UNIVERSITY OF CALIFORNIA,
IRVINE

Neurorobotic Models of Spatial Navigation and Their Applications

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Cognitive Science with a Concentration in Cognitive Neuroscience

by

Tiffany Hwu

Dissertation Committee:
Professor Jeffrey L. Krichmar, Chair
Professor Emre Neftci
Professor Mimi Liljeholm

2019

TABLE OF CONTENTS

Page

LIST OF FIGURES **v**

LIST OF TABLES **xii**

ACKNOWLEDGMENTS **xiii**

CURRICULUM VITAE **xiv**

ABSTRACT OF THE DISSERTATION **xvii**

1 Introduction **1**

2 A Self-Driving Robot Using Deep Convolutional Neural Networks on Neuromorphic Hardware **3**

2.1 Introduction . . . 3
2.2 Platforms . . . 5
2.2.1 IBM TrueNorth . . . 5
2.2.2 Android Based Robotics . . . 6
2.3 Methods and Results . . . 8
2.3.1 Data Collection . . . 8
2.3.2 Eedn Framework . . . 9
2.3.3 Data Pipeline . . . 13
2.3.4 Physical Connection of Platforms . . . 15
2.3.5 Testing . . . 15
2.4 Discussion . . . 16

3 Adaptive Robot Path Planning Using a Spiking Neuron Algorithm with Axonal Delays **18**

3.1 Introduction . . . 18
3.2 Methods . . . 21
3.2.1 Neuron Model and Connectivity . . . 21
3.2.2 Axonal Delays and Plasticity . . . 23
3.2.3 Spike Wave Propagation and Path Readout . . . 24

3.2.4 A* Path Planner . . . 26
3.2.5 Map of Environment . . . 26
3.2.6 Robot Hardware and Design . . . 27
3.2.7 Computation . . . 28
3.3 Results . . . 30
3.3.1 Path Planning Simulations . . . 30
3.3.2 Robotic Experiments . . . 31
3.4 Discussion . . . 33
3.4.1 Neurobiological Inspiration for the Spike Wavefront Algorithm . . . 35
3.4.2 Implementation of Spike Wavefront Planner on Neuromorphic Hardware 37
3.4.3 Comparison to Other Neurally Inspired Path Planning Approaches . 39
3.4.4 Simultaneous Path Planning and Mapping . . . 40
3.5 Conclusion . . . 41

4 A Neurobiological Model of Schemas and Memory Consolidation 42
4.1 Introduction . . . 42
4.2 Methods . . . 44
4.2.1 Contrastive Hebbian Learning . . . 44
4.3 Neural Model . . . 47
4.3.1 Indexing Stream . . . 47
4.3.2 Representation Stream . . . 48
4.3.3 Novelty and Schema Familiarity . . . 49
4.3.4 Experimental Design and Statistical Analyses . . . 53
4.4 Results . . . 56
4.4.1 Experiment 1 . . . 56
4.4.2 Experiment 2 . . . 57
4.4.3 Neural Activity . . . 59
4.4.4 Effects of neuromodulation . . . 63
4.5 Discussion . . . 65
4.5.1 Hippocampal Indexing . . . 66
4.5.2 Neuromodulation and Novelty Detection . . . 67
4.5.3 Interactions Between the Medial Prefrontal Cortex and Hippocampus 68
4.5.4 Relevance to Complex Spatial Navigation . . . 69

5 Applying a Neurobiological Model of Schemas and Memory Consolidation to Contextual Awareness in Robotics 70
5.1 Introduction . . . 70
5.2 Methods . . . 72
5.2.1 Neural Network Model . . . 72
5.2.2 Robot Interface . . . 72
5.2.3 Experiment Design . . . 76
5.3 Results . . . 77
5.4 Discussion . . . 79

6 Conclusion **85**
6.1 Future Directions 85
6.2 Summary 86

Bibliography **88**

LIST OF FIGURES

Page

2.1 A) Core connectivity on the TrueNorth. Each neuron on a core can be configured to connect to all, none, or an arbitrary set of input axons on the core. Neuron outputs connect to input axons on any other core in the system (including the same core) through a Network-on-Chip. B) The IBM NS1e board. Adapted from (Esser et al., 2016). 6

2.2 Left: Side view of CARLorado. A pan and tilt unit supports the Samsung Galaxy S5 smartphone, which is mounted on a Dagu Wild Thumper chassis. A plastic enclosure holds the IOIO-OTG microcontroller and RoboClaw motor controller. A velcro strip on top of the housing can attach any other small components. Top Right: Front view of CARLorado. Three front-facing sonars can detect obstacles. Bottom Right: Close-up of IOIO-OTG and motor controller. 7

2.3 Data collection setup. Video from the smartphone mounted on the robot was sent to the tablet through a Wi-Fi direct connection. A human operator used two joysticks on the touchscreen of the tablet to issue motor commands, which were sent to the phone through the same connection. With the joysticks, the operator was able to change the speed of moving and turning by changing the pulse width modulation signal sent to the motor controller. Video frames and their corresponding motor commands in the form of pulse width values were saved to the SD card on the smartphone. 8

2.4 Convolution of layers in a CNN on TrueNorth. Neurons in each layer are arranged in three dimensions, which can be convolved using a filter of weights. Convolution occurs across all three dimensions. The third dimension represents different features. The convolution can be divided along the feature dimension into groups (indicated by blue and yellow colors) that can be computed separately on different cores. Adapted from (Esser et al., 2016). . . . 10

2.5 The CNN classified images into three classes of motor output: turning left, moving forward, and turning right. Accuracy of training was above 90 percent. 11

2.6 Effect of the number of cores used on the accuracy of the CNN. One NS1e board contains 4096 cores. An accuracy above 85 percent was still maintained even with the network reduced to one quarter of a full chip. 11

2.7 Data pipeline for running CNN. Training is done separately using the Eedn MatConvNet package using Titan X GPUs. A Wi-Fi connection between the Android Galaxy S5 and IBM NS1e transmit spiking data back and forth, using the TrueNorth (TN) Runtime API. 12
2.8 Physical connection of TrueNorth NS1e and CARLorado. The NS1e is attached to the top of the housing of the electronics housing using velcro. The NS1e is powered by running connections from the motor controller within the housing. The motor controller itself is powered by a Ni-MH battery attached to the bottom of the robot chassis. 13
2.9 Mountain trail in Telluride, Colorado. Top: Google Satellite image of trail (highlighted) Imagery ©2016 Google. Bottom: Testing CNN performance. . 14

3.1 Spike wave propagation in simulation. The top panels show the network activity at timestep 3, the middle panels show network activity at timestep 18, and the bottom panels show the resulting path. The left side shows the test area without a road. The light blue is the surrounding region, which has a cost of 9. The dark blue depicts the location of an open grass field, which has a cost of 3. The right side shows the test area that takes into consideration a road, which has a cost of 1 and is shown in dark blue. The spike wave propagation is shown in yellow. The starting location is at the top middle of the map, and the goal location is at the bottom middle of the map. Note how the spike wave propagates faster when there is a low cost road present. . . . 24
3.2 In some instances, the collision of multiple spike waves generated inefficient paths (left). This was remedied by adding a second pass through the algorithm with a cost map containing just the paths from the first pass (right). 25
3.3 Google satellite image of Aldrich Park at the University of California, Irvine. Two sections of the park (boxed) were transformed into cost maps (Map 1 as bottom box and Map 2 as top box) for the spiking wave planner. Imagery ©2016 Google. 27
3.4 20 x 20 cost grids created from two areas of Aldrich Park. A.) An open area with uniform low cost in the traversable area, and high cost outside of this area. B.) The same area as A but with a lower cost for the surrounding road. C.) The same area as B but with obstacles denoting benches, bushes and trees near the road. D.) A second area in Aldrich Park with high cost for trees, low cost for asphalt roads, and medium cost for dirt roads. 28
3.5 Screenshot of app used for robot navigation. The screen displays a camera view overlaid with information about distance to the destination, bearing to the destination ranging 0-360 degrees from true north, heading direction (also ranging 0-360 degrees from true north), and the 2D cost grid. Colors on the grid ranging from dark blue to red indicated the costs of the grid, with tree locations marked at highest cost in red. The planned path of the robot is indicated in yellow and the current location of the robot is marked in green. The Grid button toggles the grid view on and off and the Auto button switches the robot into and out of autonomous navigation mode. 29

3.6 Experimental results for the spiking path planning algorithm on Map 1, with no unique costs encoded for the path. Start and end locations are noted by their row and column position on the cost map. Black lines indicate the planned route and the 4 colored lines indicate the actual route taken by the robot. Scale bars indicate the length of 10 meters along latitudinal and longitudinal axes, indicating the size of error threshold of our navigation strategy. Imagery ©2016 Google. 33
3.7 Experimental results for the spiking path planning algorithm on Map 1, with lower costs encoded for the path. Black lines indicate the planned route and the 4 colored lines indicate the actual route taken by the robot. Imagery ©2016 Google. 34
3.8 Experimental results for the spiking path planning algorithm on Map 1, with lower costs encoded for the path. Black lines indicate the planned route and the 4 colored lines indicate the actual route taken by the robot. Imagery ©2016 Google. 35
3.9 Experimental results for the spiking path planning algorithm on Map 1, with lower costs encoded for the path. Black lines indicate the planned route and the 4 colored lines indicate the actual route taken by the robot. Imagery ©2016 Google. 36

4.1 Overview of network. The light blue box contains the Indexing Stream, including the ventral hippocampus (vHPC) and dorsal hippocampus (dHPC). The light orange box contains the Representation Stream, including the cue, medial prefrontal cortex (mPFC), multimodal layer (AC), and action layer. Bidirectional weights between layers in the Representational Stream are learned via contrastive Hebbian learning (CHL). Weights from the Indexing Stream are trained using the standard Hebbian learning rule. Dotted lines indicate influences of the neuromodulatory area, which contains submodules of novelty and familiarity. Weights extend to these modules from the mPFC and dHPC. Neuromodulator activity impacts how often the vHPC and dHPC are clamped and unclamped while learning the task via contrastive Hebbian learning (CHL). 46
4.2 The four phases of a trial of training. A) In the Indexing Phase, the Indexing Stream forms indices of activity. Additionally, the activities of the novelty and familiarity modules of the neuromodulator are calculated, setting the ultimate activity of the neuromodulator to the product of these values. B) The Free Phase of CHL. C) The Clamped Phase of CHL. D) The Test Phase of a network for measuring performance during training and unrewarded probe tests. This is the same as the CHL Free Phase, except that mPFC activity is also calculated. 54
4.3 Timeline of experiments. A) Experiment 1 timeline. Schema A is trained for 20 days. Two new paired associations (PAs) are introduced on day 21. Lesioning occurs on day 22, and yet another two PAs are introduced on day 23. Probe tests are performed throughout. B) Experiment 2 timeline. Schema B is trained for 16 days, then Schema A is retrained for 7 days. 55

4.4 Results of replicating the first experiment of Tse et al. A) The performance over 20 trials, showing a gradual increase. B) Probe tests of trials 2, 9, 4, and 16, showing the proportion of activity of the correct well given a food cue compared to activity of the incorrect wells. C) Probe test after training the new PAs for 1 day, in which one of the new pairs is cued and the activities of the correct well, well of the other new pair, and original Schema A wells are compared. The new PAs were learned within one trial, as the dig time for the cued pair was significantly higher than the rest. D) Probe tests of Schema A after splitting into HPC-lesioned and control groups. Both conditions retained knowledge of the schema. E) Probe tests of the new PAs after splitting into HPC-lesioned and control groups. Both groups recalled the new PAs equally. F) Probe tests of Schema A after training another two new PAs. The HPC group could not learn. 58

4.5 Weight matrices of the network of one simulated rat after simulating all of the first experiment. Rows represent post-synaptic layers and columns represent presynaptic layers.Weights from the context pattern to mPFC show the development of a distinct schema pattern encoded with stronger weights where the wells are located. Weights from the mPFC to the AC show the effect of the gating, in which the mPFC neuron representing Schema A is associated with a set of neurons in the AC. Weights from the AC to action layer show that the actions are dependent on activity from select AC neurons, suggesting that the AC neurons are learning specific features useful for action selection. Weights going from the vHPC, cue, and action to the dHPC are displayed in one matrix to show clear encodings of triplets with one neuron from each of the three areas. Weights from mPFC to vHPC show that there is not necessarily a one-to-one mapping from a winning mPFC neuron to a vHPC neuron, but that the schema information is transferred in a distributed manner. . . . 59

4.6 Results of replicating the second experiment of Tse et al. A) The performance over 16 trials of training Schema B. The control group was able to learn the new schema while the HPC group was not. B) Probe test after training Schema B, confirming part A. C) Performance over 7 trials of retraining on Schema A. Both groups retained Schema A, though the control group recovered from a very minimal forgetting of the schema initially. D) Probe test after retraining on Schema A, confirming part C. 60

4.7 Weight matrices of the control network after simulating all of the second experiment. Rows represent post-synaptic layers and columns represent presynaptic layers. Weights from the context pattern to the mPFC now show two distinct schemas, with stronger patterns than seen in the first experiment. Weights from mPFC to the AC show two sets of gating patterns. Weights to the dHPC now show twice the amount of triplets as before, reflecting that triplets from two schemas are being encoded. 61

4.8 Neural activity for the network of a single simulated rat while performing the first and second experiments combined. Each colored line represents the activity of one neuron. Each vertical black line represents the separation of epochs into trials. Activity is measured before winner-take-all is applied. A) The mPFC activities for the first and second experiments are shown in sequence, with the training of Schema A and two instances of new PAs for the first experiment, and the training of Schema B and return of Schema A for the second experiment. When training on Schema A, one mPFC neuron is consistently chosen as winner, as its weight values increase over time. When new PAs are introduced, the winner remains the same, but decreases in activity. At the start of Experiment 2, Schema B is introduced and a new mPFC neuron wins. The number of epochs greatly increases due to the novelty. When Schema A is retrained, the neuron previously associated with schema A becomes active again. B) The neurons in vHPC follow the same general trend as the mPFC neurons. However, all neurons increase and decrease together, as they all take input from the winning mPFC neuron. C) Activity of the dHPC neurons for the first 100 epochs of the first trial is shown. As the density of switching of dHPC winners occurs at every epoch rather than at every trial, it is necessary to display the activity at the epoch level. At each epoch, a different dHPC neuron is selected to have its weights increase, as a different triplet is present each time. The activities of winning neurons gradually increase for 40 epochs, remaining stable afterwards. 62
4.9 Combining familiarity and novelty for neuromodulation. A) Familiarity increases as the network is trained on a schema. Due to the sigmoidal transfer function, familiarity activity is step-like, going from unfamiliar to familiar over some training. B) Novelty starts at a high value whenever there are new PAs, which occurs on the first trial when the schema is introduced, and on the 21st trial when two PAs are replaced. C) At each trial, the product of familiarity and novelty lead to an increase of activity when schema familiarity is high and novelty is high. 64
4.10 Neuromodulator activity and winning schemas. Each colored line represents the activity of the neuromodulator from an individual network out of 20. A) Neuromodulator activity expressed by the number of epochs trained per trial. The vertical black line separates experiments 1 and 2. On trial 21, the number of epochs rises sharply due to the combination of familiar schema and novel stimulus. B) Index of winning mPFC neuron in each trial of experiments 1 and 2. The index switches clearly each time the schema switches. 65
4.11 A) Performances of each condition. The blue line represents the original network and the remaining lines use a flat number of epochs as indicated in the legend. All conditions are able to learn the schema, but with different learning rates. B) Probe tests of each condition after introducing a new PA. The average number of epochs for each trial is displayed in parentheses. The performance of the probe test increases as more epochs are trained per trial. However, the original network can get a performance equivalent to the other conditions, but with far fewer epochs of training. 66

5.1 The Toyota Human Support Robot. A scanning lidar at the base allows for SLAM mapping of the environment. A combination of height, arm, and gripper controls allows for objects to be picked up and put down on various surfaces. An RGBD camera allows for object segmentation, identification, and localization. 73
5.2 The graphical user interface for controlling the HSR includes buttons for manual control of the robot as well as the following components for automated behavior: A) The head camera input of the HSR with object segmentation and detection. B) A map of the roamable area is displayed here. The red circle shows the current location of the HSR. Blue dots represent the locations of corresponding neurons in the action layer of the neural network. C) Retrieval commands are sent to the robot by entering the name of the object to be retrieved. D) A roam sequence is initiated by clicking this button, causing the HSR to explore the current room. 74
5.3 A graphical user interface for observing neural network activity. Network weights from training trials can be saved and loaded. The training of the current schema is initiated using a button. The current schema, visible contextual objects, and novelty activity are also displayed here. 75
5.4 Experimental setup for the classroom and breakroom schemas. Yellow dots represent manually-picked destinations for the robot to roam and scan the room during training. A) Room layout and paired associations trained in the classroom. The bottle, teddy bear, and apple were laid out in the specified locations. After training on this layout, the teddy bear was exchanged for the mouse, introducing novelty. The robot then trained on this environment. The book (starred) was not trained as a paired association, but was included as a contextual item during training. B) Physical setup of classroom and contextual items. C) Room layout and paired associations trained in the breakroom, which is adjacent to the classroom. The wine glass, cup, and apple were laid out on the central table. The banana (starred) was not trained as a paired association, but was included as a contextual item during training. D) Physical setup of breakroom and contextual items. 81
5.5 Sequence of actions in Experiment 3. The HSR was shown a banana, with nothing else in its field of view. The experimenter then placed the banana on the breakroom table, outside of the HSR's view. The robot went to the breakroom to pick up the banana and navigated to the drop off location to deposit it. 82
5.6 Performance on Experiments 1 and 2 on a population of n = 5. A) For Experiment 1, the activation of the correct location neuron of a cued object increased with training (blue line) and the retrieval time decreased (red line). When a novel object was introduced (Exp 1b), the location activation was still high and retrieval time was low, despite only having had one trial of training. B) Performance also improved over time when a novel schema was introduced in Experiment 2. When returning to the classroom schema (CR), performance of original objects and novel objects was retained. Points denote the mean performance and error bars denote the standard deviation of the population. 82

5.7 Trajectories of an individual robot at various stages of the experiments. Stars indicate the start of the trajectory, dots indicate the destination associated with the target object, and the tail end of the line indicates the final destination. A) The red line shows the trajectory of the robot when retrieving the bottle after Trial 4 of training in Experiment 1. B) The green line shows the the retrieval of an apple in Trial 4 of Experiment 2. C) The yellow line shows the trajectory of the robot retrieving the banana in Experiment 3. The blue line shows retrieval of the book. Although the entire area is accessible during retrieval of both objects, each object is associated with a prior schema, prompting the robot to search in the room corresponding to that schema. . . 83
5.8 Heatmap of action layer during Experiment 3. A) When the robot was presented with a book and asked to retrieve it, the action layer showed high activity for objects in the classroom schema. B) When the robot was presented with a banana and asked to retrieve it, the action layer showed high activity for objects in the breakroom. 83
5.9 Selected weights on one individual after the experiment. Each set of weights is depicted as a heatmap, with warmer colors representing higher values, rows representing the post-synaptic layer neurons, and columns representing the pre-synaptic layer neurons. 84

LIST OF TABLES

Page

3.1 Comparison between spike wavefront planner and A* planner in different environments 31
3.2 Fréchet distances (in meters) between planned route and actual route for spike wavefront planner 31

4.1 Parameters used in schema and memory consolidation model. 53

5.1 Parameters used in robot experiment. 72

ACKNOWLEDGMENTS

I owe my thanks to the following people:

My advisor, Professor Jeffrey Krichmar, for years of patient guidance. Professor Mimi Liljeholm and Professor Emre Neftci for serving on my dissertation committee and providing feedback to strengthen my work. Members of my second year exam and advancement committee, Professor Bruce McNaughton, Professor Michael Lee, and Professor Mark Steyvers, for helping me formulate my research focus.

