptic connection can be an effective mechanism that does not rely on increasing spike rates or modulating synaptic strength. Although the Curved Openspace algorithm and its neural model are proposed for obstacle avoidance on a 2-D plane in this chapter, it would not be difficult for the algorithm to operate in the 3-D world by extending the motor actions to 3-D trajectories.

Chapter 4: A CMOS Floating-gate Circuit for a Fixed Rate of Increase and Proportional Decay of Synaptic Weights

4.1 Introduction

The hardware implementation of synaptic memory circuits for artificial neural systems has long been a tricky proposition. While biological synapses exhibit dynamics with multiple state variables and long-term storage of memory, commercial VLSI technology has yet to provide a more compact, high-precision memory than storing charge on the floating-gate of a MOSFET. The challenge in their widespread use, however, has remained the cumbersome circuitry needed to modify their values in a controlled fashion.

The most common approach has been to use channel hot-electron injection [39] to decrease the floating-gate voltage and to use Fowler-Nordheim tunneling [40] to increase the floating-gate voltage. To implement typical neural network learning rules, it is important to carefully control how much each synapse is modified. The perceptron learning rule, for example, adds/subtracts a scaled version of the input vector. Normalization, on the other hand, requires weight modifications that are proportional to the value of the weight itself. Due to the complexity of controlling the exponential tunneling and injection processes, it is more common to see floating-gate synapses used in the implementation of individual synapse modification (e.g., spike timing-dependent plasticity, STDP) or modified through feedback using off-chip control loops.

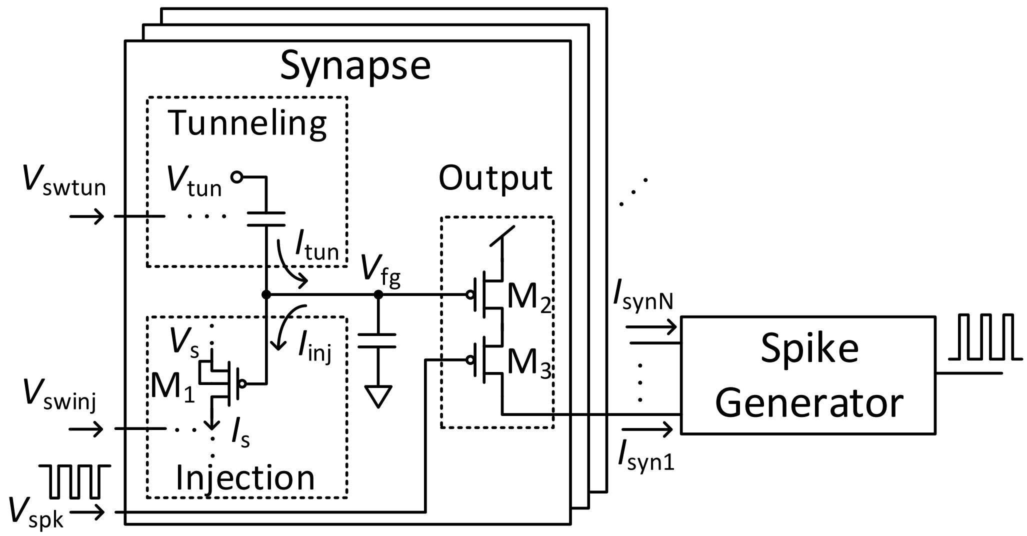


Fig. 30. Structure of the neuron circuit. Multiple synapses are connected to a spike generator and provide synaptic current. The synaptic weight of each synapse is store as the floating-gate voltage.

In this chapter, we explore the use of local feedback circuits that use the floating-gate voltage to control the rate of weight modification, allowing linear increases (for adding vectors) and exponential decreases (used in normalization) in weights. We demonstrate the addition of two vectors and weight normalization.

4.2 Circuit Implementation

The structure of our neuron circuit is shown in Fig. 30. In the synapse circuit, the synaptic weights are represented as synaptic currents (I_{syn}) that are stored as floating-gate voltages (V_{fg}). When a spike input is present, the synaptic output circuit is turned on and I_{syn} is sent to the spike generator. An integrate-and-fire spike generator integrates the synaptic currents from several synapses and fires a spike train with a frequency proportional to the total input currents. The synaptic weights are changed by changing V_{fg} during training.

4.2.1 Output Circuit

The equivalent synaptic output circuit is shown in Fig. 30. Transistor M_2 is operating in subthreshold region and M_3 is a switch that turns the output on and off according to an inverse spike input. When the input is low, the current output of the synapse is

$$I_{\mathrm{syn}} = \mathrm{S}_2 I_{0\mathrm{p}} e^{\frac{\kappa_{\mathrm{p}}(\mathrm{Vdd}-V_{\mathrm{fg}})}{V_{\mathrm{T}}}} = I_{\mathrm{C}} e^{\frac{-\kappa_{\mathrm{p}} V_{\mathrm{fg}}}{V_{\mathrm{T}}}} \tag{19}$$

$V_{\mathrm{T}} = \mathrm{kT/q}$ is the thermal voltage, $I_{0\mathrm{p}}$ is the reference current for the pFET when $V_{\mathrm{gs}} = V_{\mathrm{bs}} = 0V$, $\mathrm{S} = \mathrm{W/L}$ is the ratio of the transistor width and length. $0 < \kappa < 1$ is a parameter specific to the technology, and we will assume it is constant in this analysis. $I_{\mathrm{C}} = \mathrm{S}_0 I_{0\mathrm{p}} \exp(\kappa_{\mathrm{p}} \mathrm{Vdd}/V_{\mathrm{T}})$ is a constant current.

The synaptic current I_{syn} is defined as the synaptic weight w for the synapse and the firing frequency of the input spike train V_{spk} is representing the input strength. The synaptic weight is stored as the floating-gate voltage V_{fg} in a logarithmic fashion since I_{syn} is an exponential function of V_{fg} (19). To keep all the transistors in subthreshold region while having a significant output current, we keep V_{fg} in a small range from 3.9 V to 4 V. Due to the exponential nature of subthreshold transistors, the output current could still change up to 15-fold in this small range.