Professor Philippe Gaussier, Professor Will Browne, Professor Nikil Dutt, and Dr. John Shepanski for mentoring me and sharing their expertise.

The Cognitive Anteater Robotics Laboratory. Emily Rounds, Alexis Craig, Michael Beyeler, Ting-Shuo Chou, Nicolas Oros, and Kristofor Carlson for welcoming me into the lab and showing me the ropes. Georgios Detorakis, Meropi Topalidou for advice on research and navigating academia. Hirak Kashyap, Xinyun Zou, Jinwei Xing, Kexin Chen, Stas Listopad, Colleen Chen, Timo Oess, and Alexandro Ladron for sharing their diverse and interesting viewpoints, collaborating on challenging projects, and going on crazy adventures. The undergraduate research assistants who worked alongside us and gave it their all.

The Telluride Neuromorphic Cognition Workshop for creating a vibrant atmosphere of collaboration, where half of my dissertation work was completed.

The funding sources that made my Ph.D. studies possible: Northrop Grumman Aerospace Systems, HRL Laboratories, Defense Advanced Research Projects Agency (DARPA), Toyota Motor North America, and the UCI School of Social Sciences.

My family, Shing-Shing, Shyang, Janet, Raisin, and extended relatives for raising me with the right foundation and supporting my dreams.

Dear friends, Connie, Felicia, Samie-Jo, Charling, Charles, Brian, and UCI Wushu for grounding me outside of academia.

Dr. Si-Yuan Kong for countless discussions and support, helping me climb out of ruts and stay sane.

CURRICULUM VITAE

EDUCATION

University of California, Irvine June 2019
Ph.D. in Cognitive Science Irvine, CA
- Concentration in Cognitive Neuroscience
Advisor: Prof. Jeffrey Krichmar
Relevant coursework: Cortical Neuroscience, Machine Learning, Computational Neuroscience, Systems Neuroscience, Cognitive Robotics, Brain-Inspired Machine Learning, Neuromorphic Engineering
Dissertation: Neurorobotic Models of Spatial Navigation and Their Applications

University of California, Irvine December 2016
M.S. in Cognitive Neuroscience Irvine, CA

University of California, Berkeley May 2014
B.A. in Computer Science Berkeley, CA
B.A. in Cognitive Science
- Concentration in Computational Modeling
- Graduated with High Honors
Relevant coursework: Artificial Intelligence, Computational Cognitive Science, Cellular and Molecular Neurobiology
Senior Thesis: Exploring the uses of psychological models of generalization in music recommendation
Regents and Chancellor's Scholar

WORK EXPERIENCE

Summer Intern Summer 2018 - Present
HRL Laboratories *Malibu, CA*

Designed and implemented neural networks for unsupervised learning of virtual lane estimation, enhanced existing hypothesizers of agent trajectories, and modified planning algorithms to tackle complex traffic scenarios.

Systems Engineering Intern Spring 2016 - Winter 2017
Northrop Grumman Corporation *Redondo Beach, CA*
Basic Research, Northrop Grumman Next

Improved contextual awareness in robotic applications with the Biological Processing and Sensing Group, developed spiking neural network algorithms for neuromorphic computing with a team of research scientists, designed neurorobotic demonstrations on the Pioneer P3-DX robot.

Software Engineer Intern Summer 2013
Ancestry.com *San Francisco, CA*
Data Services

Reported on Ancestry.com user segmentation using Hive, Hadoop, and R frameworks, queried and summarized search term frequencies in Hive for the analytics team, researched and presented RHadoop use cases to all developer teams.

SKILLS

Fluent in: Java, Python, Matlab, Android
Experience with: OpenCV, ROS, PyTorch, TensorFlow, C, C++, R, Hadoop, Hive, HTML/CSS, JavaScript, Node, MongoDB, Express, PHP, MySQL, Lisp, Max/MSP

PUBLICATIONS

T. Hwu and J. L. Krichmar. (In review). A neural model of schemas and memory consolidation. *Biological Cybernetics*.

J.L. Krichmar, T. Hwu, X. Zou, and T. Hylton. (2019). The advantage of prediction and mental imagery for goal directed behavior in agents and robots. *IET Cognitive Computations and Systems*.

T. Hwu, A. Y. Wang, N. Oros, and J. L. Krichmar. (2017). Adaptive robot path planning using a spiking neuron algorithm with axonal delays. *IEEE Transactions on Cognitive and Developmental Systems*.

T. Hwu, J. L. Krichmar, and X. Zou. (2017). A complete neuromorphic solution to outdoor navigation and path planning. Proceedings of *IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, Maryland.

T. Hwu, J. Isbell, N. Oros, and J. L. Krichmar. (2017). A self-driving robot using deep convolutional neural networks on neuromorphic hardware. *IEEE International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK.

M. Frank, T. Hwu, S. Jain, R.T. Knight, I. Martinovic, P. Mittal, D. Perito, I. Sluganovic, and D. Song. (2017). Using EEG-Based BCI Devices to Subliminally Probe for Private Information. *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, Dallas, TX.

RESEARCH EXPERIENCE

Graduate Student Researcher Fall 2014 - Present
Cognitive Anteater Robotics Laboratory *University of California, Irvine*
Advisor: Prof. Jeffrey Krichmar

Collaborated with HRL Laboratories and other universities on Super Turing Lifelong Learning Architecture (STELLAR) project.

Created systems models of human memory to tackle the problem of catastrophic forgetting in artificial neural networks.

Applied neural models to improve situational awareness in an outdoor search and rescue task for ground robots.

Developed novel functions on the Toyota Human Support Robot (HSR), collaborating with the UC Irvine School of Education to research the effectiveness of telepresence robots in classrooms.

Research Assistant Fall 2011 - Spring 2014
Computational Cognitive Science Lab *University of California, Berkeley*
Advisors: Joshua Abbott, Prof. Thomas Griffiths

Ran human subjects in experiments on causal induction and probabilistic reasoning, reimplemented data analyses, completed senior thesis on Bayesian music recommendation.

Research Assistant Summer 2012 - May 2013
Computer Security Lab *University of California, Berkeley*
Advisor: Prof. Dawn Song

Applied machine learning algorithms and signal processing to EEG data, ran subjects in EEG recording sessions, implemented a card matching game that covertly extracts EEG data to test the security risks of EEG brain computer interface devices, piloted eyetracking experiment testing gender differences in visual attention.

AWARDS

Summer Fellowship in Honor of Christian Werner. 2018
John I. Yellott Scholar Award - Honorable Mention. 2017
National Science Foundation Graduate Research Fellowship - Honorable Mention. 2016
School of Social Sciences Merit Fellowship. 2014

ABSTRACT OF THE DISSERTATION

Neurorobotic Models of Spatial Navigation and Their Applications

By

Tiffany Hwu

Doctor of Philosophy in Cognitive Science with a Concentration in Cognitive Neuroscience

University of California, Irvine, 2019

Professor Jeffrey L. Krichmar, Chair

Spatial navigation requires many parallel cognitive processes, from low-level perception to high-level planning and decision-making. Findings from neuroscience provide inspiration for spatial navigation in robotic applications, improving the efficiency and adaptability of the systems. For example, neuromorphic hardware mimics the computational strategy of the brain for event-driven, massively parallel processing of information. In this dissertation, a working demonstration of a ground robot performing road following in an outdoor environment is introduced, with control computations run on neuromorphic hardware. Next, a neuromorphic path planning algorithm that accounts for environmental costs is presented, using spiking wave propagation inspired by hippocampal function. On top of perception and path planning, a memory system is necessary for contextual awareness and adaptation when navigating in a dynamic environment. To address this, a model of the hippocampus, medial prefrontal cortex, and neuromodulatory areas is discussed, in which the model is able to rapidly learn new information when consistent with a familiar contextual schema and prevent catastrophic forgetting of previously learned tasks. The model is successfully demonstrated on the Toyota Human Support Robot, tasked with finding and retrieving objects in multiple contexts. In summary, these models and demonstrations show how neurobiological systems of navigation can be used to improve robotic navigation at many levels of processing, which in turn reveals more insights on how biological systems navigate.

Chapter 1

Introduction

The ability to navigate the world enables intelligent behavior. It allows agents to find resources and shelter, learn about the surrounding environment, interact with other agents, and carry out decisions. Navigation is a complex task, requiring the coordination of many cognitive functions. First, perceptual input must be processed to localize the agent in space. This perceptual information is then used in short-term reactive tasks such as obstacle avoidance and road following, as well as long-term tasks such as path planning. At an even higher level of processing, navigation is intertwined with the cognitive processes of memory and decision-making, which use navigation to discover the environment and select future navigation goals.

These levels of processing are challenges for both autonomous mobile robots and biological organisms. Therefore, knowledge of navigation in one area can benefit the other. For example, mobile robots are limited by size, weight and power constraints and are limited in the amount of on-board computing they can handle, whereas biological agents have developed energy efficient approaches to computation at fractions of the power that robots require. On the other hand, many brain processes related to higher order cognition in navigation are

difficult to study. The modular and physically accessible nature of robotic systems provides a platform for modeling the brain systems related to navigation, allowing us to examine the interactions between processing areas and motor output.

This dissertation covers a variety of neurorobotic models of spatial navigation that exhibit the benefits of an interdisciplinary study between biological and artificial navigation systems. Chapter 2 shows how neuromorphic hardware can run on a mobile platform to process perceptual information into motor controls in a closed loop system. Chapter 3 continues to show how neuromorphic algorithms extend to path planning in dynamic environments. Chapter 4 transitions to a model of schemas and memory consolidation that addresses how agents are able to rapidly acquire new skills in dynamic environments without catastrophically forgetting older tasks. Chapter 5 demonstrates the effectiveness of this model on a robot performing object retrieval in two spatial contexts. Finally, Chapter 6 discusses future directions and conclusions in the neurorobotics of spatial navigation.

Chapter 2

A Self-Driving Robot Using Deep Convolutional Neural Networks on Neuromorphic Hardware

2.1 Introduction

As the need for faster, more efficient computing continues to grow, the observed rate of improvement of computing speed shows signs of leveling off (Backus, 1978; Danowitz et al., 2012). In response, researchers have been looking for new strategies to increase computing power. Neuromorphic hardware is a promising direction for computing, taking a brain-inspired approach to achieve magnitudes lower power than traditional Von Neumann architectures (Mead, 1990; Indiveri et al., 2011). Mimicking the computational strategy of the brain, the hardware uses event-driven, massively parallel and distributed processing of information. As a result, the hardware has low size, weight, and power, making it ideal for mobile embedded systems. While the traditional solution to performing computation with a

limited power supply is often to offload computation through cloud computing, this is unreliable in areas of limited connectivity and would be ineffective for tasks requiring immediate results. With neuromorphic hardware, computationally intensive algorithms could be run at low power on the device itself.

One such application is autonomous driving (Thrun, 2010). In order for an autonomous mobile platform to perform effectively, it must be able to process large amounts of information simultaneously, extracting salient features from a stream of sensory data and making decisions about which motor actions to take (Levinson et al., 2011). Particularly, the platform must be able to segment visual scenes into objects such as roads and pedestrians (Thrun, 2010). Deep convolutional networks (CNNs) have proven very effective for many of these tasks (LeCun et al., 1989). A CNN is a multi-layer feedforward neural network, most often used to classify input data with spatial information such as images (Hornik et al., 1989). From one layer to the next, a filter of weights is convolved along the spatial dimensions. The weights are trained using the standard backpropagation rule, comparing the desired and actual output of the final layer of the network (Rumelhart et al., 1988). Training is performed iteratively, by running a batch of the training data through a forward pass of the network, calculating the difference between expected and actual output, and incrementally adjusting network weights by gradient descent.

Neural networks have long been applied in autonomous driving (Pomerleau, 1989), with more recent focus on CNNs. For instance, Huval et al. (2015) trained a CNN on a large dataset of highway driving to perform a variety of functions such as object and lane detection. Recently, Bojarski et al. (2016) showed that tasks such as lane detection do not need to be explicitly trained. In their DAVE-2 network, an end-to-end learning scheme was presented in which the network is simply trained to classify images from the car's cameras into steering commands learned from real human driving data. Intermediate tasks such as lane detection were automatically learned within the intermediate layers, saving the work of selecting these

tasks by hand.

Such networks are suitable for running on neuromorphic hardware due to the large amount of parallel processing involved. In fact, many computer vision tasks have already been successfully transferred to the neuromorphic domain, such as handwritten digit recognition (Lee et al., 2016), as well as scene segmentation (Cao et al., 2015). However, less work has been done in embedding the neuromorphic hardware on mobile platforms. An example includes NENGO simulations embedded on SpiNNaker boards controlling mobile robots (Conradt et al., 2015; Galluppi et al., 2014). Addressing the challenges of physically connecting these components, as well as creating a data pipeline for communication between the platforms is an open issue, but worth pursuing given the small size, weight and power of neuromorphic hardware.

At the Telluride Neuromorphic Cognition Workshop 2016, we embedded the the IBM NS1e board containing the IBM Neurosynaptic System (IBM TrueNorth chip) (Merolla et al., 2014) on the Android-Based Robotics platform (Oros and Krichmar, 2013) to create a self-driving robot that uses a deep CNN to travel autonomously along an outdoor mountain path. The result of our experiment is a robot that is able to use video frame data to steer along a road in real time with low-powered processing.

2.2 Platforms

2.2.1 IBM TrueNorth

The IBM TrueNorth (Figure 2.1) is a neuromorphic chip with a multicore array of programmable neurons. Within each core, there are 256 input axon lines connected to 256 neurons through a 256x256 synaptic crossbar array. Each neuron on a core is configurably

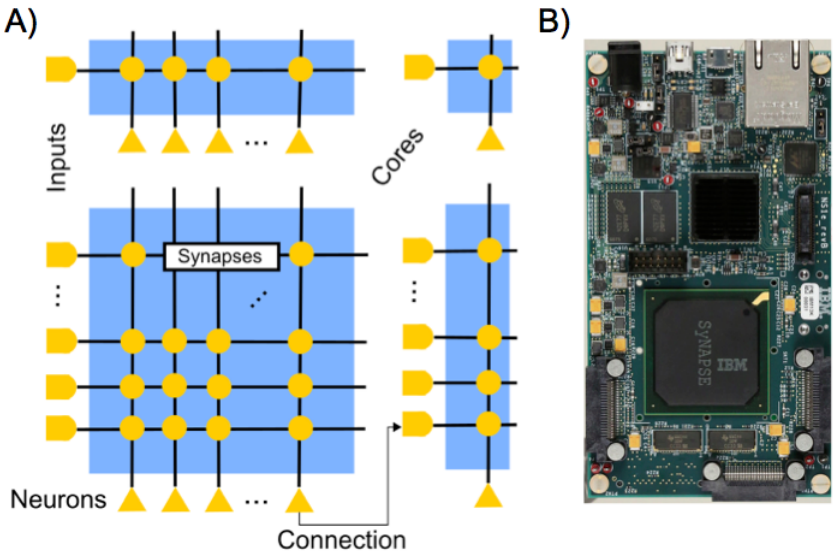

Figure 2.1: A) Core connectivity on the TrueNorth. Each neuron on a core can be configured to connect to all, none, or an arbitrary set of input axons on the core. Neuron outputs connect to input axons on any other core in the system (including the same core) through a Network-on-Chip. B) The IBM NS1e board. Adapted from (Esser et al., 2016).

connected with every other neuron on the same core through the crossbar, and can communicate with neurons on other cores through their input axon lines. In our experiment, we used the IBM NS1e board, which contains 4096 cores, 1 million neurons, and 256 million synapses. The integrate-and-fire neuron model has 23 parameters and may be configured to use trinary synaptic weights of -1, 0, and 1. As the TrueNorth has been used to run many types of deep convolutional networks, and is able to be powered by an external battery, it served as ideal hardware for this task (Akopyan et al., 2015; Esser et al., 2016).

2.2.2 Android Based Robotics

The Android-Based Robotics platform (Figure 2.2) was created at the University of California, Irvine, using entirely off-the-shelf commodity parts and controlled by an Android phone (Oros and Krichmar, 2013). The robot used in the present experiment, the CARLorado, was constructed from a Dagu Wild-Thumper All-Terrain chassis that could easily travel through difficult outdoor terrain. A IOIO-OTG microcontroller (SparkFun Electronics) communi-

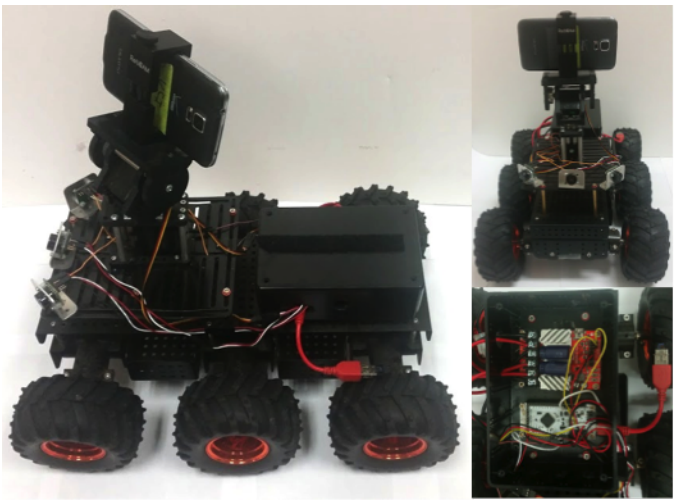

Figure 2.2: Left: Side view of CARLorado. A pan and tilt unit supports the Samsung Galaxy S5 smartphone, which is mounted on a Dagu Wild Thumper chassis. A plastic enclosure holds the IOIO-OTG microcontroller and RoboClaw motor controller. A velcro strip on top of the housing can attach any other small components. Top Right: Front view of CARLorado. Three front-facing sonars can detect obstacles. Bottom Right: Close-up of IOIO-OTG and motor controller.

cated through a Bluetooth connection with the Android phone (Samsung Galaxy S5). The phone provided extra sensors such as a built-in accelerometer, gyroscope, compass, and global positioning system (GPS). The IOIO-OTG controlled a pan and tilt unit for changing the camera view and a motor controller for the robot wheels, using pulse width modulation to change the intensity and direction of movement. The IOIO-OTG also communicated with ultrasonic sensors for detecting obstacles. A differential steering technique was used, moving the left and right sides of the robot at different speeds for turning. The modularity of the platform made it easy to add extra units such as the IBM TrueNorth.

Software for controlling the robot was written in Java using Android Studio. With various support libraries for the IOIO-OTG, open-source libraries for computer vision such as OpenCV, and sample Android-Based Robotics code, it was straightfoward to develop intelligent controls.

2.3 Methods and Results

2.3.1 Data Collection

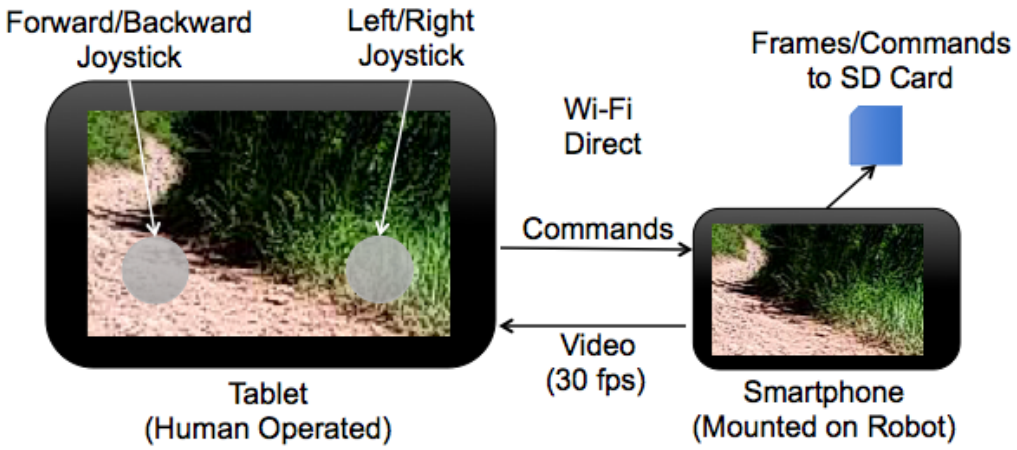

Figure 2.3: Data collection setup. Video from the smartphone mounted on the robot was sent to the tablet through a Wi-Fi direct connection. A human operator used two joysticks on the touchscreen of the tablet to issue motor commands, which were sent to the phone through the same connection. With the joysticks, the operator was able to change the speed of moving and turning by changing the pulse width modulation signal sent to the motor controller. Video frames and their corresponding motor commands in the form of pulse width values were saved to the SD card on the smartphone.

First, we created datasets of first-person video footage of the robot and motor commands issued to the robot as it was manually driven along a mountain trail in Telluride, Colorado (Figures 2.5 and 2.9 top). This was done by creating an app in Android Studio that was run on both a Samsung Galaxy S5 smartphone and a Samsung Nexus 7 tablet (Figure 2.3). The smartphone was mounted on the pan and tilt unit of the robot with the camera facing ahead. JPEG images captured by the camera of the smartphone were saved to an SD card at 30 frames per second. The JPEGs had a resolution of 176 by 144 pixels. Through a Wi-Fi direct connection, the video frame data was streamed from the phone to a handheld tablet that controlled the robot. The tablet displayed a control for moving the robot forward and backward at an adjustable speed and a control for steering the robot left and right at an adjustable speed. These commands from the tablet were streamed continuously in the form

of pulse width values to the smartphone via the Wi-Fi direct connection. The smartphone then relayed these values to the IOIO-OTG, which generated the pulse width modulation signals for controlling the steering power of the robot. These values were also saved on the smartphone as a text file for training purposes. A total of 4 datasets was recorded on the same mountain trail, with each dataset recording a round trip of .5 km up and down a single trail segment. To account for different lighting conditions, we spread the recordings across two separate days, and on each day we performed one recording in the morning and one in the afternoon. In total we collected approximately 30 minutes of driving data. By matching the time stamps of motor commands to video images, we were able to determine which commands corresponded to which images. Images that were not associated with a left, right, or forward movement such as stopping were excluded. Due to lack of time, only the first day of data collection was used in actual training.

2.3.2 Eedn Framework

We used the dataset to train a deep convolutional neural network using an energy-efficient deep neuromorphic network (Eedn), a network that is structured to run efficiently on the TrueNorth (Esser et al., 2016). As the TrueNorth currently does not support on-chip training, the framework provides a method for training network weights off-line and assigning them to the chip. In summary, a CNN is transferred to the neuromorphic domain by first structuring the network according to the core, axon, and neuron structure of TrueNorth. By dividing each layer into groups along the feature dimension (Figure 2.4), the convolutional operation may be distributed among cores of the TrueNorth to take advantage of the parallel processing. When a neuron targets multiple core inputs, exact duplicates of the neuron and synaptic weights are created, either on the same core or a different core. The response of each neuron is the binary thresholded sum of synaptic input, in which the trinary weight values are determined by different combinations of two input lines. Then the weights of

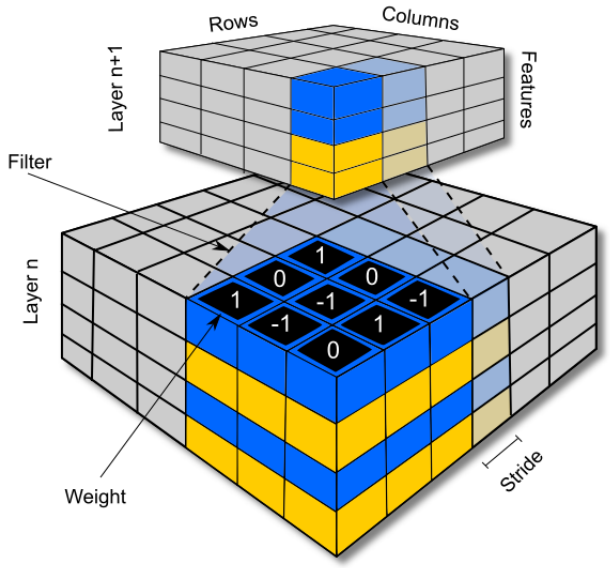

Figure 2.4: Convolution of layers in a CNN on TrueNorth. Neurons in each layer are arranged in three dimensions, which can be convolved using a filter of weights. Convolution occurs across all three dimensions. The third dimension represents different features. The convolution can be divided along the feature dimension into groups (indicated by blue and yellow colors) that can be computed separately on different cores. Adapted from (Esser et al., 2016).

the restructured CNN are learned using a backpropagation algorithm that trinarizes the network weights, consistent with the low dimensional representation of synaptic weights in TrueNorth. Information from one forward pass of the externally trained CNN is represented in Eedn according to the binary spike patterns of each neuron's response to its input, at every timestep. A more complete explanation of the Eedn flow and structure of the convolutional network used (1 chip version) can be found in (Esser et al., 2016).

The video frames were preprocessed by down-sampling them to a resolution of 44 by 36 pixels and separating them into red, green, and blue channels. The output is a single layer of three neuron populations, corresponding to three steering movements of "left," "straight," or "right," as seen in Figure 2.5. Since the steering commands in the training data allowed for different intensities of steering, the commands were simplified such that any amount of turning left was classified as "left" and any amount of turning right was classified as "right."

Figure 2.5: The CNN classified images into three classes of motor output: turning left, moving forward, and turning right. Accuracy of training was above 90 percent.

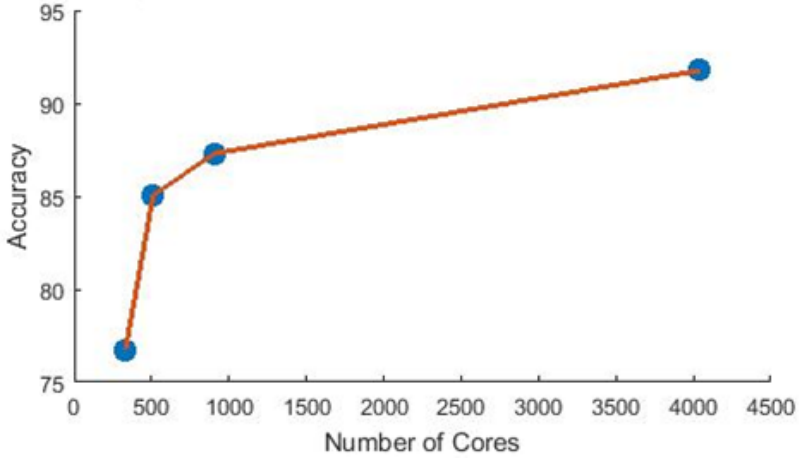

Figure 2.6: Effect of the number of cores used on the accuracy of the CNN. One NS1e board contains 4096 cores. An accuracy above 85 percent was still maintained even with the network reduced to one quarter of a full chip.

Using the Eedn MatConvNet package, a Matlab toolbox for implementing convolutional neural networks, the network was trained to classify images using the motor command class labels. Every image in the dataset was labeled with the corresponding human-trained command of steering to the left, right, or straight. In this way, the CNN learned the types of images and image characteristics associated with each command, completely from the dataset provided. No hand labeling of images or a priori human knowledge of scene characteristics was required. To test accuracy, the dataset was split into train and test sets by using every fifth frame as a test frame (in total 20 percent of the dataset). We achieved an accuracy of

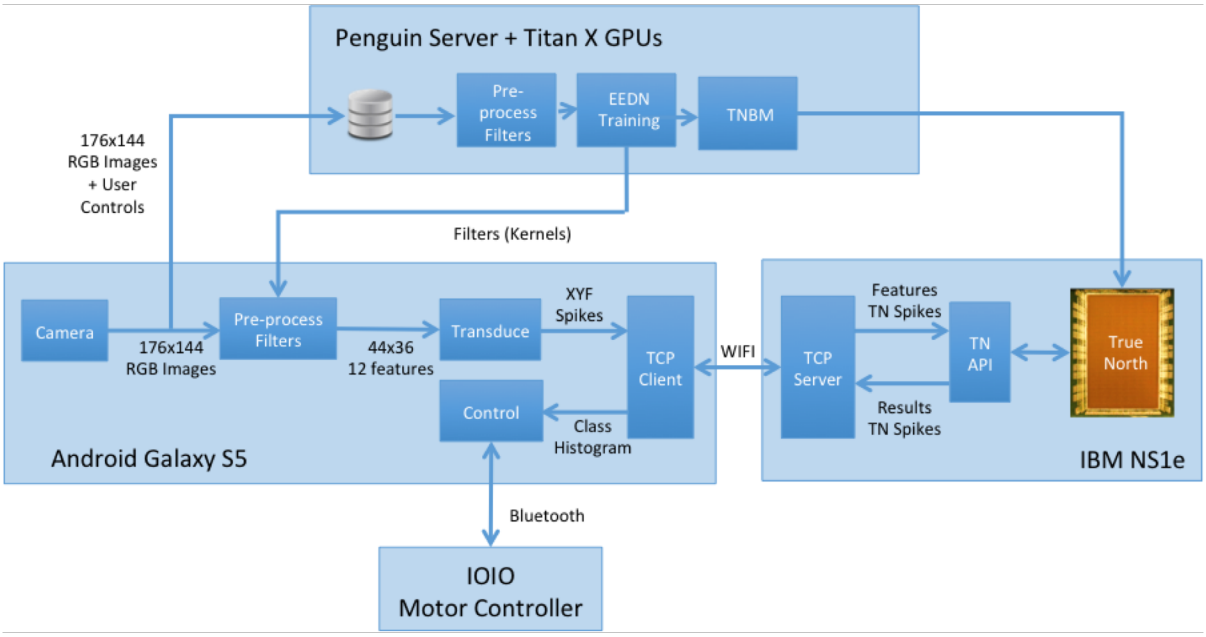

Figure 2.7: Data pipeline for running CNN. Training is done separately using the Eedn MatConvNet package using Titan X GPUs. A Wi-Fi connection between the Android Galaxy S5 and IBM NS1e transmit spiking data back and forth, using the TrueNorth (TN) Runtime API.

over 90 percent, which took 10K iterations and a few hours to train. Training was performed separately from the TrueNorth chip, producing trinary synaptic weights (-1,0,1) that could be used interchangeably in the traditional CNN or Eedn.

For more extensive analysis on the training accuracy achieved by the TrueNorth, multiple CNNs were designed that used different amounts of processor capability. Although one NS1e board contains 4096 cores, not all of them need to be used. Using fewer cores reduces the amount of power consumed by the chip (Esser et al., 2016), at the cost of reduced accuracy. To explore this tradeoff, we trained a range of CNNs that filled from one tenth of a chip to the full chip. The sizes of the CNNs were changed by controlling the stride, filter size, or number of layers (Zeiler and Fergus, 2014). Using only one quarter of the chip resulted in 85 percent accuracy, and using the complete chip increased the accuracy to over 90 percent (Figure 2.6).

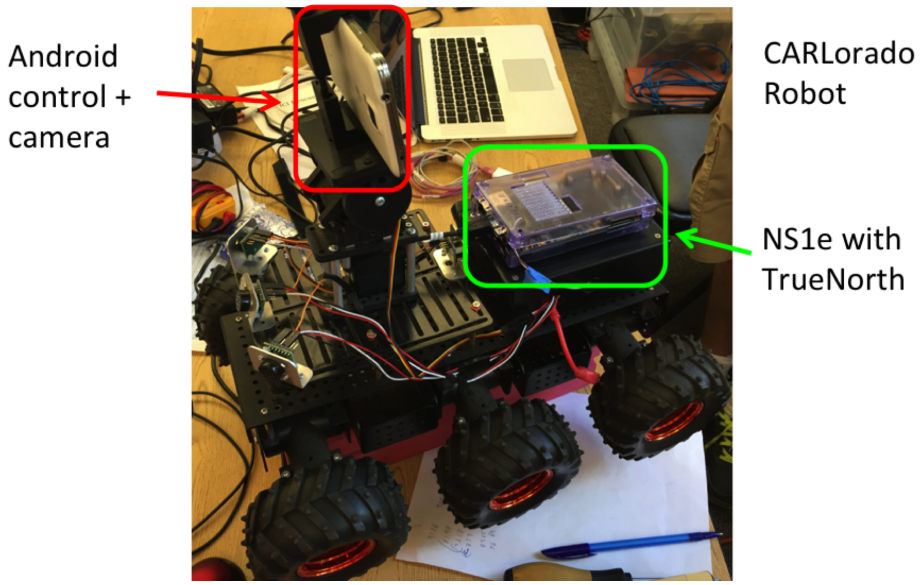

Figure 2.8: Physical connection of TrueNorth NS1e and CARLorado. The NS1e is attached to the top of the housing of the electronics housing using velcro. The NS1e is powered by running connections from the motor controller within the housing. The motor controller itself is powered by a Ni-MH battery attached to the bottom of the robot chassis.

2.3.3 Data Pipeline

With the methods used in (Esser et al., 2016), the weights of the network were transferred to the TrueNorth chip. The CNN was able to run on the chip by feeding input from the camera on the Android Galaxy S5 to the TrueNorth using a TCP/IP connection. In order to achieve this, the phone had to replicate the preprocessing used when training the network. The preprocessing on the phone was achieved by using the Android OpenCV scaling function to downsample the images. Then, the images were separated into red, green, and blue channels. Next, the filter kernels from the first layer of the CNN were pulled from the Eedn training output and applied to the image using a 2D convolution function from the Android OpenCV library. The result of the convolution was thresholded into binary spiking format, such that any neuron with an activity greater than zero was set to spike. The spiking input to the TrueNorth was sent in XYF format, where X and Y are 2D coordinates within a frame and F is a filter. The XYF value describes the identity of a spiking neuron within a layer. At each tick of the TrueNorth, a frame was fed into the input layer by sending the XYF

Figure 2.9: Mountain trail in Telluride, Colorado. Top: Google Satellite image of trail (highlighted) Imagery ©2016 Google. Bottom: Testing CNN performance.

coordinates of all neurons that spiked for that frame. A detailed diagram of the pipeline is found in Figure 2.7. Output from the TrueNorth was sent back to the smartphone through the TCP/IP connection in the form of a class histogram, which indicated the firing activity of the output neurons. The smartphone could then calculate which output neuron was the most active and issue the corresponding motor command to the robot.

2.3.4 Physical Connection of Platforms

The TrueNorth was powered by connecting the robot's battery terminals from the motor controller to a two-pin battery connection on the NS1e board. It was then secured with velcro to the top of the housing for the IOIO and motor controller. A picture of the setup is seen in Figure 2.8. The robot, microcontroller, motor controller, servos, and NS1e were powered by a single Duratrax NiMH Onyx 7.2V 5000mAh battery.