4.2.2 Injection Circuit

Hot-electron injection mechanism is used to decrease the floating-gate voltage and increase the synaptic weight. As discussed in the introduction, increasing the synaptic weight in a linear fashion gives many advantages. Increasing the synaptic weight linearly with time can be written as

$$w(t) = I_{\mathrm{syn}} = \mathrm{k}_w * t + \mathrm{b} \tag{20}$$

where k_w is the slope of the linear function and b is the intercept. Substituting in (19) and differentiating the equation with time show that to increase the weight linearly, the injection current I_{inj} should satisfy

$$I_{\mathrm{inj}} = -\mathrm{C}_{\mathrm{fg}}\frac{dV_{\mathrm{fg}}}{dt} = \frac{V_{\mathrm{T}}\mathrm{C}_{\mathrm{fg}}}{\kappa_{\mathrm{p}}} \cdot \frac{\mathrm{k}_w}{w(t)} \tag{21}$$

where C_{fg} is the total capacitance at the floating-gate node. Hence, the injection current would need to satisfy (21) to increase the synaptic weight linearly. For a pFET injection model shown in Fig. 30 in the injection box, the empirical model of the injection current [39] is

$$I_{\mathrm{inj}} = \alpha I_{\mathrm{s}} \exp\left[-\frac{\beta}{\left(V_{\mathrm{gd}} + \delta\right)^2} + \lambda V_{\mathrm{sd}}\right] \tag{22}$$

where I_{s} is the source current, V_{gd} is the gate-to-drain voltage and V_{sd} is the source-to-drain voltage. α, β, λ and δ are fit parameters and they are calculated based on a previously fabricated floating-gate chip. The source current I_{s} for M_1 can be written as

$$I_{\mathrm{s}} = \mathrm{S}_1 I_{0\mathrm{p}} e^{\frac{\kappa_{\mathrm{p}}(V_{\mathrm{s}} - V_{\mathrm{fg}})}{V_{\mathrm{T}}}} \tag{23}$$

To control the injection current and increase synaptic weight linearly, we modulate the source voltage V_{s} according to V_{fg} in this work. Combining (19), (21), (22) and (23), and separating V_{s} and V_{fg}, we get

$$V_{\mathrm{s}} = \mathrm{K}_1 + 2\kappa_{\mathrm{p}}\mathrm{K}_0 V_{\mathrm{fg}} + \beta V_{\mathrm{T}}\mathrm{K}_0/\left(V_{\mathrm{fg}} - V_{\mathrm{d}} + \delta\right)^2 \tag{24}$$

where the constants are

$$\mathrm{K}_0 = 1/\left(\kappa_{\mathrm{p}} + \lambda V_{\mathrm{T}}\right), \mathrm{K}_1 = V_{\mathrm{T}}\mathrm{K}_0 \left[\ln\left(\frac{V_{\mathrm{T}}\mathrm{C}_{\mathrm{fg}}\mathrm{k}_w}{\alpha \mathrm{S}_1 I_{0\mathrm{p}}\kappa_{\mathrm{p}} I_{\mathrm{C}}}\right) + \lambda V_{\mathrm{d}}\right]$$

The effective slope $\mathrm{k}_{\mathrm{inj}}$ for V_{s} as a function of V_{fg} can be calculated as

$$\mathrm{k}_{\mathrm{inj}} = \frac{dV_{\mathrm{s}}}{dV_{\mathrm{fg}}} = 2\mathrm{K}_0\left[\kappa_{\mathrm{p}} - \frac{\beta V_{\mathrm{T}}}{\left(V_{\mathrm{fg}} - V_{\mathrm{d}} + \delta\right)^3}\right] \tag{25}$$

Plugging in the number shows that when $V_{\mathrm{d}} = 0$, $\mathrm{k}_{\mathrm{inj}}$ changes less than 0.6% over the operating range of V_{fg} (3.9 V to 4 V). Hence, (25) can be approximated to a linear function

$$V_{\mathrm{s}} \approx \mathrm{K}_1 + \mathrm{k}_{\mathrm{inj}} V_{\mathrm{fg}} \tag{26}$$

The intercept K_1 depends on the desired rate of increase in the synaptic weight k_w while the slope $\mathrm{k}_{\mathrm{inj}}$ is fixed. Hence, we can change k_w by only changing the intercept K_1 without changing the slope $\mathrm{k}_{\mathrm{inj}}$. The circuit implementation of this function is shown in Fig. 31.

When the input V_{swinj} is set to Vdd, the switch M_{14} is on and the drain voltage of the injection transistor M_7 is set to ground, turning on the injection mechanism. When the input V_{swinj} is set to ground, the drain voltage of M_7 is set to a high voltage V_{high} thus turning off the injection. M_{10}-M_{12} are used as mirror transistors to M_7, M_{13} and M_{14} with inverted input to cancel the coupling effect from the change in the drain voltage on V_{fg}. M_4-M_5 are two transistors with the same size acting as a current mirror.

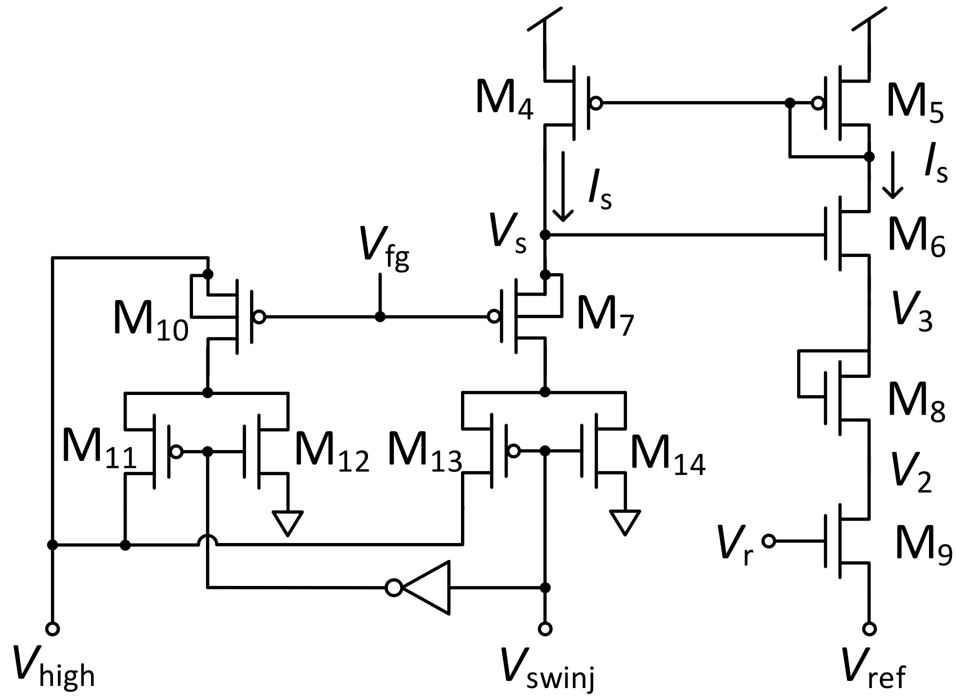


Fig. 31. Implementation of the injection circuit. V_s is modulated during the injection ($V_{\mathrm{swinj}} = \mathrm{Vdd}$) to decrease V_{fg} logarithmically, resulting in a linear increase in the synaptic weight.