2.3.5 Testing

With this wireless, battery-powered setup, the trained CNN was able to successfully drive the robot on the mountain trail (Figure 2.9). A wireless hotspot was necessary to create a TCP connection between the NS1e and the Android phone. We placed the robot on the same section of the trail used for training. For testing, the robot was programmed to drive forward constantly while steering left, right, or straight according to the class histograms received from the TrueNorth output, which provided total firing counts for each of the three output neuron populations. Steering was done by using the histogram to determine which output population fired the most, and steering in that direction. As a result, the robot stayed near the center of the trail, steering away from green brush on both sides of the trail. At some points, the robot did travel off the trail and needed to be manually redirected back towards the center of the trail. The robot drove approximately 0.5 km uphill, and returned 0.5 km downhill with minimal intervention. It should be noted that there was a steep dropoff on the south side of the trail. Therefore, extra care was taken to make sure the robot did not tumble down the mountainside. A video of the path following performance can be seen at `https://www.youtube.com/watch?v=CsZah2hydeY`.

2.4 Discussion

To the best of our knowledge, the present setup represents the first time the IBM NS1e has been embedded on a mobile platform under closed loop control. It demonstrated that a low power neuromorphic chip could communicate with a smartphone in an autonomous system. Furthermore, it showed that a CNN using the Eedn framework was sufficient to achieve a self-driving application. Additionally, this complete system ran in real-time and was powered by a single off-the-shelf hobby grade battery, demonstrating the power efficiency of the TrueNorth chip. This was possible due to the power savings of running the CNN computations on neuromorphic hardware instead of directly on the smartphone. In comparison with current methods of general-purpose computing on graphics processing unit (GPGPU) approaches to running CNNs which require up to 235 W to operate (Ovtcharov et al., 2015), the Eedn framework is able to train on classic benchmark datasets at 1,200 and 2,600 frames/s and uses between 25 and 275 mW (Esser et al., 2016). Similar power savings should also hold for our application, as our processing rate of 30 frames/s was sufficient for the road following task.

An expansion of this work would require better quantification of the robot's performance. This could be achieved by tracking the number of times the robot had to be manually redirected, or comparing the CNN classifier accuracy on the training set of images versus the classifier accuracy on the actual images captured in real time. Increasing the amount of training data would likely increase the classifier accuracy, since only 15 minutes of data were used for the training as compared to other self-driving CNNs, which have used several days or even weeks of training (Huval et al., 2015; Bojarski et al., 2016). Our success was due in part to the simplicity of the landscape, with an obvious red hue to the dirt road and bold green hue for the bordering areas. It would therefore be useful to test the network in more complex settings.

With further development, path following on Eedn can integrate with a larger system of autonomous driving on a neuromorphic system. For instance, a variety of other sensors such as sonars may be used to train the CNN for better obstacle detection. Additionally, the system can be combined with longer-term strategies, such as path planning and decision-making.

Chapter 3

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

3.1 Introduction

The previous chapter discussed the use of neuromorphic computing for reactionary functions in autonomous robotics, such as path following and obstacle avoidance. However, a complete autonomous navigation system also requires a system for map representation and path planning. Path planning involves calculating an efficient route from a starting location to a goal, while avoiding obstacles and other impediments. Despite much advancement over several decades of robotic research, there are still many open issues for path planners (LaValle, 2011a,b). Classic path planning algorithms include Dijkstras algorithm, A*, and D*. Dijkstras algorithm uses a cost function from the starting point to the desired goal. A* additionally considers the distance from the start to the goal as the crow flies (Hart et al.,

1968). D* extends the A* algorithm by starting from the goal and working towards the start positions. It has the ability to re-adjust costs, allowing it to replan paths in the face of obstacles (Stentz and Carnegle, 1993). These algorithms can be computationally expensive when the search space is large.

Rapidly-Exploring Random Trees (RRT) are a less expensive approach because they can quickly explore a search space with an iterative function (LaValle and Kuffner, 2001). Still these path planners are computationally expensive and may not be appropriate for autonomous robots and other small, mobile, embedded systems. Because of their event driven design and parallel architecture, neuromorphic hardware holds the promise of decreasing size, lowering weight and reducing power consumption. These systems are modeled after the brains architecture and typically use spiking neural elements for computation (Krichmar et al., 2015). Spiking neurons are event driven and typically use an Address Event Representation (AER), which holds the neuron ID and the spike time, for communicating between neurons. Since spiking neurons do not fire often and post-synaptic neurons do not need to calculate information between receiving spikes, neuromorphic architectures allow for efficient computation and communication.

Path planning approaches that may be a good fit for neuromorphic applications are wavefront planners (Barraquand et al., 1992; Soulignac, 2011) and diffusion algorithms (Bellman, 1958). In a standard wavefront planner, the algorithm starts by assigning a small number value to the goal location. In the next step, the adjacent vertices (in a topological map) or the adjacent cells (in a grid map) are assigned the goal value plus one. The wave propagates by incrementing the values of subsequent adjacent map locations until the starting point is reached. Typically, the wave cannot propagate through obstacles. A near-optimal path, in terms of distance and cost of traversal, can be read out by following the lowest values from the starting location to the goal location.

We recently introduced a spiking neuron wavefront algorithm for path planning that adapts

to changes in the environment (Krichmar, 2016). The adaptive element is inspired by empirical findings supporting experience dependent plasticity of axonal conduction velocities. Unlike prior implementations of spiking wavefront path planners, our algorithm introduces an adjustable spike delay that could potentially allow for dynamic online adaptation to realistic environmental costs, while maintaining a temporally sparse coding of the path. This is also similar navigation in biology, in which a map is acquired and used to make intelligent decisions on where to go and what to do (O'Keefe and Nadel, 1978; Gallistel, 1993). The idea of a cognitive map, which was proposed by E.C. Tolman in the last century (Tolman, 1948), is where the animal takes costs, context, and its needs into consideration when moving through its environment.

In simulations, we were able to show that this algorithm was more computationally efficient and sensitive to cost than existing path planning algorithms. Versions of the wavefront algorithm have been implemented on neuromorphic hardware that supports spiking neurons (Koul and Horiuchi, 2015; Koziol et al., 2013, 2014), including our own algorithm which was successfully implemented on the IBM TrueNorth chip (Fischl et al., 2017). In addition, it has been shown to plan efficient paths on mobile robots and robotic arms (Soulignac, 2011; Koul and Horiuchi, 2015; Koziol et al., 2012). However, these algorithms are not flexible in dynamic environments or in situations where the context changes. Other neurally-inspired implementations of the wavefront or diffusion idea use learning rules to learn routes through environments (Gaussier et al., 2002; Ponulak and Hopfield, 2013; Quoy et al., 2002; Samsonovich and Ascoli, 2005). These models rely on synaptic plasticity to learn and adapt to environments. However, most of the environments used in these experiments have been static and highly constrained.

In the present chapter, we expand upon this previous work, demonstrating the algorithms effectiveness in complex natural environments. Rather than using a manually constructed map, we create maps from an outdoor park with an abundance of natural obstacles and varied

terrain. These obstacles and varied terrain are reflected in the algorithms cost function. We further show how this algorithm can be implemented on an autonomous robot navigating an outdoor environment. The mobile robot had to consider real-world costs and tradeoffs in the environment, such as smooth roads versus rough grass, as well as obstacles such as benches, bushes, and trees. The robot demonstrated context-dependent path planning through this environment and the spiking wavefront was efficient enough to run on an Android smartphone that was mounted on and controlled the robot.

3.2 Methods

3.2.1 Neuron Model and Connectivity

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

$$v_i(t+1) = u_i(t) + I_i(t) \tag{3.1}$$

$$u_i(t+1) = min(u_i(t) + 1, 0) \tag{3.2}$$

$$I_i(t+1) = \sum_j \begin{cases} 1, & \text{if } d_{ij} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

$$d_{ij}(t+1) = max(d_{ij}(t) - 1, 0) \tag{3.4}$$

$d_{ij}(t)$ is the axonal delay between when neuron $v_j(t)$ fires an action potential and neuron $v_i(t)$ receives the action potential. When $v_j(t)$ fires an action potential, $d_{ij}(t)$ is set to a delay value of $D_{ij}(t)$, which was assigned according to variable costs in the environment. Note from Equation 3.4 that d_{ij} has a null value of zero unless the pre-synaptic neuron fires an action potential. Equations 3.1-3.4 calculate the membrane potential, recovery variable, synaptic input, and axonal delay for neuron i at time step t, which is connected to j pre-synaptic neurons. The neuron spiked when v in Equation 3.1 was greater than zero, in which case, v was set to 1 to simulate a spike, u was set to minus 5 to simulate a refractory period, and the axonal delay buffer, d, was set to D. The recovery variable, u, changed each time step per Equation 3.2. The delay buffer, d, changed each time step per Equation 3.4. I in Equation 3.3 was the summation of the j pre-synaptic neurons that delivered a spike to post-synaptic neuron i at time t. Because of the refractory and delay periods, most neurons will be inactive during each timestep, resulting in sparse activity. Although neurons have to check if they fired a spike or received a spike each timestep, most of the computation occurs when neurons emit or receive a spike. The neural network consisted of a 20x20 grid of spiking neurons as described in Equations 3.1-3.4. The 20x20 grid corresponded to grid locations used in the outdoor robot experiments. Each neuron corresponded to a location in the environment and was connected to its eight neighbors (i.e., N, NE, E, SE, S, SW, W, NW). At initialization ($t = 0$), v and u were set to 0. All delays, D, were initially set to 5,

but could vary depending on experience in the environment. D represented the time it takes to propagate a pre-synaptic spike to its post-synaptic target.

3.2.2 Axonal Delays and Plasticity

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

$$D_{ij}(t+1) = D_{ij}(t) + \delta(map_{xy} - D_{ij}(t)) \tag{3.5}$$

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

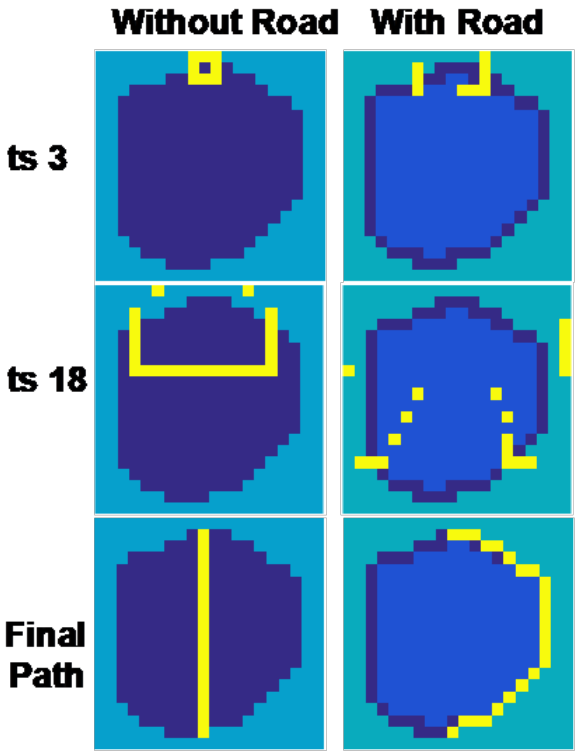

Figure 3.1: Spike wave propagation in simulation. The top panels show the network activity at timestep 3, the middle panels show network activity at timestep 18, and the bottom panels show the resulting path. The left side shows the test area without a road. The light blue is the surrounding region, which has a cost of 9. The dark blue depicts the location of an open grass field, which has a cost of 3. The right side shows the test area that takes into consideration a road, which has a cost of 1 and is shown in dark blue. The spike wave propagation is shown in yellow. The starting location is at the top middle of the map, and the goal location is at the bottom middle of the map. Note how the spike wave propagates faster when there is a low cost road present.

3.2.3 Spike Wave Propagation and Path Readout

Figure 3.1 shows the progression of a spike wave in a typical environment. The example shows how the algorithm is sensitive to different costs; the left columns of Figure 3.1 show an environment where the best path is the most direct route, and the right columns show an environment where it is advantageous to take a longer route along a smooth road. The neuron at the starting location is triggered to emit a spike. The top panels of Figure 3.1 show the start of the spike wave emanating from the start position. Note how the spike wave is propagating faster on the regions that depict a smooth road (Figure 3.1, right column).

This is because the road has a traversal cost of 1, whereas the grass field has a cost of 3. Each spike, which is shown in yellow in Figure 3.1, is recorded in the AER table, with its neuron ID and time step.

To find the best path between the start and goal locations, we used the list of spikes held in the AER table. From the goal, the list was searched for the most recent spike from a neuron whose location was adjacent to the goal location. If more than one spike met this criterion, the neuron whose location corresponded to the lowest cost and was closest to the start location was chosen. This iteratively proceeded from the goal through other neuron locations until a spike at the start location was found. The bottom right image in Figure 3.1 shows the found path.

In complex environments there was the potential for multiple waves to occur and collide (Figure 3.2). In this case, the AER table could contain more than one path. To find the best path, a second pass was made through algorithm with a temporary map that had a cost of 1 for the paths from the first pass, with the rest of the map set to 20. This second pass of the spike wave algorithm ensured that the resulting path was most efficient in terms of length and cost.

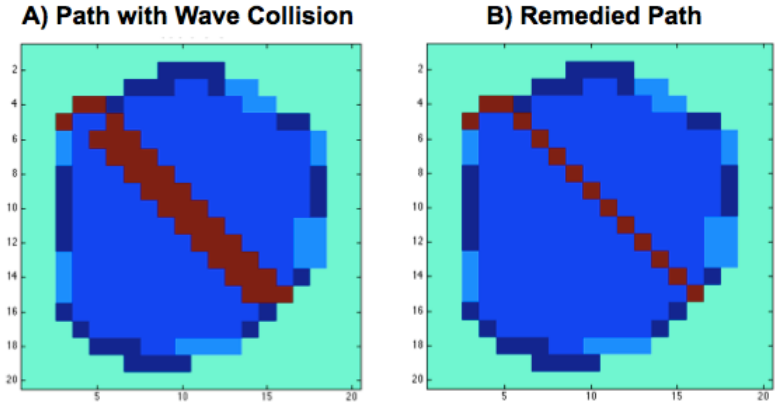

Figure 3.2: In some instances, the collision of multiple spike waves generated inefficient paths (left). This was remedied by adding a second pass through the algorithm with a cost map containing just the paths from the first pass (right).

3.2.4 A* Path Planner

For comparison purposes, we implemented the A Star (A*) algorithm (Hart et al., 1968), which is commonly used in path planning. A* uses a best-first search and attempts to find a least-cost path from the start location to the goal location. The cost includes the Euclidean distance from the start, the Euclidean distance from the goal, and the cost of traversing the location. From the start location, adjacent locations are placed in a node list. Then the node list is searched for the node with the lowest cost. The location corresponding to this low cost node is expanded by placing adjacent, unevaluated locations on the node list. The process is repeated until the goal location is reached. The A* algorithm can find the shortest path based on its cost function.

3.2.5 Map of Environment

To demonstrate the effectiveness of our spiking wavefront planner, we tested the algorithm in a real environment through a variety of terrains in Aldrich Park, a 19-acre botanical garden at the University of California, Irvine. Two sections of the park, an open area and a cluttered area (Figure 3.3) were transformed into 20x20 grid maps encoding the costs of traveling and the GPS coordinates. We generated the GPS coordinates by pacing off the area with a smartphone (Samsung Galaxy S5) and recording the GPS points with an Android application.

Two maps created from the sections of Aldrich Park consisted of a 20x20 grid of GPS coordinates and terrain costs (see Figure 3.3). The first, referred to as Map 1, was in an open grassy area of the park, which was surrounded by a paved sidewalk. In one variant, referred to as Without Road (Figure 3.4A), the grassy area had a cost of 3, and all other areas had a cost of 9. In the With Road variant (Figure 3.4B), the paved sidewalk around the grassy area was given a cost of 1. In the With Road and Obstacles variant (Figure

3.4C), benches, bushes, and trees were given a cost of 6. The second map, referred to as Map 2 (Figure 3.4D), had an outer region with a cost of 6, large trees and brush had a cost of 10, the paved road had a cost of 1, and the gravel road had a cost of 2. Map 2 was stretched horizontally such that the asphalt path location roughly matched the asphalt path of Map 1, allowing for better route comparisons between maps. These maps were used in both simulations and in autonomous robot experiments.

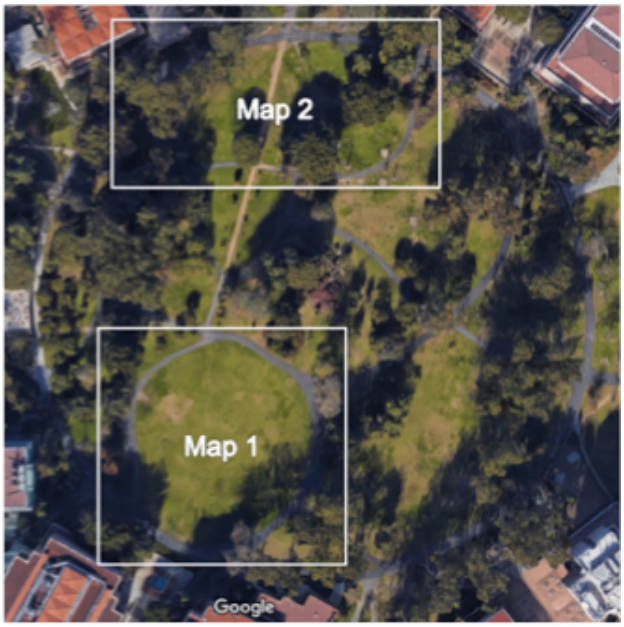

Figure 3.3: Google satellite image of Aldrich Park at the University of California, Irvine. Two sections of the park (boxed) were transformed into cost maps (Map 1 as bottom box and Map 2 as top box) for the spiking wave planner. Imagery ©2016 Google.

3.2.6 Robot Hardware and Design

For the robot experiments, we used the same Android-Based Robotic Platform as described in the previous chapter. However, it did not include the IBM True North, instead relying on the Android smartphone for most of the computation.