M_9 is abovethreshold transistor operating in the triode region and acting like a resistor with resistance R decided by V_r and V_{ref}. Transistors M_6-M_8 are in the subthreshold region and in saturation. When the injection is on ($V_{\mathrm{swinj}} = \mathrm{Vdd}$), the relationship between V_s and V_{fg} can be written as

$$V_s = \mathrm{C}_2\left(V_T \ln \mathrm{C}_1 - \frac{1}{\kappa_n + 1} V_{\mathrm{ref}}\right) + \kappa_p \mathrm{C}_2 V_{\mathrm{fg}} - \frac{\mathrm{C}_2 \mathrm{R}}{\kappa_n + 1} I_s(V_{\mathrm{fg}}) \qquad (27)$$

where

$$\mathrm{C}_1 = \frac{\mathrm{S}_3 I_{0n}}{\mathrm{S}_4 I_{0p}} \left(\frac{\mathrm{S}_3}{\mathrm{S}_5}\right)^{-1/(\kappa_n + 1)}$$

$$\mathrm{C}_2 = (\kappa_n + 1)/(\kappa_p \kappa_n + \kappa_p - \kappa_n{}^2)$$

$S = W/L$ is the ratio of the transistor width and length and I_s is a function of V_{fg}. If R=0, the third term in (27) is 0 and V_s is a linear function of V_{fg} with a fixed slope of $\kappa_p C_2$. Since this slope might not match well with the desired slope k_{inj} in (26), we modulate the slope with the extra resistance R added by M9. Here we define $k_{fit}(V_{fg}) = dV_s/dV_{fg}$ as the slope function and differentiating (27) with V_{fg} two times gives

$$\frac{dk_{fit}}{dV_{fg}} = \frac{d^2V_s}{dV_{fg}^2} = -\frac{(\kappa_p C_2 - 1)C_3 R \cdot I_s'}{(1 + C_3 R \cdot I_s)^2} \tag{28}$$

Because $\kappa_p C_2 > 1$, it can be proven that $k_{fit} > 1$. This means that V_s changes faster then V_{fg}, making $I_s(V_{fg})$ a monotonically increasing function with V_{fg} and $I_s' > 0$. Hence, $dk_{fit}/dV_{fg} < 0$ and the slope k_{fit} is a monotonically decreasing function with V_{fg}. The bigger the R is, the faster k_{fit} drops. In our work, the value of k_{fit} at R=0 is bigger than the desired slope k_{inj} in (26) and we can use R to drop k_{fit} to approximate k_{inj} over the operating range of V_{fg}. The rate of increase in the synaptic weight k_w (20) can be changed through V_{ref} according to (27).

4.2.3 Tunneling Circuit

A tunneling mechanism is used to remove electrons from the floating-gate and increase the floating-gate voltage to decrease the synaptic weight. As discussed in the introduction, a method to scale down the weight vector magnitude without changing the direction is desirable. Scaling down the weight vector means proportionally decaying the weights along different dimensions. Since the synaptic weight has an exponential relationship with V_{fg} (19), changing each floating gate voltage by the same amount results in scaling each weight by the same factor. In this case, increasing V_{fg}

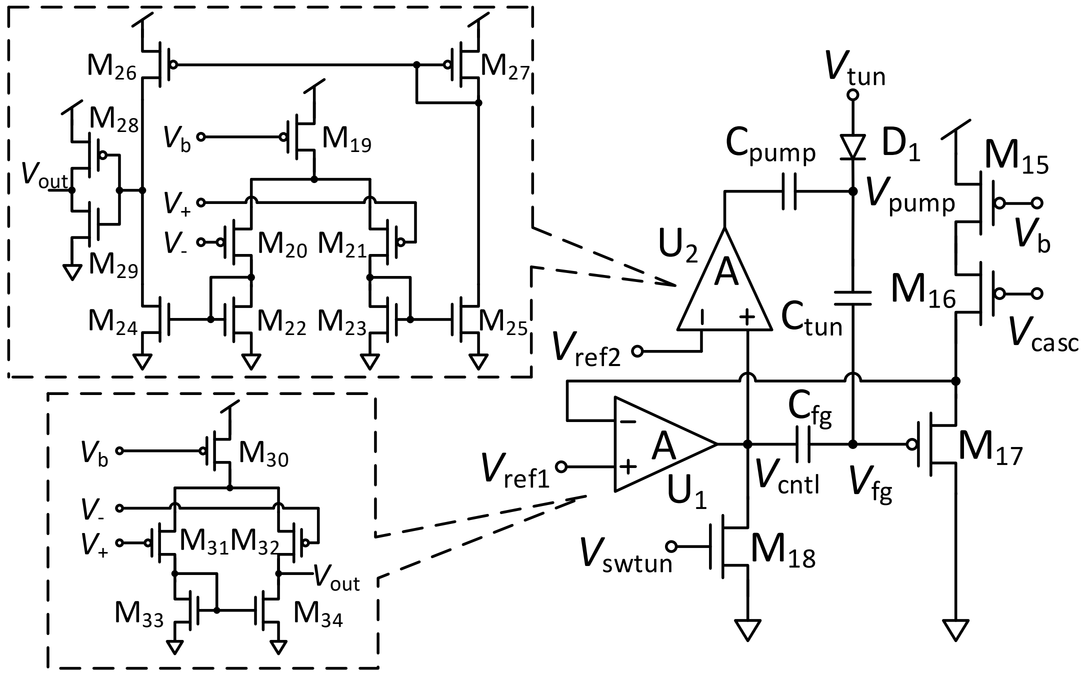


Fig. 32. Tunneling circuit. V_{fg} is held constant while tunneling to create a constant tunneling current, resulting in an exponential decay on the synaptic weight.

linearly with a constant tunneling current results in decaying the weight exponentially. According to [39], the tunneling current can be written as

$$I_{\mathrm{tun}} = -I_{\mathrm{tun0}} W L \mathrm{e}^{-\frac{V_{\mathrm{f}}}{V_{\mathrm{ox}}}} \tag{29}$$

where I_{tun0} is a reference current, V_{f} is a constant that varies with oxide thickness and V_{ox} is the voltage across the tunneling oxide. W and L are the width and length of the tunneling pFET. To keep I_{tun} constant, we can simply keep V_{ox} constant.