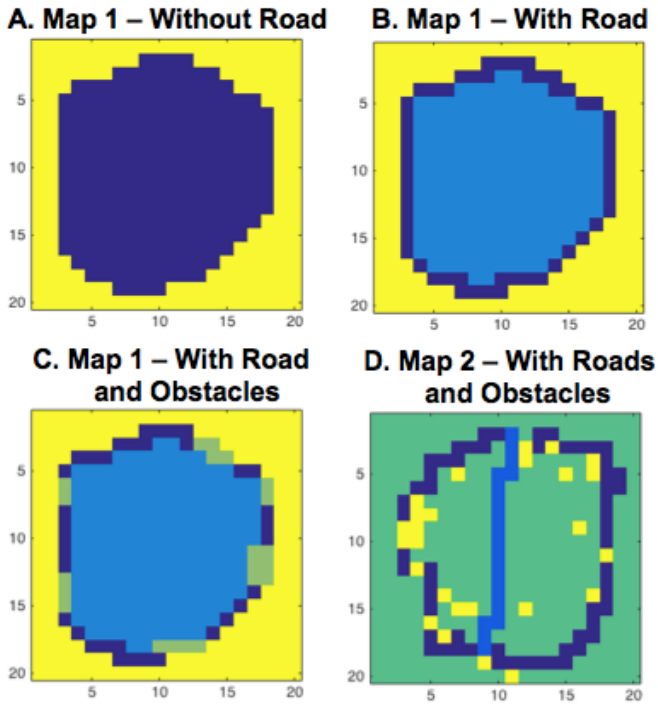

Figure 3.4: 20 x 20 cost grids created from two areas of Aldrich Park. A.) An open area with uniform low cost in the traversable area, and high cost outside of this area. B.) The same area as A but with a lower cost for the surrounding road. C.) The same area as B but with obstacles denoting benches, bushes and trees near the road. D.) A second area in Aldrich Park with high cost for trees, low cost for asphalt roads, and medium cost for dirt roads.

3.2.7 Computation

Simulations and robot experiments were run to test the spike wave algorithm. The simulations of the spike wave and the A* algorithm were run in MATLAB. For robot experiments, the spike wavefront algorithm, robot I/O, and robot control software were implemented in Java using Android Studio, and run as an app on a Samsung Galaxy S5. Figure 3.5 shows a screenshot of the Android application. A graphical user interface (GUI) on the phone allowed the user to input start and goal grid locations, as well as select a map. The app then generated a path using the spike wavefront planner. The phone then displayed the desired path on the GUI (see Figure 3.5). Once the operator pressed the Auto button on the GUI, the app generated a list of ordered GPS waypoints, from the start to the goal location, from the path grid locations. The robot then used a navigation strategy to visit each waypoint on

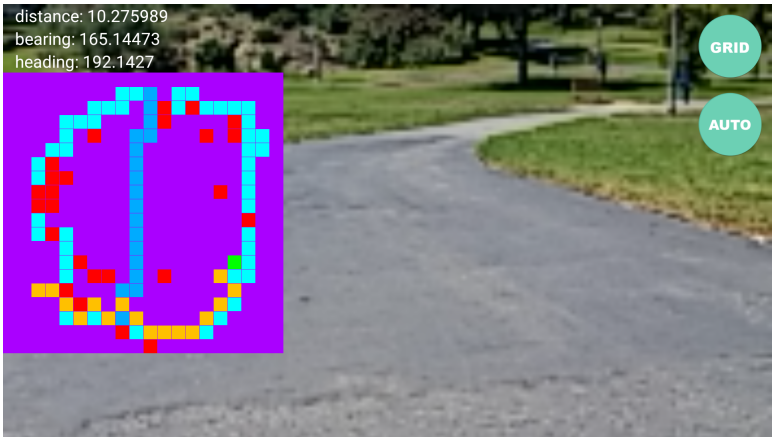

Figure 3.5: Screenshot of app used for robot navigation. The screen displays a camera view overlaid with information about distance to the destination, bearing to the destination ranging 0-360 degrees from true north, heading direction (also ranging 0-360 degrees from true north), and the 2D cost grid. Colors on the grid ranging from dark blue to red indicated the costs of the grid, with tree locations marked at highest cost in red. The planned path of the robot is indicated in yellow and the current location of the robot is marked in green. The Grid button toggles the grid view on and off and the Auto button switches the robot into and out of autonomous navigation mode.

the list in succession. The robot stopped moving once the last waypoint, which represented the goal location, was reached.

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

3.3 Results

3.3.1 Path Planning Simulations

Table 3.1 shows path and cost metrics for simulated path planning that compared the spike wave algorithm with the A* path planner. Simulations were run with all four map variants: 1) Map 1 - Without Road, 2) Map 1 - With Road, 3) Map 1 - With Road and Obstacles, and 4) Map 2 - With Road and Obstacles. 100 start and goal locations were randomly chosen, in which the locations could not be out of bounds and the Euclidean distance between the start and goal was greater than 5 grid units.

The path lengths between the two algorithms were nearly identical (see Table 3.1), but the spike wave algorithm found lower cost paths, especially when there were obstacles and roads present ($p < 0.001$; Wilcoxon Ranksum). This is because the spike wave algorithm depends primarily on cost, whereas our A* implementation uses the common and standard heuristic of Euclidian distance in addition to the cost of a node on the map. Although A* is proven to be optimal given an admissible heuristic (Hart et al., 1968), our heuristic is only admissible when calculating for shortest path. A varied and dynamically changing environment would make a cost-admissible heuristic more difficult to determine, whereas the spike wave algorithm inherently includes both distance and cost in its calculation by combining neighbor connectivity and axonal delay.

The A* algorithm ran faster than the spike wave algorithm when calculating the paths, as measured by the tic/toc functions in MATLAB (see Table 3.1). Interestingly, this difference became smaller as the maps became more complex (see Map 2 in Table 3.1). This is due to the presence of low cost roads among high cost obstacles, which leads to the algorithm requiring less neural activity to calculate a path. We later discuss how the spike wave algorithm can be made parallel, asynchronous, and implemented on neuromorphic hardware. This should

Table 3.1: Comparison between spike wavefront planner and A* planner in different environments

	Path Length		***Path Cost***		***Time(ms)***	
	Spike Wave	A*	Spike Wave	A*	Spike Wave	A*
Map 1 - Without Road	8.5	8.5	25.5	25.5	6.11	1.08
Map 2 - With Road	10	9	21	24	3.81	.76
Map 1 - Road and Obstacles	9	9	24	24.5	4.45	0.88
Map 2 - Road and Obstacles	13	11	34.5	42.5	5.61	2.71

Table 3.2: Fréchet distances (in meters) between planned route and actual route for spike wavefront planner

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

allow for substantial speedups in processing, and reduction in power consumption, which can be quantifiably measured against the baseline run times reported here.

3.3.2 Robotic Experiments

Given that the spike wavefront planner showed possible advantages over a traditional approach in simulation, we aimed to test the plausibility of embedding the planning algorithm on an autonomous robot with limited power and computational resources. Robot experiments were conducted in Aldrich Park on the campus of the University of California, Irvine (see Figure 3.3). For each map, we tested a set of six routes with the same start and end coordinates on the cost grids. See Figures 3.6-3.9 for route start and end coordinates. The generated routes were different depending on whether roads and obstacles were taken into account. For each route in a given map, the robot ran four trials, following the route produced by the spiking wavefront path planner. To account for changing satellite conditions and other environmental factors, we spread out the testing times to sample the variance

of GPS signal quality. The first two runs were performed in the morning and the last two runs were performed in the afternoon. Since the robot only relied on GPS and compass to navigate, it was sometimes necessary to manually redirect the robot slightly away from undetected obstacles. This occurred very infrequently in Map 1, and more frequently in Map 2 due to the presence of dense vegetation and an abundance of obstacles. In Section 3.4, we discuss ways to mitigate these interventions.

Overall, the actual robot trajectories matched the desired trajectories calculated from the spike wavefront algorithm quite well. For each map condition, Figs. 7-10 show the satellite image of the area, the cost map used by the path planner, and the trajectories for the 6 routes from a starting grid location to a goal grid location. The trajectories were superimposed on the street view of Google Maps. The black line in these figures shows the desired path, and the four colored lines show a robot trajectory. Occasionally the GPS signal became unreliable due to buildings, trees, and other environmental noise. This sometimes caused the robot to drive away from the desired destination, requiring the robot to backtrack and visit a missed waypoint.

We used the discrete Fréchet distance (Eiter and Mannila, 1994) as a metric for calculating the similarities of trajectories between the actual robots movements and the intended route. Intuitively, Fréchet distance is the minimal leash length necessary to physically connect two agents as they walk along their two separate paths. The agents are allowed to pause at any time but not permitted to backtrack, and both must complete their respective paths from start to endpoint. Compared to other comparison techniques such as Hausdorff distance, Fréchet distance takes into account the specific ordering of points on the trajectory, ideal for our experimental conditions. Table 3.2 shows mean and standard deviation of Fréchet distances for each of the maps and routes, with the sample size of 4 trials for each condition.

As our navigation strategy defined reaching a waypoint as arriving within 10 meters of

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

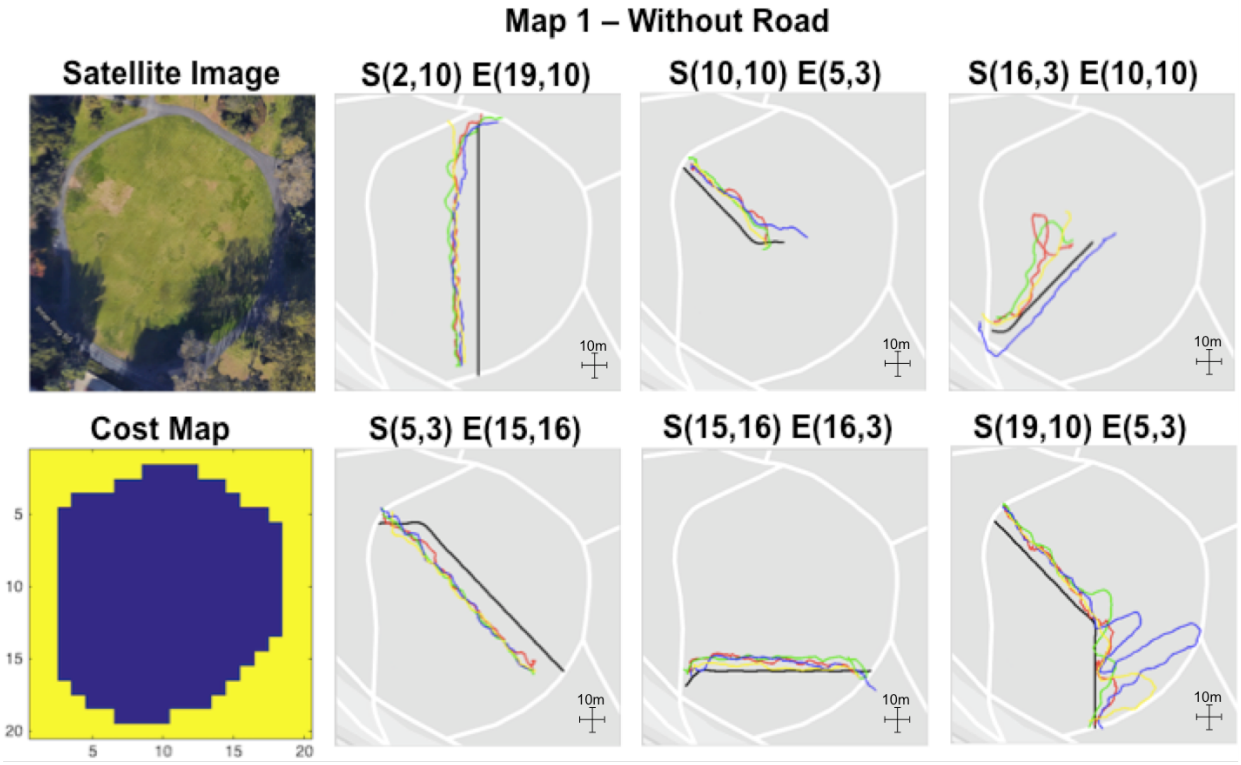

Figure 3.6: Experimental results for the spiking path planning algorithm on Map 1, with no unique costs encoded for the path. Start and end locations are noted by their row and column position on the cost map. Black lines indicate the planned route and the 4 colored lines indicate the actual route taken by the robot. Scale bars indicate the length of 10 meters along latitudinal and longitudinal axes, indicating the size of error threshold of our navigation strategy. Imagery ©2016 Google.

3.4 Discussion

In prior work, we introduced a path planning algorithm that used spiking neurons and axonal delays to compute efficient routes (Krichmar, 2016). The spike wavefront path planner could generate near optimal paths and was comparable to conventional path planning algorithms,

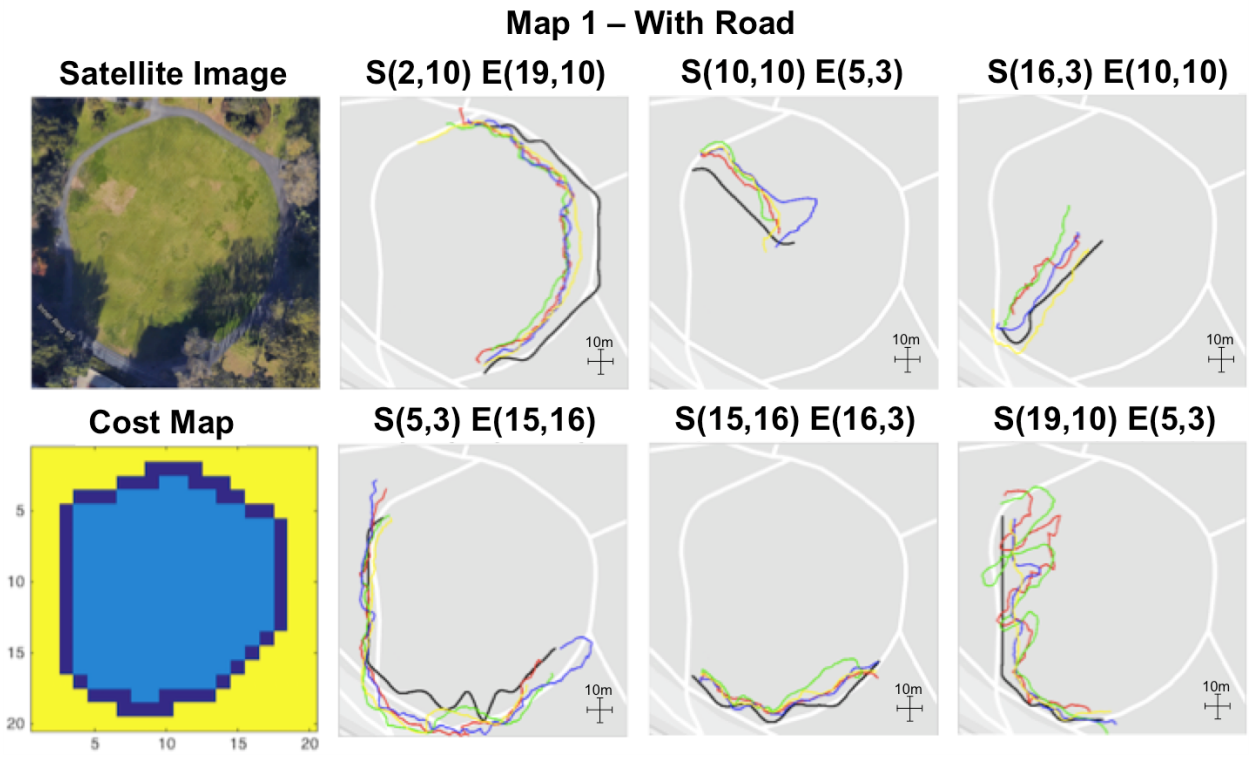

Figure 3.7: Experimental results for the spiking path planning algorithm on Map 1, with lower costs encoded for the path. Black lines indicate the planned route and the 4 colored lines indicate the actual route taken by the robot. Imagery ©2016 Google.

such as the A* algorithm or a standard wavefront planner. We introduced a learning rule that represented the cost of traversal in axonal delays. Because the spike wavefront is a local algorithm (i.e., computations are independent and based on neighboring neurons), it is suitable for parallel implementation on neuromorphic hardware, as was shown recently with both grid-based and topological maps (Fischl et al., 2017).

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

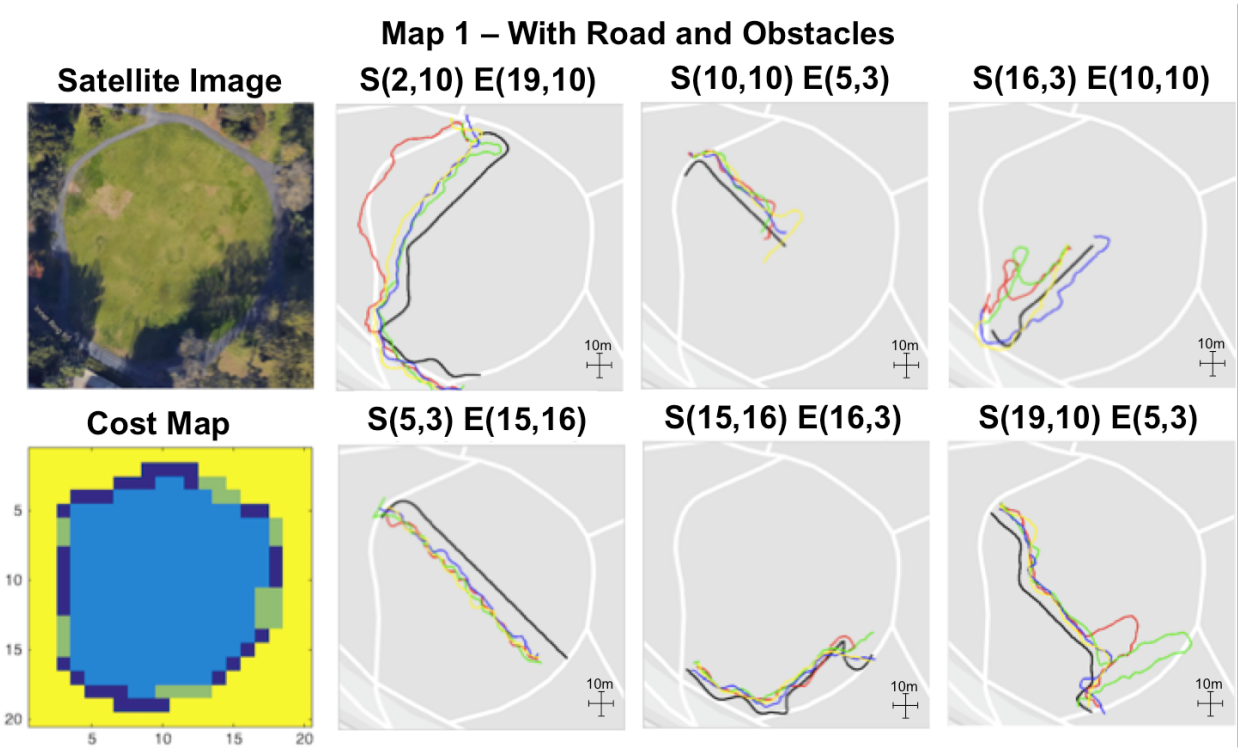

Figure 3.8: Experimental results for the spiking path planning algorithm on Map 1, with lower costs encoded for the path. Black lines indicate the planned route and the 4 colored lines indicate the actual route taken by the robot. Imagery ©2016 Google.

demonstrated that the algorithm was lightweight and could support autonomous navigation in real-time.

3.4.1 Neurobiological Inspiration for the Spike Wavefront Algorithm

The present algorithm was inspired by recent evidence suggesting that the myelin sheath, which wraps around and insulates axons, may undergo a form of activity-dependent plasticity (Fields, 2008, 2015). These studies have shown that the myelin sheath becomes thicker with learning motor skills and cognitive tasks. A thicker myelin sheath implies faster conduction velocities and improved synchrony between neurons.

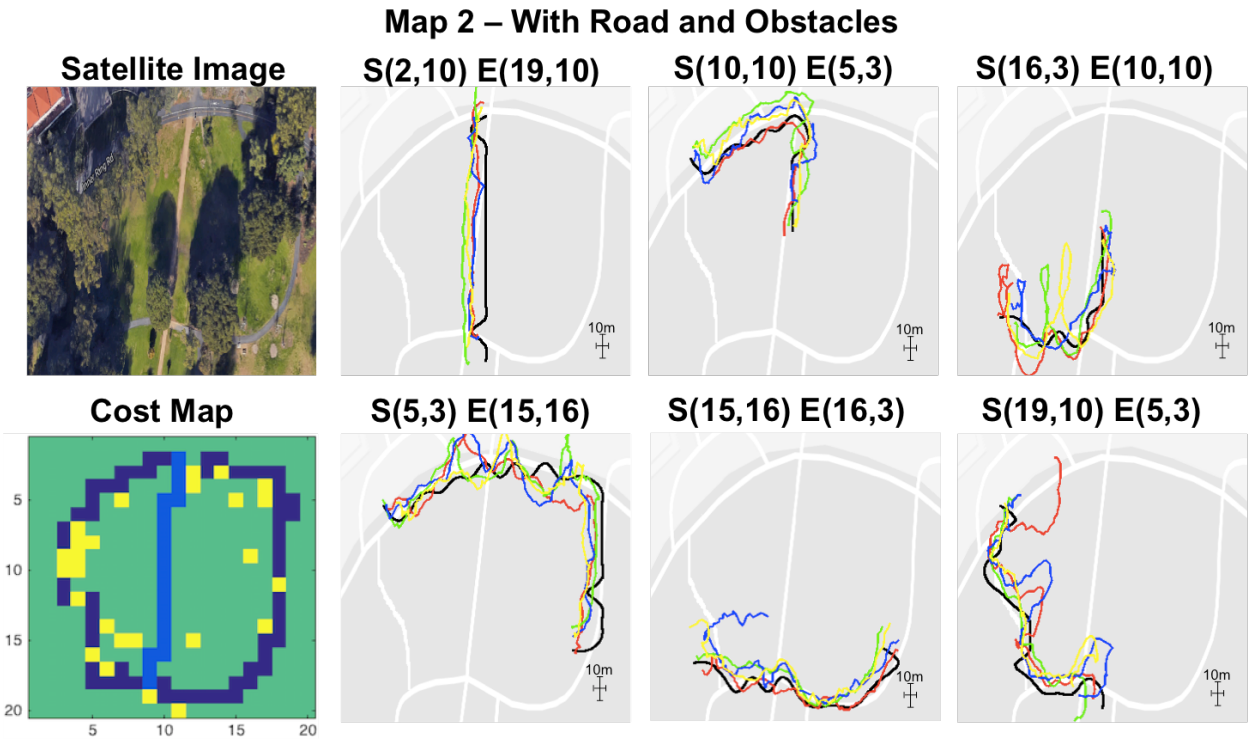

Figure 3.9: Experimental results for the spiking path planning algorithm on Map 1, with lower costs encoded for the path. Black lines indicate the planned route and the 4 colored lines indicate the actual route taken by the robot. Imagery ©2016 Google.

Based on these findings, we developed a learning rule in which a path that traverses through an easy portion of the environment (e.g., via a road) would have shorter axonal delays than a path that travels through rough terrain. Although it is not known if such spatial navigation costs are represented in the brain in this way, and most likely they are not, this learning rule does investigate a rarely considered form of plasticity. Moreover, manipulating the delays, as network can solve a real-world problem using a purely temporal code. Other groups have investigated learning rules based on axonal delays. Wang and colleagues have implemented a Spike Timing Delay Dependent Plasticity (STDDP) rule that can shorten or lengthen the axonal delay between two connected neurons (Wang et al., 2013, 2014). They showed that altering axonal delays had advantages in forming polychronous neuronal groups, which represent spatiotemporal memories (Izhikevich, 2006), over altering synaptic weights via Spike Timing Dependent Plasticity (STDP).

The present algorithm is also inspired by the notion of neuronal waves in the brain. Wave propagation has broad empirical support in motor cortex, sensory cortex, and the hippocampus (Rubino et al., 2006; Benucci et al., 2007; Han et al., 2008; Wu et al., 2008; Lubenov and Siapas, 2009; Sato et al., 2012; Zanos et al., 2015). These waves have been suggested as a means to solve the credit assignment problem for associating a conditioned stimulus with the later arrival of an unconditioned stimulus (Palmer and Gong, 2014). In our own work, we have shown that neuronal wave dynamics in complex spiking neural network models can be used to associate visual stimuli with noisy tactile inputs in a physical robot (Chou et al., 2015). Therefore, the idea of solving problems with spike timing generated by propagating waves of activity has biological and theoretical support.

Relevant to the present task, is the experimental observation of neural activity that represent potential paths through space. Sequences of place cell activity in the hippocampus prior can predict an animals trajectory through the environment (Dragoi and Tonegawa, 2011, 2013; Pfeiffer and Foster, 2013, 2015; Silva et al., 2015). These so-called preplays may be a means to assess different possible paths prior to selecting a specific path plan. In a way, this is similar to how the spike wavefront planner operates. Sequences of place activity are generated, and the spike sequence that arrives first at the goal is the one selected for execution.

3.4.2 Implementation of Spike Wavefront Planner on Neuromorphic Hardware

The present path planner calculates paths based on the timing of spiking neurons. Because each neuron can calculate its state independently, the algorithm could realize impressive speedups through parallelization. Moreover, spiking neuron networks are inherently event-driven, that is, a new state is only calculated when an incoming spike has been received. This further reduces computational load. Lastly, by stopping as soon as the first spike is received

at a goal node, the spike wavefront planner algorithm only calculates what is necessary. For example, when there were variable costs, such as in Map 2, the amount of time to calculate a path with the spike wavefront planner was reduced relative to the A* path planner (see Table 3.1). It should be noted that the A* path planner can be parallelized (Zhou and Zeng, 2015), but unlike this and other conventional algorithms, they cannot take advantage of neuromorphic hardware as can spiking neuron algorithms.

As has been shown in prior implementations, the spike wavefront algorithm is compatible with neuromorphic hardware (Koul and Horiuchi, 2015; Koziol et al., 2012, 2013, 2014; Fischl et al., 2017). These implementations show the feasibility and parallelization of the wavefront planner. Moreover, they show how this neuromorphic algorithm can generate optimal paths. In addition, IBMs TrueNorth neuromorphic chip was recently embedded on the robot used in the present experiments in an autonomous self-driving application (Hwu et al., 2016). Considering that the present spike wavefront algorithm has been implemented on TrueNorth (Fischl et al., 2017), a complete neuromorphic path planning system is now feasible on our robot.

The present paper builds on these implementations by adding a learning rule to make the planner more flexible and to consider the relative costs of traversing an environment. Axonal delays have been introduced in large-scale spiking neural network simulations (Izhikevich and Edelman, 2008; Izhikevich et al., 2004), but are not typical for neuromorphic hardware. However some neuromorphic designs include axonal delays (Wang et al., 2013, 2014; Cruz-Albrecht et al., 2012; Wang et al., 2015). To implement the present algorithm in neuromorphic hardware, all that would be needed is a delay buffer, delay line, or a means to schedule spikes at specific times in the future. Because a synaptic based learning rule, such as STDP, is not needed for the present algorithm, the circuitry to support the spike wavefront planner could be simplified.

In the present algorithm, the AER representation is used to read out the path, which may

be a limitation since it requires saving the AER list for each planned route. It also requires a planning calculation and readout for every route. A more natural implementation might use the rank order of the spike wave in a similar way to that proposed by Thorpe and colleagues (Thorpe et al., 2001; VanRullen et al., 2005). Such alternative readout implementations will be explored in the future.

3.4.3 Comparison to Other Neurally Inspired Path Planning Approaches

The result of our algorithm has complementary parallels to past work in bioinspired algorithms for mobile robot control (Ni et al., 2016). For instance, Ni and Yang (2011) propose a neural network for multirobot cooperative hunting in unknown environments, representing space in a 2D neuron grid and using a shunting model to represent attraction and repulsion agents on the field. Similarly, our algorithm could draw upon these principles, representing not only environmental costs but costs of interacting with other dynamic agents cooperatively and competitively. It also opens the possibility of neuromorphic solutions for the complex tasks of swarm coordination in mobile robots.

Aside from neural navigation models inspired by hippocampal activity and cognitive map representation, cerebellar models of motor control using the delayed eligibility trace learning rule have also been used for spatial motion planning (McKinstry et al., 2006), with further developments increasing its efficacy in real environments such as urban expressways and tracks (Shim et al., 2015). Perhaps a model of predictive motor control combined with a larger cognitive map representation could be implemented in neuromorphic hardware to form an effective multi-scale motion planning system.

3.4.4 Simultaneous Path Planning and Mapping

The present algorithm could be modified to build a map as the robot explores its environment. It would need additional sensors to measure the cost of traversal or some other cost function related to navigation. Rather than assuming that the environment is known and static, the robot could update the map with each path it generates. This would require setting the learning rate in Eqn. 5 to be less than one. In addition, if the spike wavefront planner had a learning rate between 0 and 1, the uncertainty of the cost at a location would be represented. Similar to (Schultz et al., 1997), this would result in the spike wavefront planner predicting the cost of traversing locations in an environment. Moreover, the planner could utilize an exploration/exploitation tradeoff to decide whether to explore unknown regions, or exploit previously navigated regions. Such tradeoffs have been implemented in neurobiologically inspired algorithms (Cox and Krichmar, 2009; Krichmar, 2008; Aston-Jones and Cohen, 2005b). Such a planner could respond flexibly and fluidly to dynamic environments, or the changing needs of the robot. For example, if the robot needed to get to a location as fast as possible, it might take a direct, but more risky route from the start to goal location. However, if the robot wanted to conserve energy, it might take a longer, but easier path. The different trajectories taken by the robot demonstrate this capability. Context is represented in the map itself. In a future implementation, one could change the cost values of the map based on the robots needs, thus changing the robots behavior.

Additionally, building a map incrementally opens up many possibilities of representing the map besides a 2D grid configuration. For example, a more flexible arrangement such as a topological map is compatible with our spike wave propagation algorithm, and in fact has recently been achieved with large-scale maps (Fischl et al., 2017). For increased resolution of map representation, a system of multi-scale place recognition may also be considered (Chen et al., 2014). Further, a hierarchical spiking neural network (Beyeler et al., 2013) could be involved in forming multi-scale representation compatible with neuromorphic hardware. Any

of these suggested implementations would ease the computational load of 2D grid implementations of the A* and the spike wavefront propagation algorithm, both of which increase in complexity with the grid resolution.

3.5 Conclusion

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

Chapter 4

A Neurobiological Model of Schemas and Memory Consolidation

4.1 Introduction

On top of short-term reactive motion planning and long-term path planning, navigation is also part of more abstract cognitive processes such as memory and decision making. In this chapter, we turn our attention to a model of memory consolidation, explaining the cognitive mechanisms that ultimately control the navigation goals of agents.

Despite the large amount of information in a dynamic world, humans develop a structured understanding of the environment, learning to recognize scenarios and apply the appropriate behaviors. A longstanding goal in neuroscience is to understand how the brain learns these structures. The stability-plasticity dilemma asks how the brain is plastic enough to acquire memories quickly and yet stable enough to recall memories over a lifetime (Mermillod et al., 2013; Abraham and Robins, 2005). A related question is how does the brain avoid catastrophic forgetting, which is when a neural network forgets previously learned skills after

being trained on new skills (French, 1999; Soltoggio et al., 2017; Kirkpatrick et al., 2017)? We believe that the brain avoids catastrophic forgetting and balances stability and plasticity by storing information in schemas, or memory items bound together by common contexts.

Our work builds on existing theories of memory consolidation, including Complementary Learning Systems (CLS), which states that memories are acquired through rapid associations in the hippocampus and then gradually stored in long-term connections within the neocortex (McClelland et al., 1995; Kumaran et al., 2016). This aligns with hippocampal indexing theory (Teyler and DiScenna, 1986), which states that memories in the form of neocortical activation patterns are stored as indices in the HPC, which are later used to aid recall. Tse et al. (2007) later showed that new memories are acquired much more quickly when consistent with a preexisting schema. In their experiment, rats learned a schema of spatially arranged food wells, each containing a different food. Once familiar with the schema, two new food wells were introduced. The rats rapidly consolidated this new information into the neocortex within a day. Lesion studies showed a dependency on the HPC for this short time span of consolidation. Further studies by Tse et al. (2011) show activation of plasticity-related genes in the medial prefrontal cortex (mPFC) and related regions, suggesting their involvement in rapid consolidation.

Studies of connectivity between the mPFC and HPC have yielded theories of how these areas process schemas. Van Kesteren et al. (2013) introduced the SLIMM framework (Schema-Linked Interactions between Medial prefrontal and Medial temporal regions), which states that when a stimulus is familiar to a schema, the mPFC inhibits the HPC. However, if the stimulus is novel, the HPC activates to encode the information. In this way, the two brain areas enhance memory acquisition via different pathways. As for how the mPFC and HPC communicate, Eichenbaum (2017) hypothesized that theta phase synchronization between the mPFC and MTL is controlled by the thalamic nucleus reuniens (Re) to change the directional flow of information when encoding or retrieving. As theta oscillations control

many aspects of timing and control in the HPC, they play an important role in schema consolidation.

In addition to the HPC-mPFC pathways for memory consolidation, supporting areas modulate the speed of consolidation. The neuromodulatory system is important for detecting salience and making quick adaptations (Krichmar, 2008). The basal forebrain (BF) modulates attention and cortical information processing (Baxter and Chiba, 1999), and the locus coeruleus (LC) modulates network activity in response to environmental changes (Aston-Jones and Cohen, 2005a). The LC also drives single trial learning of new information via inputs to the HPC (Wagatsuma et al., 2018). Moreover, activity from the LC selectively increases oscillations in the HPC in the theta and gamma range according to novelty (Walling et al., 2011; Berridge and Foote, 1991), suggesting that it may be involved in single-trial learning.

This type of processing is likely necessary for the fast learning that occurs when novel stimuli are introduced within a familiar schema. Based on this background, we created a neural network architecture with the simulated brain regions and pathways comparable to those described above. The network portrays the involved neurobiological mechanisms at a level of abstraction that transfers easily to addressing the problem of avoid catastrophic forgetting in artificial neural networks.

4.2 Methods

4.2.1 Contrastive Hebbian Learning

To model representations of tasks, a multilayer network can store information of increasing levels of abstraction from input to output layers. Backpropagation is commonly used to

train such networks, and has had many successful applications in artificial neural networks (Rumelhart et al., 1988; LeCun et al., 2015). However, many view backpropagation as biologically implausible, as there is no widely accepted mechanism in the brain to account for sending error signals backwards along one-way synapses. An alternative account for developing representations in the brain may be contrastive Hebbian learning (CHL) (Movellan, 1991), which uses a local Hebbian learning rule and does not require explicit calculations of an error gradient.

Given a multilayered network with layers 0 through L, neuron activations of the kth layer are denoted as vector $\mathbf{x}_k$ and weight matrices from the k-1 to kth layer are denoted as W_k. Each weight matrix W_k has a feedback matrix of γW_k^T such that every weight has a feedback weight of the same value but scaled by γ. The learning process consists of cycling between three phases of the network. The first phase is known as the free phase of the network, in which the input layer $\mathbf{x}_0$ is fixed and the following equation is applied to update neurons in layer k from $k = 1$ to L at time t:

$$\mathbf{x}_k(t) = f_k(W_k \mathbf{x}_{k-1}(t-1) + \gamma W_{k+1}^T \mathbf{x}_{k+1}(t-1)) \tag{4.1}$$

where f is any monotonically increasing transfer function. This equation is applied for T_s time steps, which is when network activity converges to a fixed point. The resulting settled activity for $\mathbf{x}_k$ is noted as $\check{\mathbf{x}}_k$, which is the final neural activity for the free phase. The network then transitions to the clamped phase, in which the input layer is fixed as before and the output layer is fixed to a target value, as in supervised learning. Again neuron activities are updated using equation 4.1 for T_s time steps which allows the network activity to converge. The settled activity for $\mathbf{x}_k$ is noted as $\hat{\mathbf{x}}_k$, which is the final neural activity for the clamped phase. The third phase combines an anti-Hebbian update rule for neurons in

the free phase and Hebbian update rule for neurons in the clamped phase:

$$\Delta W_k = \eta_{CHL}(\hat{\mathbf{x}}_k \hat{\mathbf{x}}_{k-1}^T - \check{\mathbf{x}}_k \check{\mathbf{x}}_{k-1}^T), k = 1, ..., L. \tag{4.2}$$

Although the local learning rule is more biologically plausible than backpropagation, the use of symmetric weights is viewed as less plausible. To address this, versions of CHL that do not use symmetric weights have been implemented (Detorakis et al., 2018). However, we use the original version for simplicity.