The tunneling circuit implementation (Fig. 32) is based on a method introduced by [41] for hot-electron injection. In [41], the injection current was kept constant by holding a constant floating-gate voltage, source voltage and source current. We adopt this method to hold V_{fg} constant while tunneling so that V_{ox} is kept constant. When $V_{\mathrm{swtun}} = 0$, V_{cntl} is holding V_{fg} to be constant through the amplifier U_1. At the same

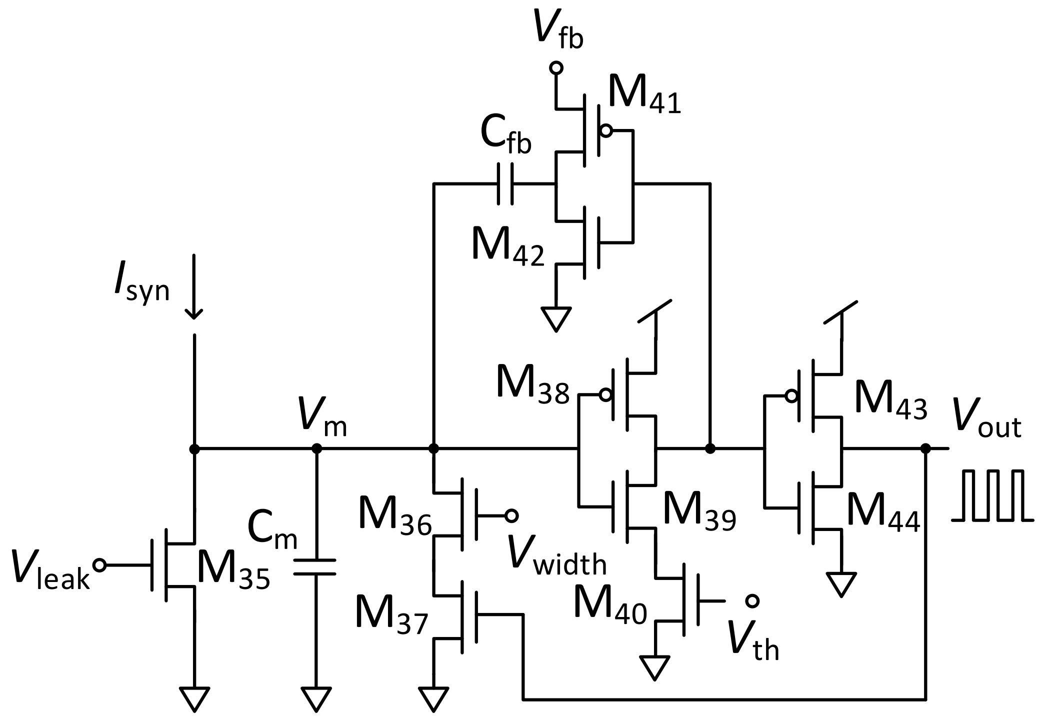


Fig. 33. Integrate-and-fire spike generator circuit generates an output spike train with a firing frequency representing the total input synaptic current.

time, when $V_{\mathrm{cntl}} > V_{\mathrm{ref2}}$ the output of the voltage comparator U2 will be high, pumping up V_{pump} through $\mathrm{C}_{\mathrm{pump}}$ and starting the tunneling process with a constant tunneling current. The voltage comparator U_2 is for automatically stopping the tunneling while V_{fg} reach a certain threshold and this threshold can be adjusted by changing V_{ref2}. When $V_{\mathrm{swtun}} = 1$, V_{cntl} is held at ground. In this case, V_{fg} is released and the pump is not active, thus stopping the tunneling.

4.2.4 Spike Generator Circuit

The leaky integrate-and-fire spike generator circuit (Fig. 33) generates the output spike train. In steady state, the firing frequency of the spike train is proportional to the charging rate of V_{f}. When the leakage current is set to 0 ($V_{\mathrm{leak}} = 0$), the firing frequency is proportional to the input synaptic current I_{syn}.

4.3 Experimental Results

The chip is fabricated in commercially available 0.5um CMOS technology. A microcontroller is used to control the tunneling and injection switches and to measure the firing frequency of the output spike train.

4.3.1 Single Synapse Testing

To show that the synaptic weight, represented as the steady-state synaptic current I_{syn} in Fig. 30, can be increased linearly (in time) through the injection circuit, the injection is turned on with a fixed pulse width for 50 pulses. Tunneling is then turned on with a fixed pulse width for another 50 pulses to show the exponential decay. After each training pulse, to probe the new value of synaptic current, the spike input V_{spk} is held low for the synaptic current to reach steady state and the corresponding neuron output firing frequency is measured. Since the leakage current of the spike generator is set to zero during the test, the firing frequency is proportional to the synaptic current. The synaptic currents can then be inferred from the neuron firing frequency. The results for two synapses are shown in Fig. 34. The result for each synapse is tested individually while other synapses were turned off during testing. As is shown in Fig. 34, both the linear increase and the exponential decay fit the corresponding function accurately. Mismatch between transistors in the different synapses produce different injection and tunneling rates, and this mismatch can be compensated by calibrating the tunneling and injection pulse widths for different synapses. After compensation, the curves for different synapses match each other (Fig. 34).