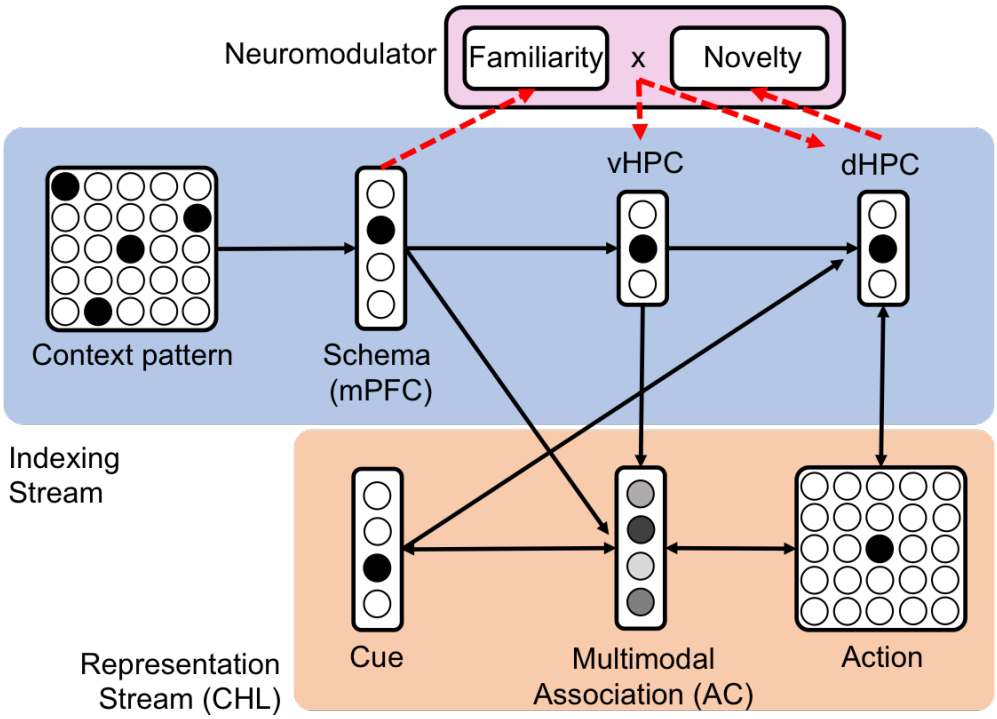

Figure 4.1: Overview of network. The light blue box contains the Indexing Stream, including the ventral hippocampus (vHPC) and dorsal hippocampus (dHPC). The light orange box contains the Representation Stream, including the cue, medial prefrontal cortex (mPFC), multimodal layer (AC), and action layer. Bidirectional weights between layers in the Representational Stream are learned via contrastive Hebbian learning (CHL). Weights from the Indexing Stream are trained using the standard Hebbian learning rule. Dotted lines indicate influences of the neuromodulatory area, which contains submodules of novelty and familiarity. Weights extend to these modules from the mPFC and dHPC. Neuromodulator activity impacts how often the vHPC and dHPC are clamped and unclamped while learning the task via contrastive Hebbian learning (CHL).

4.3 Neural Model

Our model consists of two main information streams, the Indexing Stream and the Representation Stream, in a network that is trained on context-dependent tasks such as the one found in (Tse et al., 2007). Figure 4.1 shows an overview of the network.

4.3.1 Indexing Stream

The Indexing Stream begins with a context pattern, which can be any encoding of context using patterns of neuron activity. In our case, we used a 2D grid with inputs of 1 if a food well existed in that grid location or 0 otherwise. This input projects to the mPFC for the schema to be learned from sensory input. The dynamic mPFC neuron activity is calculated by the following synaptic input equation at time t:

$$\mathbf{x}_k(t) = f_k(W_k \mathbf{x}_{k-1}(t-1)), \tag{4.3}$$

where layer k is the mPFC layer, layer k-1 is the context pattern, and f_k is the Rectified Linear Unit (ReLU) transfer function:

$$f(\mathbf{x}) = max(\mathbf{x}, 0). \tag{4.4}$$

A hard winner-take-all selection is then applied, in which all activations are set to zero except for the one with maximum value. Weights from the context pattern to the mPFC are then trained by the standard Hebbian learning rule:

$$\Delta W_k = \eta_{pattern} \mathbf{x}_k \mathbf{x}_{k-1}^T, \tag{4.5}$$

where $\mathbf{x}_k$ is the mPFC layer, $\eta_{pattern}$ is the learning rate, and $\mathbf{x}_{k-1}$ is the context pattern layer. The weights are normalized such that the norm of weight vectors going to each post-synaptic neuron i is 1:

$$w_i = \frac{w_i}{||\mathbf{w}||} \tag{4.6}$$

where $\mathbf{w}$ is the vector of weights going to one postsynaptic neuron, and w_i is an individual weight in $\mathbf{w}$. The Indexing Stream continues on to the dorsal hippocampus (dHPC) and ventral hippocampus (vHPC), matching observations that the HPC encodes with increased specificity along the dorsal-ventral axis Jung et al. (1994). The vHPC learns an index of mPFC activity using the same synaptic input function and learning rule described in Equations 4.3-4.6. The dHPC learns indices in the same way as the vHPC, except that it uses a learning rate of $\eta_{indexing}$ and has weights from the vHPC, cue, and action selection layer. The term $\eta_{indexing}$ is a separate learning rate used for weights that index activity from the Representation Stream. Rather than indexing context, the dHPC indexes triplets of context, cue, and action. This agrees with how context is encoded in the vHPC and specific experiences are encoded in the dHPC in Eichenbaum (2017). It also aligns with the fact that selectivity of encoding increases from the ventral to dorsal end of the HPC (Jung et al., 1994).

4.3.2 Representation Stream

The Representation Stream is a multilayered network with sensory cue input areas and the mPFC that encodes the current schema or context. The middle layer of the Representation Stream acts as the association cortex (AC), and makes multimodal associations of its inputs and conjoins context and cue information. The output layer selects an action response to the sensory cue, which is a spatial grid with activation of 1 at the locations of food cues and

0 otherwise. To train the correct actions, the multilayered network uses CHL as described in the background section, with a transfer function as in Equation 4.4. The alternation of clamped and free phases is controlled by the Indexing Stream. The dHPC alternates between clamping and unclamping the action layer, providing input to the action layer during the clamped stage of contrastive Hebbian learning. During clamping, the winning neuron in the vHPC gates neurons in the AC. This is done via a static weight matrix of strong inhibitory weights from the vHPC to AC layer, with sparse random excitatory weights that allow only some neurons in the AC to be active. All weights in this matrix are first initialized by a strong negative value of w_{inh}, then a random selection of the weights in this matrix is set to 0. The number of weights selected to be 0 is determined by P, which is a number in the range of 0 and 1 that defines the proportion of randomly selected weights. This is meant to mimic hippocampal indexing. While there is little evidence that the HPC projects widespread inhibition to the representation areas of the brain, the mix of inhibition and zeroed weights allows specified patterns of neurons to be active with their usual activity levels, which is meant to mimic an attentional sharpening effect in the model.

Taken as a whole, the Representation Stream models how the neocortex learns representations (Hawkins et al., 2017), with the Indexing Stream driving the learning process and preventing catastrophic forgetting by allocating different sets of AC neurons for each task. By using CHL for the Representation Stream, we form an equivalence with backpropagation methods that allows us to expand our model to help improve traditional neural networks in the future.

4.3.3 Novelty and Schema Familiarity

In the SLIMM framework, the encoding strength combines schema familiarity and cue novelty. Novelty is defined as how infrequently a stimulus has been experienced before, whereas

schema familiarity is how frequently a schema has been experienced. Furthermore, the SLIMM framework proposes that resonance occurs in the presence of schema familiarity. However, SLIMM also suggests that the mPFC inhibits the HPC, whereas we suggest that a combination of schema familiarity and novelty from the mPFC and dHPC, respectively, affect learning by controlling oscillatory activity in the HPC.

In our model, a neuromodulatory area detects novelty in the presence of a familiar schema and modulates the strength of learning that occurs within the Representation Stream. To detect novelty, each neuron in the dHPC projects to a novelty submodule with $w_{novelty}$ as the starting weight. $w_{novelty}$ represents the baseline level of surprise when a new stimulus is presented. Whenever the activity of the dHPC is updated, the activity of the novelty submodule is the rectified weighted sum of inputs from dHPC after applying winner-take-all, as in equations 4.3 and 4.4. The weights from the dHPC to the novelty submodule are then updated with an anti-Hebbian learning rule:

$$\Delta W = -\eta_{indexing} \mathbf{x}_{novelty} \mathbf{x}_{dHPC}^{T}. \tag{4.7}$$

where W is the matrix of weights from the dHPC to novelty submodule, $\eta_{indexing}$ is the learning rate, $x_{novelty}$ is the activity of the novelty submodule, and x_{dHPC} is the activity of neurons in the dHPC. Therefore, the weight between an active dHPC neuron and novelty submodule will experience long term depression and decrease the novelty score of a stimulus after many exposures. Since the dHPC uses winner-take-all, each weight from dHPC represents an individual novelty score for each possible triplet. The activity of the familiarity module, $x_{familiarity}$ is similarly calculated through the weighted sum of inputs from the mPFC after winner-take-all, as in Equation 4.3. However, rather than a rectified linear unit, we apply a shifted sigmoidal transfer function:

$$f(\mathbf{x}) = \frac{1}{1 + e^{-s(\mathbf{x} - x_{shift})}}. \tag{4.8}$$

where s is the sigmoidal gain and x_{shift} is the amount of input shift. The shifted sigmoidal transfer function ensures that the familiarity module requires a baseline amount of training on a schema to be considered familiar with it, and that familiarity does not continue to increase in an unbounded manner with extended exposure. The activity of the familiarity module is thus mostly bimodal, with a very low activity if the schema is unfamiliar and a high activity if the schema is familiar. The effect of the sigmoidal function will be seen in the results. The weights from the mPFC start with the same value of w_{fam}, which is a very small value close to zero that represents low familiarity of schemas prior to training. These weights are updated after mPFC activity is updated, using the Hebbian learning rule from Equation 4.5 with $\eta_{pattern}$ as the learning rate. $\eta_{pattern}$ is small to model the long-term consolidation of schemas. The neuromodulator activity is then set to the simple product of activity from the familiarity and novelty modules:

$$neuromodulator = x_{novelty} * x_{familiarity} \tag{4.9}$$

This value determines the number of times the vHPC and dHPC will clamp and unclamp the representation layer in a single trial and thus determine the number of extra epochs in a trial that are added to a default number of epochs, $e_{default}$:

$$epochs = e_{default} + neuromodulator * e_{boost} \tag{4.10}$$

The model therefore suggests that the rapid consolidation that occurred in Tse et al. was due to increased replay of important information during quiet waking periods. This follows the idea from the SLIMM framework suggesting that resonance occurs in the presence of schema familiarity.

When training on a task, the network runs trials of training that consist of four phases as shown in Figure 4.2. Each trial consists of many epochs of training, the number of which is

determined by Equation 4.10. This mimics the high resonance caused by schema familiarity, as postulated in the SLIMM framework. Each epoch consists of an indexing phase, a CHL free phase, and a CHL clamped phase. During indexing, the mPFC, vHPC, and dHPC form indices using the unsupervised Hebbian rule and winner-take-all rule described previously. During the Free Phase (Figure 4.2B) the Representation Stream runs freely. During the Clamped Phase (Figure 4.2C), the the Representation Stream is clamped to input from the dHPC using Equation 4.3, and the CHL learning rule is applied. Also during clamping, the AC receives input from the vHPC also using Equation 4.3, which effectively inhibits most of the AC neurons except for the few that have a weight of 0 from the winning vHPC neuron. At the beginning of a trial, the number of training epochs is undetermined, but tentatively set at e_{settle} epochs (Figure 4.2A). During this time, activity levels of the neuromodulator are tracked, and the maximum neuromodulator activity found within this period is used to determine the ultimate number of training epochs within the trial. After each trial, the performance of the network is measured during the Test Phase (Figure 4.2D) by presenting a cue to the network and allowing the network to settle on an action. The network parameters used in our network can be found in Table 4.1.

Population Sizes		Learning Parameters	
N_{cue}	18	T_s	5
N_{mPFC}	10	$\eta_{indexing}$	.1
$N_{multimodal}$	40	$\eta_{pattern}$	.0001
$N_{context}$	25	η_{CHL}	.001
N_{vHPC}	5	g	.001
N_{dHPC}	40	e_{boost}	1000
		$e_{default}$	600
		e_{settle}	20
		w_{min}	0.3
		w_{max}	0.8
		w_{inh}	-10
		w_{fam}	.0001
		$w_{novelty}$	1
		s	200
		x_{shift}	.03
		P	0.3

Table 4.1: Parameters used in schema and memory consolidation model.

4.3.4 Experimental Design and Statistical Analyses

We validated our model by simulating the experiments in Tse et al. (2007). We simulated a population of 20 rats. For each individual, weights from vHPC to AC were sparsely connected with a probability P and set to w_{inh}. Weights to the novelty and familiarity module were all set to $w_{novelty}$ and $w_{familiarity}$, respectively. Remaining network weights were initialized along a uniform distribution on the range w_{min} and w_{max}, fully connected. The arena was discretized into a 5x5 grid, corresponding to location in the arena for the context pattern and action layers. Each trial consisted of multiple epochs to account for replays facilitated by oscillations during sleep and quiet waking periods. In Tse et al. (2007), performance was measured during training by counting the errors in a trial, and non-rewarded probe tests (PT) were performed intermittently by recording the amount of time spent searching the correct well when given a food cue. In our model, our test of performance was to present each flavor in a schema during the Test Phase. Upon presenting the cue, the network ran in Free Phase until the activity converged. In the action selection layer, the activity of the

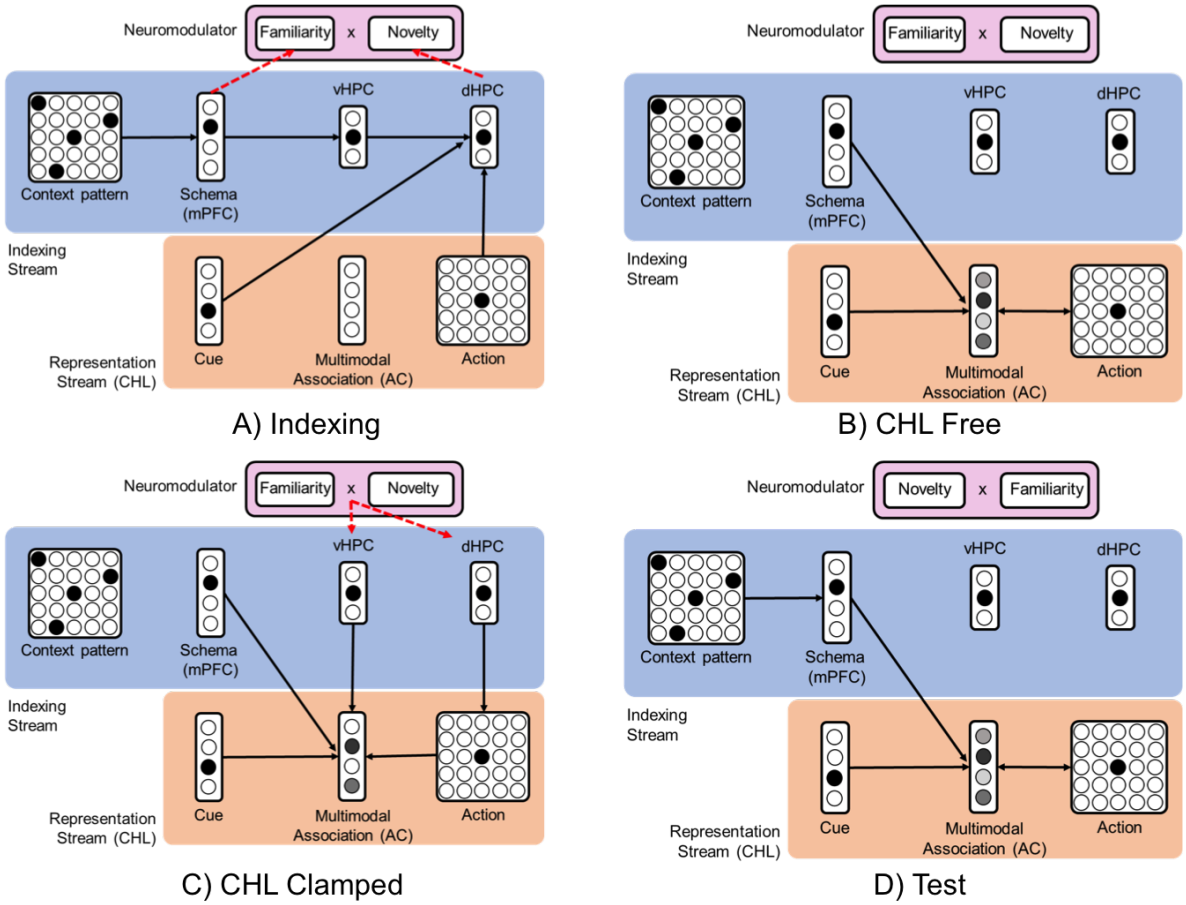

Figure 4.2: The four phases of a trial of training. A) In the Indexing Phase, the Indexing Stream forms indices of activity. Additionally, the activities of the novelty and familiarity modules of the neuromodulator are calculated, setting the ultimate activity of the neuromodulator to the product of these values. B) The Free Phase of CHL. C) The Clamped Phase of CHL. D) The Test Phase of a network for measuring performance during training and unrewarded probe tests. This is the same as the CHL Free Phase, except that mPFC activity is also calculated.

neuron corresponding to the correct location of the food was divided by the sum of all of the neurons corresponding to the wells in the arena. This value corresponds to the amount of time a rat would spend digging in a well given a food cue. We used this value to simulate performance during both training and unrewarded probe tests. In the Tse et al. (2007) experiment, the unrewarded probe test was conducted with a food cue and no food reward in the correct well, thus limiting learning for those trials. Although our experiment does not model reward representation in the brain, we approximate unrewarded probe tests by not

updating weights after cueing and running the network.