If a linear decay in the synaptic weight is desired, we can modulate the width of the tunneling pulse. With the information of the current synaptic weight, the tunneling pulse width needed to achieve linear decay can be calculated as

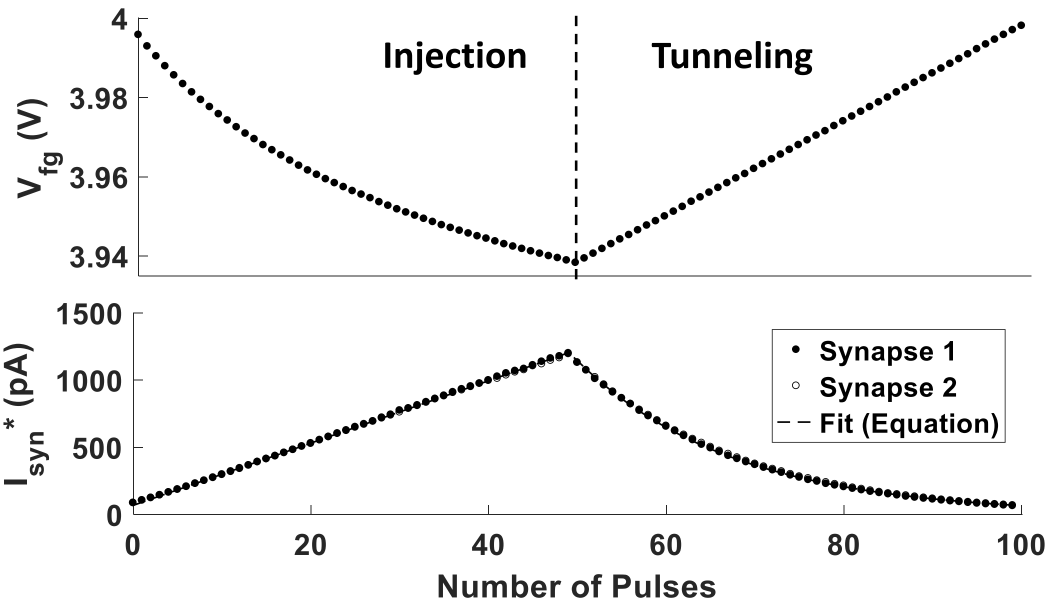


Fig. 34. (a) Floating-gate voltage of synapse 1 as a function of number of training pulses during injection followed by tunneling. As expected, V_{fg} drops logarithmically during the injection phase and rises linearly during tunneling phase. (b) Two synaptic currents rise linearly in the injection phase and decay exponentially in the tunneling phase. The dotted line is the equation fit with linear fit for injection and exponential fit for tunneling. The injection pulse widths are 1.67s for synapse 1 and 1.8s for synapse 2, and the tunneling pulse widths are 0.23s for synapse 1 and 0.2s for synapse 2. The curves for the two synapses and the fitted curve are on top of each other which makes it hard to see.

*The synaptic current is inferred through the output firing frequency.

$$T = \frac{1}{r} \log \frac{I_{\text{syn}}}{I_{\text{syn}} - \Delta I} \tag{30}$$

where r is the fitting parameter at the exponent for the exponential decay, ΔI is the desired change in the synaptic weight and T is the needed pulse width. In our test, the tunneling window is calculated and modulated using a microcontroller. As is shown in Fig. 35, the synaptic weight can be linearly decayed by modulating the tunneling pulse width. Combined with the linear increase, we can change the synaptic weight by a specific amount in both positive and negative direction.

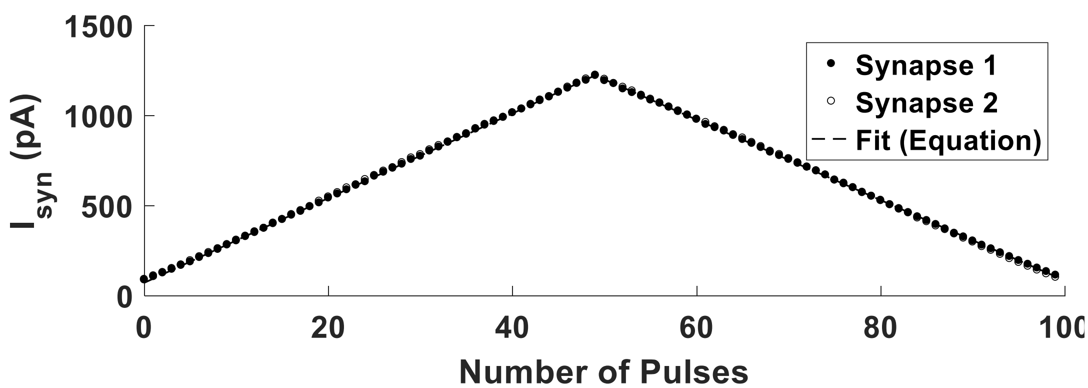
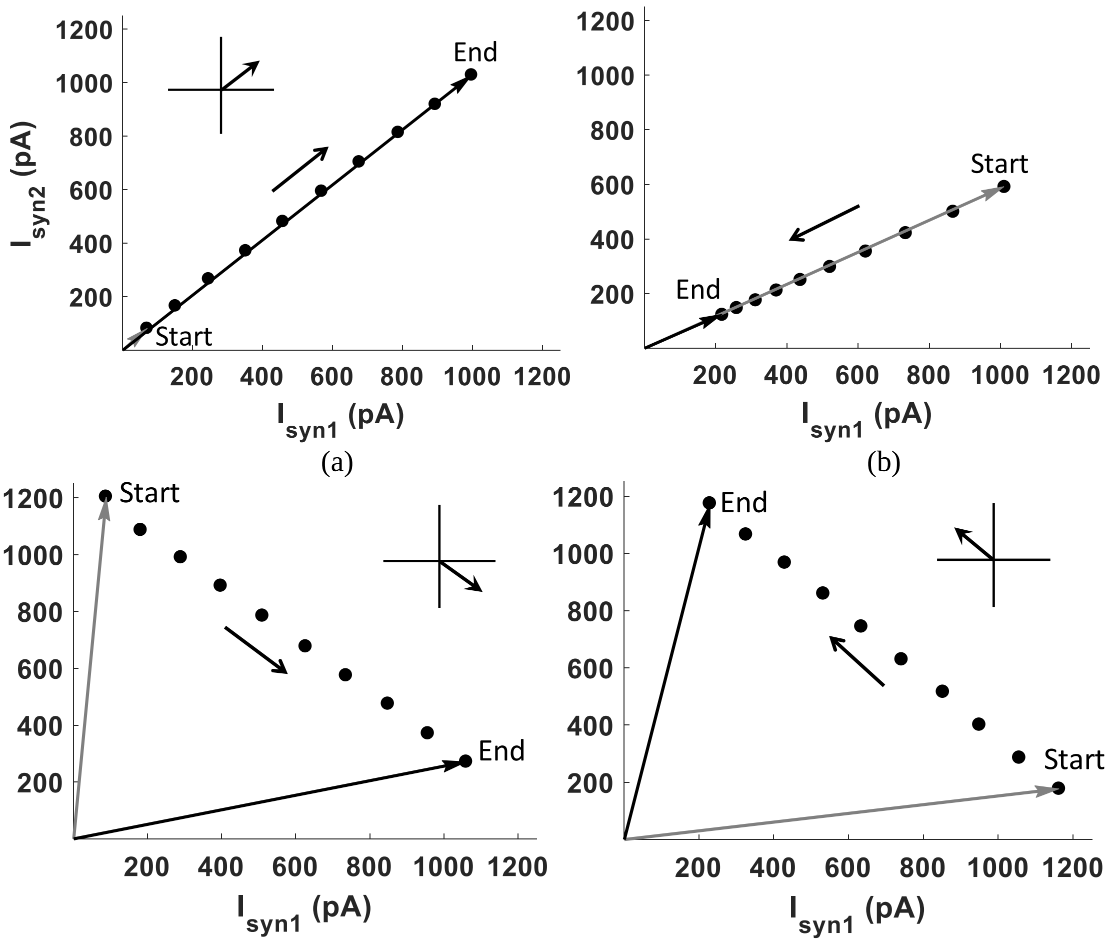


Fig. 35. Inferred synaptic currents as a function of the number of training pulses using linear increase and decay. The dotted line is the linear fit for both injection phase and tunneling phase.

4.3.2 Weight Vector Manipulation

Adding a vector to the weight vector and scaling it down without changing its direction are two commonly desired functions in neural network training. Here we're



(c) (d)

Fig. 36. Weight vector addition and normalization. Panels (a), (c), and (d) show the addition of a 2-quadrant, 2-D weight vector (black dot) and a 4-quadrant, 2-D vector (inset). Additionally, the starting weight vector is shown with a gray arrow and the final weight vector is shown with a black arrow. Panel (b) shows normalization through proportional scaling along each dimension. Panel (a) shows increases in both X and Y directions, panel (c) shows increases in the X direction, but decreases in the Y direction, and panel (c) shows increases in Y direction, but decreases in the X direction.

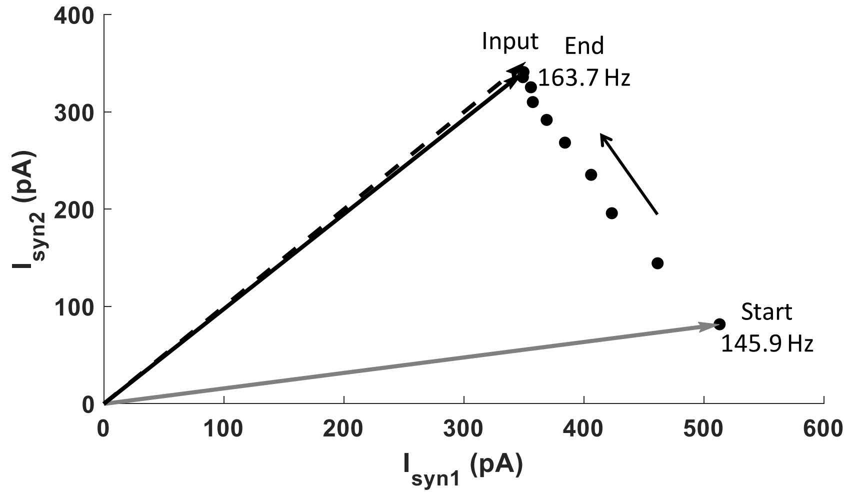
demonstrating these functionalities through two synaptic weights, which are representing a 2-dimensional weight vector.

Since the injection circuit can increase the synaptic weight linearly, we can add a positive vector to the weight vector by simply turning on the injection and keeping it for a period proportional to the values of the added vector. As is shown in Fig. 36(a), the 2-D weight vector is added by a constant vector after every step. Scaling the weight vector down without changing its direction is also straightforward because the synaptic weight can be exponentially decayed through the tunneling circuit. By turning on the tunneling and keeping it for the same period for both synapses, the 2-D weight vector was scaled down, as is shown in Fig. 36(b).

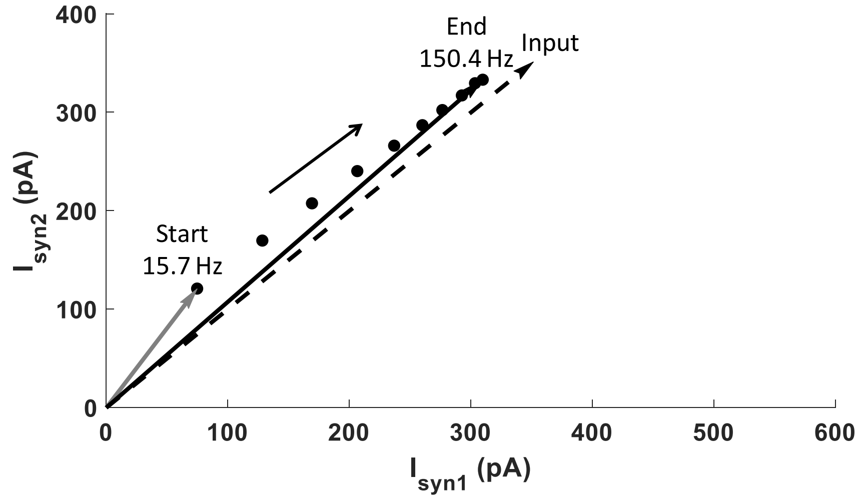
Adding a vector with negative values, however, is also often desired when training. Using the linear decay method introduced in section A to add negative values to the synaptic weights, we can add any vectors to the 2-D weight vector with external control on the tunneling time (Fig. 36(c)(d)).

4.3.3 Unsupervised Learning

Here we demonstrate the ability for the circuits to learn an input vector. The unsupervised training method we're using can be written as $w = (1 - \mathrm{p})w + \mathrm{p} * x$, where w is the weight vector, x is the input vector and p is a mixing factor ($\mathrm{p} < 1$). The input vector is entered to the synapses through the firing frequency of the spike input V_{spk} and the neuronal response is represented as the firing frequency of the output spike train. The microcontroller runs the training algorithm and controls the tunneling and injection for each synapse. The decaying part $(1 - \mathrm{p})w$ is achieved by turning on the tunneling with a pulse width decided by p. Adding vector $\mathrm{r} * x$ to the decayed



(a)



(b)

Fig. 37. Two examples of the training process with different initial weight vectors. The weight vector (dots in the plot) starts at the location shown by the gray arrow. As training proceeds, the weight vector approaches the input vector (dashed arrow). After nine iterations, the weight vector is at the location shown by the black arrow. The wide arrows show the evolution of the weight vectors during training. The corresponding neuronal responses (firing frequency) for the weight vector are shown with the starting and ending vectors. As expected, the neuronal responses increase with training.1.67s for synapse 1 and 1.8s for synapse 2, and the tunneling pulse widths are 0.23s for synapse 1 and 0.2s for synapse 2. The curves for the two synapses and the fitted curve are on top of each other which makes it hard to see.

weight vector are done by using the method described in section B. The result (Fig. 37) shows that the 2-D weight vector is learning the input vector and the neuronal response to the input vector is getting stronger as the training goes.

4.4 Conclusion

The CMOS floating-gate can be an excellent storage medium for artificial neural network synapse memories once they have been programmed. Changing their value, however, typically involves extensive circuitry to control the mechanisms for electron injection (e.g., hot-electron injection) and electron removal (e.g., Fowler-Nordheim tunneling). In this chapter, we describe a circuit that uses the floating-gate voltage to maintain a fixed rate of weight increase (adding vectors) or proportional decay (normalization) on the synaptic weight. Using these circuits, simple perceptron learning is demonstrated.

Chapter 5: Conclusion

This research aimed to explore two neuromorphic mechanisms involved in the ability of bats to pursue a goal while avoiding obstacles using ultrasonic echolocation. The first work presented in this dissertation focused more on the level of echolocation. In that work, it was noted that the equations for the neuron membrane potential suggest the input signals could be power-law compressed to significantly increase the input dynamic range without changing the operation. Successful implementation of both square-root compression and cube-root compression of the echo envelope amplitude was described to detect the angle of arrival of an echo from an object placed over a wide range of distances.

The work on the Curved Openspace algorithm and its spike-latency neural network focused on the level of algorithm and control. The analog version of the Curved Openspace algorithm overcame several limitations of the original Openspace algorithm [20], and the robotic implementation showed its capability of navigating through cluttered environments using a realistic sonar system. Instead of using a large population of neurons to represent analog signals with spike rates, the proposed neural network of the Curved Openspace algorithm uses spike-latency to represent signal values for computation. The neural network takes advantage of the intrinsic time code of sonar systems where low echo latency represents a close object and high echo latency represents a distant one. A spike-latency-based implementation of evaluation memory was proposed to receive spikes with different latencies and reproduce the spikes with the same latencies even when the sonar does not receive new information.

This dissertation also described a CMOS floating-gate circuit that can maintain a fixed rate of weight increase or proportional decay on the synaptic weight.

An issue with the proposed spiking neural network implementation of the Curved Openspace algorithm is that it runs extremely slow even on powerful CPUs (Intel Xeon Cascade Lake on Google Cloud). It limited the amount of data that was collected and made it impractical to do a comprehensive analysis of the performance of the spiking neural network. It also prevented a robotic implementation in real-time. Future work can be done on a neuromorphic VLSI chip to vastly improve the running speed of the proposed spiking neural network, allowing the network to run in real-time on a mobile platform.

Appendices

In Fig. 2(a), transistors M_1 and M_2 (W/L in microns) are 2.8/5.4, M_8 and M_{12} are 12.6/2.1, M_{19} is 1.4/1.4, and all the other transistors are 2.1/2.1. C_m=300fF and C_{fb}=30fF. The voltages are V_{casc1}=1.5V, V_{casc2}=2.5V, V_b=4.12V and V_{thresh}=0.49V.

In Fig. 2(b), all the transistors are 2.1/2.1 and V_{casc3}=3V.

In Fig. 3, LM6134 op-amps were used for U_1-U_5 and U_7-U_9, an LM339 voltage comparator was used for U_6, a 2N3906 PNP transistor was used for Q_1 and 1N4148 diodes were used for D_{01}, D_{02} and D_{11} to D_{1N}. R_1, R_3 and R_6 are 4.5kΩ, R_2 and R_{14} are 80kΩ, R_4=470kΩ, R_5=8.1kΩ, R_7=150kΩ, R_8=2.79kΩ, R_9=2.21kΩ, and R_{10}-R_{13} and R_{15} are 28kΩ. C_1=47pF, C_2=100nF, C_3=470pF, C_4=22nF, C_5 and C_6 are 470pF. The reference voltages are V_{ref1}=2.2V and V_{ref2}=1.65V for the square-root configuration (N=2) or V_{ref2}=1.13V for the cube-root configuration (N=3).

Bibliography

[1] Sändig, Sonja, Hans-Ulrich Schnitzler, and Annette Denzinger. "Echolocation behaviour of the big brown bat (Eptesicus fuscus) in an obstacle avoidance task of increasing difficulty." Journal of Experimental Biology 217.16 (2014): 2876-2884.

[2] Simmons, James A. "The resolution of target range by echolocating bats." The Journal of the Acoustical Society of America 54.1 (1973): 157-173.

[3] Peff, T. C., and James A. Simmons. "Horizontal-Angle Resolution by Echolocating Bats." The Journal of the Acoustical Society of America 51.6B (1972): 2063-2065.

[4] W. M. Masters, S. C. Jacobs, and J. A. Simmons, "The structure of echolocation sounds used by the big brown bat Eptesicus fuscus: Some consequences for echo processing," J. Acoust. Soc. Am., vol. 89, no. 3, 1991, doi: 10.1121/1.400660.

[5] D. McAlpine, D. Jiang, and A. R. Palmer, "A neural code for low-frequency sound localization in mammals," Nature Neuroscience, vol. 4, no. 4, pp. 396–401, 2001.

[6] J. M. Groh, K. A. Kelly, A. M. Underhill, "A monotonic code for sound azimuth in primate inferior colliculus" J Cogn Neuroscience, vol. 15, no. 8, pp. 1217-1231, 2003.

[7] R. Z. Shi and T. K. Horiuchi, "A Neuromorphic VLSI Model of Bat Interaural Level Difference Processing for Azimuthal Echolocation," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 54, no. 1, pp. 74–88, 2007.

[8] T. K. Horiuchi, C. Bansal, and T. M. Massoud, "Binaural intensity comparison in the echolocating bat using synaptic conductance," 2009 IEEE International Symposium on Circuits and Systems (ISCAS'04), May 24- 27, 2009.

[9] Wen, Chenxi, and Timothy K. Horiuchi. "Power-Law Compression Expands the Dynamic Range of a Neuromorphic Echolocation System." 2018 IEEE Biomedical Circuits and Systems Conference (BioCAS). IEEE, 2018.

[10] T. Horiuchi, "'Seeing' in the dark: neuromorphic VLSI modeling of bat echolocation," IEEE Signal Processing Magazine, vol. 22, no. 5, pp. 134–139, 2005.

[11] A. Aggarwal, "Azimuthal Sound Localisation with Electronic Lateral Superior Olive," Engineering Applications of Neural Networks, pp. 231–240, 2015.

[12] S. A. Kick and J. A. Simmons, "Automatic Gain Control in the Bat's Sonar Receiver and the Neuroethology of Echolocation", J Neuroscience, vol. 4, no. 11, pp. 2725-2737, 1984

[13] Latombe, Jean-Claude. Robot Motion Planning: Edition en anglais. Springer Science & Business Media, 1991.

[14] Braitenberg, Valentino. Vehicles: Experiments in synthetic psychology. MIT press, 1986.

[15] Khatib, Oussama. "Real-time obstacle avoidance for manipulators and mobile robots." Autonomous robot vehicles. Springer, New York, NY, 1986. 396-404.

[16] Borenstein, Johann, and Yoram Koren. "Real-time obstacle avoidance for fast mobile robots." IEEE Transactions on systems, Man, and Cybernetics 19.5 (1989): 1179-1187.

[17] Rostami, Seyyed Mohammad Hosseini, et al. "Obstacle avoidance of mobile robots using modified artificial potential field algorithm." EURASIP Journal on Wireless Communications and Networking 2019.1 (2019): 70.

[18] Borenstein, Johann, and Yoram Koren. “The vector field histogram-fast obstacle avoidance for mobile robots.” IEEE transactions on robotics and automation 7.3 (1991): 278-288.

[19] Ulrich, Iwan, and Johann Borenstein. “VFH+: Reliable obstacle avoidance for fast mobile robots.” Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146). Vol. 2. IEEE, 1998.

[20] Horiuchi, Timothy K. “A spike-latency model for sonar-based navigation in obstacle fields.” IEEE Transactions on Circuits and Systems I: Regular Papers 56.11 (2009): 2393-2401.

[21] Simmons, Reid. “The curvature-velocity method for local obstacle avoidance.” Proceedings of IEEE international conference on robotics and automation. Vol. 4. IEEE, 1996.

[22] Fox, Dieter, Wolfram Burgard, and Sebastian Thrun. “The dynamic window approach to collision avoidance.” IEEE Robotics & Automation Magazine 4.1 (1997): 23-33.

[23] Molinos, Eduardo J., Ángel Llamazares, and Manuel Ocaña. “Dynamic window based approaches for avoiding obstacles in moving.” Robotics and Autonomous Systems 118 (2019): 112-130.

[24] Yasin, Jawad N., et al. "Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches." IEEE Access 8 (2020): 105139-105155.

[25] Pérez-Carabaza, Sara, et al. "UAV trajectory optimization for Minimum Time Search with communication constraints and collision avoidance." Engineering Applications of Artificial Intelligence 85 (2019): 357-371.

[26] Bry, Adam, and Nicholas Roy. "Rapidly-exploring random belief trees for motion planning under uncertainty." 2011 IEEE international conference on robotics and automation. IEEE, 2011.

[27] Bry, Adam, Abraham Bachrach, and Nicholas Roy. "State estimation for aggressive flight in GPS-denied environments using onboard sensing." 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012.

[28] Ghose, Kaushik, and Cynthia F. Moss. "The sonar beam pattern of a flying bat as it tracks tethered insects." The Journal of the Acoustical Society of America 114.2 (2003): 1120-1131.

[29] Gaudette, Jason E., et al. "High resolution acoustic measurement system and beam pattern reconstruction method for bat echolocation emissions." The Journal of the Acoustical Society of America 135.1 (2014): 513-520.

[30] Surlykke, Annemarie, Kaushik Ghose, and Cynthia F. Moss. "Acoustic scanning of natural scenes by echolocation in the big brown bat, Eptesicus fuscus." Journal of Experimental Biology 212.7 (2009): 1011-1020.

[31] Cheely, Matthew, and Timothy Horiuchi. "Analog VLSI models of range-tuned neurons in the bat echolocation system." EURASIP Journal on Advances in Signal Processing 2003.7 (2003): 598240.

[32] Feng, A. S., J. A. Simmons, and S. A. Kick. "Echo detection and target-ranging neurons in the auditory system of the bat Eptesicus fuscus." Science 202.4368 (1978): 645-648.

[33] Horiuchi, Timothy K., and Matthew Cheely. "A systems view of a neuromorphic VLSI echolocation system." 2007 IEEE International Symposium on Circuits and Systems. IEEE, 2007.

[34] Häusser, Michael, et al. "The beat goes on: spontaneous firing in mammalian neuronal microcircuits." Journal of Neuroscience 24.42 (2004): 9215-9219.

[35] Lee, Choongkil, William H. Rohrer, and David L. Sparks. "Population coding of saccadic eye movements by neurons in the superior colliculus." Nature 332.6162 (1988): 357-360.

[36] Hodgkin, Alan L., and Andrew F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve." The Journal of physiology 117.4 (1952): 500-544.

[37] Simmons, James A. "The resolution of target range by echolocating bats." The Journal of the Acoustical Society of America 54.1 (1973): 157-173.

[38] Peff, T. C., and James A. Simmons. "Horizontal-Angle Resolution by Echolocating Bats." The Journal of the Acoustical Society of America 51.6B (1972): 2063-2065.

[39] K. Rahimi, C. Diorio, C. Hernandez and M. D. Brockhausen, "A simulation model for floating-gate MOS synapse transistors," 2002 IEEE International Symposium on Circuits and Systems. Proceedings, Phoenix-Scottsdale, AZ, USA, 2002, pp. II-II.

[40] Fowler, Ralph Howard, and Lothar Nordheim. "Electron emission in intense electric fields." Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character 119.781 (1928): 173-181.

[41] C. Huang, P. Sarkar and S. Chakrabartty, "Rail-to-Rail, Linear Hot-Electron Injection Programming of Floating-Gate Voltage Bias Generators at 13-Bit Resolution," in IEEE Journal of Solid-State Circuits, vol. 46, no. 11, pp. 2685-2692, Nov. 2011.