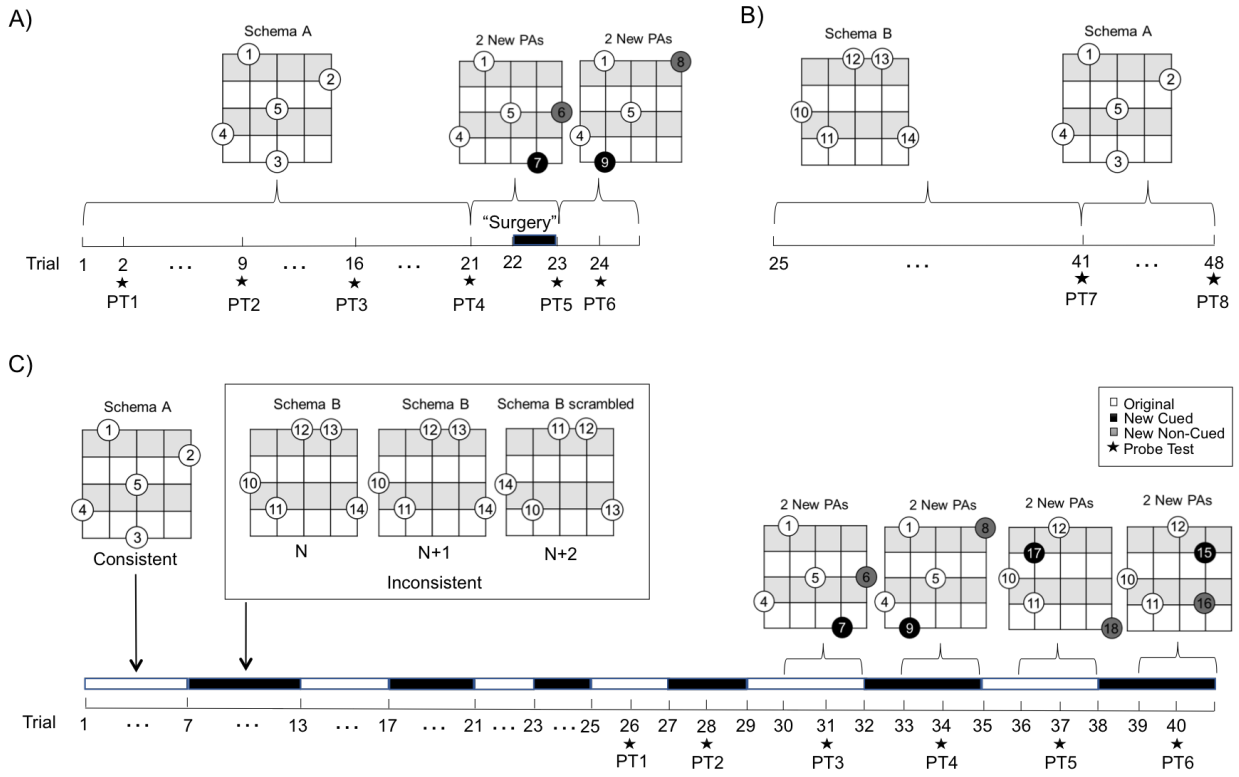

Figure 4.3: Timeline of experiments. A) Experiment 1 timeline. Schema A is trained for 20 days. Two new paired associations (PAs) are introduced on day 21. Lesioning occurs on day 22, and yet another two PAs are introduced on day 23. Probe tests are performed throughout. B) Experiment 2 timeline. Schema B is trained for 16 days, then Schema A is retrained for 7 days.

A timeline of the two replicated experiments can be seen in Figure 4.3. The first experiment was to train the network on the Schema A layout for 20 trials with paired associations (PA) between food cue and location (Figure 4.3A; PTs 1-3). On the 21st trial, two of the PAs were replaced by two new ones (Figure 4.3A). The original schema with two new PAs was trained for one trial, and then a probe test was performed by cuing one of the new foods and examining the output of the well locations of the new cued PA, new non-cued PA, and other wells (Figure 4.3A; PT 4). After that, the network was split into a control network and a lesioned HPC group. The HPC group was copied with weights from the original network and had all connections to and from the vHPC and dHPC removed. Another probe test was performed to see whether both groups could still recall the original schema as well as

the schema with two new PAs (Figure 4.3A; PT 5). Next, the two new PAs were replaced with yet another two new PAs and trained on both groups for one trial. Another probe test was performed after this (Figure 4.3A; PT 6). The second experiment was performed on the resulting networks from the the first experiment (Figure 4.3B). For 16 trials, both conditions were trained on an entirely new schema, Schema B, and a probe test was conducted (Figure 4.3B; PT 7). After this, the groups returned to train and test on the original schema for 7 days, with yet another probe test at the end (Figure 4.3B; PT8).

4.4 Results

Statistical significance was determined through the Wilcoxon rank sum test with Bonferroni correction. For performance results involving cued versus non-cued dig time, we compared cued and non-cued samples. For performance results of new PAs, we compared the samples of cued and non-cued, as well as cued and original. To test the differences between HPC and and control groups, we compared the cued samples of both groups. Unless otherwise noted, all findings are significant at $p<<.001$.

4.4.1 Experiment 1

The goal of the first experiment was to show that new information matching a schema can be quickly learned, and that the HPC is necessary for this learning. Figure 4.4A shows that the model was able to gradually learn Schema A over 20 trials of training. Figure 4.4B confirms this with probe tests, which show the proportion of neuron activity corresponding to the correct food location given a cue. Figure 4.4C shows the probe test results after training for one day on Schema A with two new PAs added. During the Test Phase, the longer dig time in the correct well shows that the new PAs were learned in just one trial, with much

more dig time in the cued wells than non-cued wells. This supports the finding that new information matching an existing schema is learned rapidly.

Next, the role of the HPC in consolidation was examined. In Figures 4.4D and 4.4E, upon splitting into two conditions of lesioned HPC and control, both groups were still able to recall the original schema, as well as the two new PAs. This suggests that the new information had been consolidated within a short period and no longer required the HPC for retrieval. This was due to the neuromodulator detecting when new information was present within a familiar schema and increasing the consolidation accordingly. When training yet another two PAs on both conditions in Figure 4.4F, the group with the lesioned HPC was unable to learn at all, while the control group was still able to learn. This suggests that the HPC was responsible for driving the index-based clamping and unclamping of the output layer, as CHL mechanism was unable to update the weights properly.

Figure 4.5 shows the weights of the network after training on the first experiment. Our results for the first experiment were able to capture most of the effects seen in (Tse et al., 2007). We were able to show that information is acquired rapidly when consistent with a prior schema and that the HPC is necessary for acquisition.

4.4.2 Experiment 2

The purpose of the second experiment was to test whether multiple schemas could be learned by the same network, and whether the HPC was necessary for this. Figure 4.6A shows that the control group was able to learn Schema B. However, the HPC-lesioned group was unable to learn, staying at chance levels the entire time. This is shown again in the probe tests in Figure 4.6B. When the HPC-lesioned model returned to retrain on Schema A in Figure 4.6C, it had good performance the entire time and retained the information learned prior to surgery, but did not improve performance beyond what it had learned in Experiment 1. The

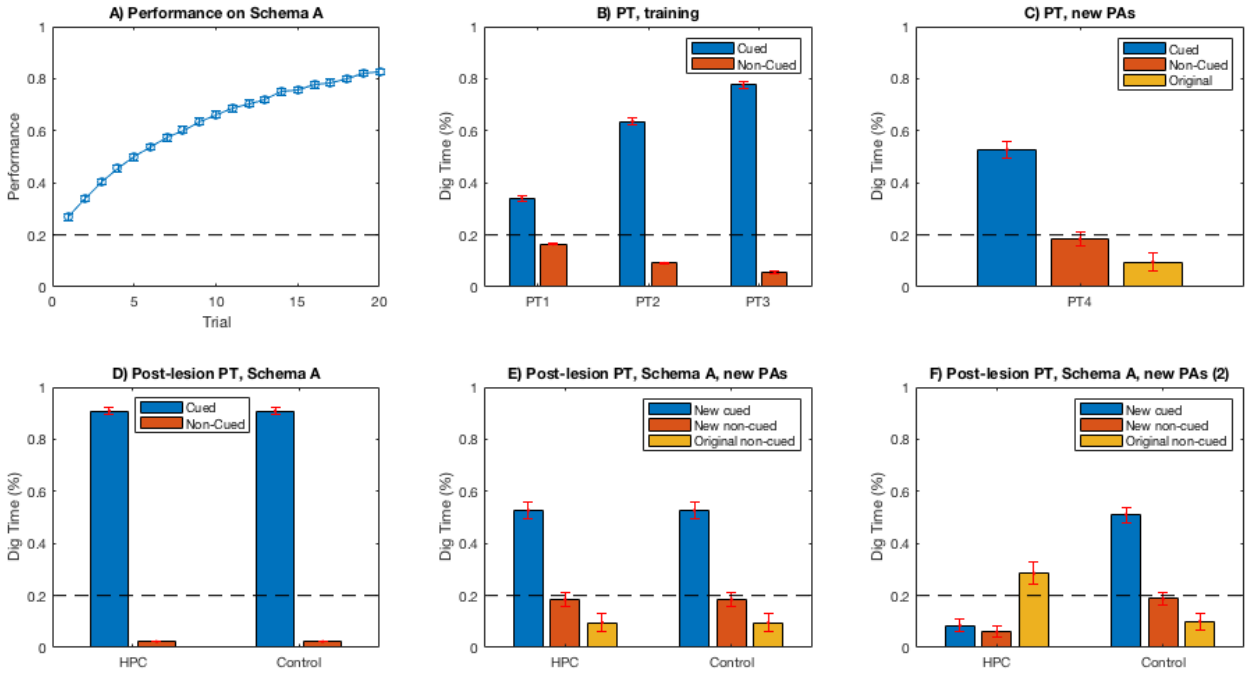

Figure 4.4: Results of replicating the first experiment of Tse et al. A) The performance over 20 trials, showing a gradual increase. B) Probe tests of trials 2, 9, 4, and 16, showing the proportion of activity of the correct well given a food cue compared to activity of the incorrect wells. C) Probe test after training the new PAs for 1 day, in which one of the new pairs is cued and the activities of the correct well, well of the other new pair, and original Schema A wells are compared. The new PAs were learned within one trial, as the dig time for the cued pair was significantly higher than the rest. D) Probe tests of Schema A after splitting into HPC-lesioned and control groups. Both conditions retained knowledge of the schema. E) Probe tests of the new PAs after splitting into HPC-lesioned and control groups. Both groups recalled the new PAs equally. F) Probe tests of Schema A after training another two new PAs. The HPC group could not learn.

control group displayed a minute decrease in performance of Schema A at the beginning, but quickly regained prior performance. The decrease was due to small overlaps in the gating patterns from vHPC. This is confirmed again in the probe tests in Figure 4.6D.

Figure 4.7 shows the weights of the network in the control condition after training on the second experiment. The results show that our network is able to learn multiple schemas in succession, without forgetting prior schemas. We were thus able to match the effect seen in Tse et al. (2007). A small difference is that our network retained information about Schema A much better than their experiments for both of their HPC-lesioned and control

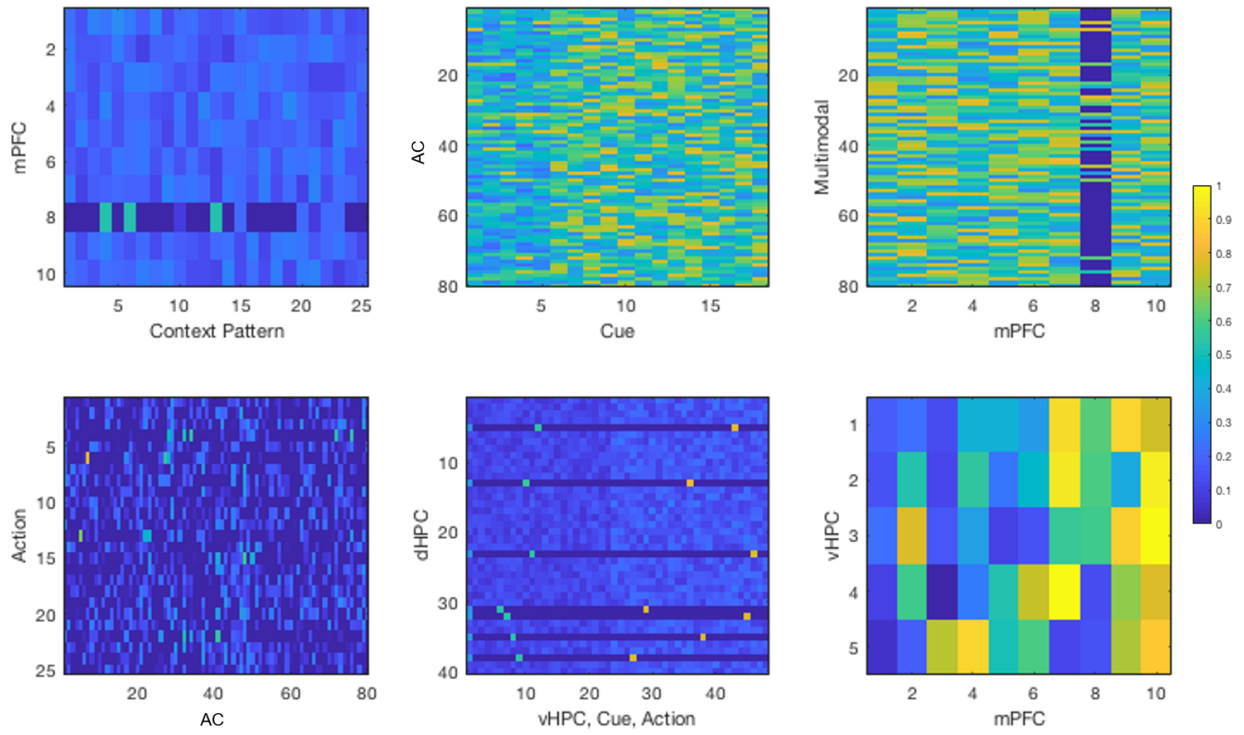

Figure 4.5: Weight matrices of the network of one simulated rat after simulating all of the first experiment. Rows represent post-synaptic layers and columns represent presynaptic layers.Weights from the context pattern to mPFC show the development of a distinct schema pattern encoded with stronger weights where the wells are located. Weights from the mPFC to the AC show the effect of the gating, in which the mPFC neuron representing Schema A is associated with a set of neurons in the AC. Weights from the AC to action layer show that the actions are dependent on activity from select AC neurons, suggesting that the AC neurons are learning specific features useful for action selection. Weights going from the vHPC, cue, and action to the dHPC are displayed in one matrix to show clear encodings of triplets with one neuron from each of the three areas. Weights from mPFC to vHPC show that there is not necessarily a one-to-one mapping from a winning mPFC neuron to a vHPC neuron, but that the schema information is transferred in a distributed manner.

groups.

4.4.3 Neural Activity

To gain a better understanding of the network activity, we plotted the neuron traces of the mPFC, vHPC, and dHPC before winner-take-all was applied. Figure 4.8 shows the neuron

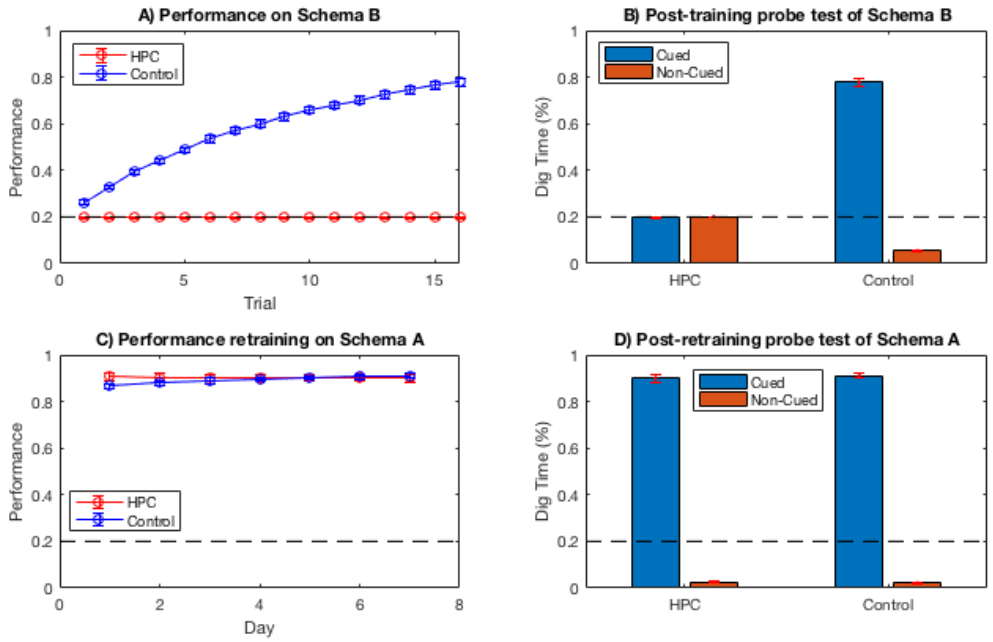

Figure 4.6: Results of replicating the second experiment of Tse et al. A) The performance over 16 trials of training Schema B. The control group was able to learn the new schema while the HPC group was not. B) Probe test after training Schema B, confirming part A. C) Performance over 7 trials of retraining on Schema A. Both groups retained Schema A, though the control group recovered from a very minimal forgetting of the schema initially. D) Probe test after retraining on Schema A, confirming part C.

activities for the first two experiments, with Schema A and new PAs for Experiment 1, and Schema B with a retraining of Schema A for Experiment 2. Each colored line in the figure represents the activity of a single neuron in a single simulated rat. The black vertical lines separate epochs into trials. The neuron traces show that the mPFC chooses a single neuron to represent Schema A and continues to strengthen its weights as it is trained. When new PAs are introduced, the same neuron is still active, but decreases in activity. When Schema B is introduced at the beginning of Experiment 1, a different neuron becomes active, and when Schema A is reintroduced immediately after training Schema B, the original winning neuron returns. By viewing the spacing between black vertical lines, we see that the large spacing for trials with new PAs indicates that more epochs of training occur during those times. This is due to the novelty and familiarity detection from the neuromodulator. vHPC activity in Figure 4.8B shows the same effect, except that the weights of all of the neurons

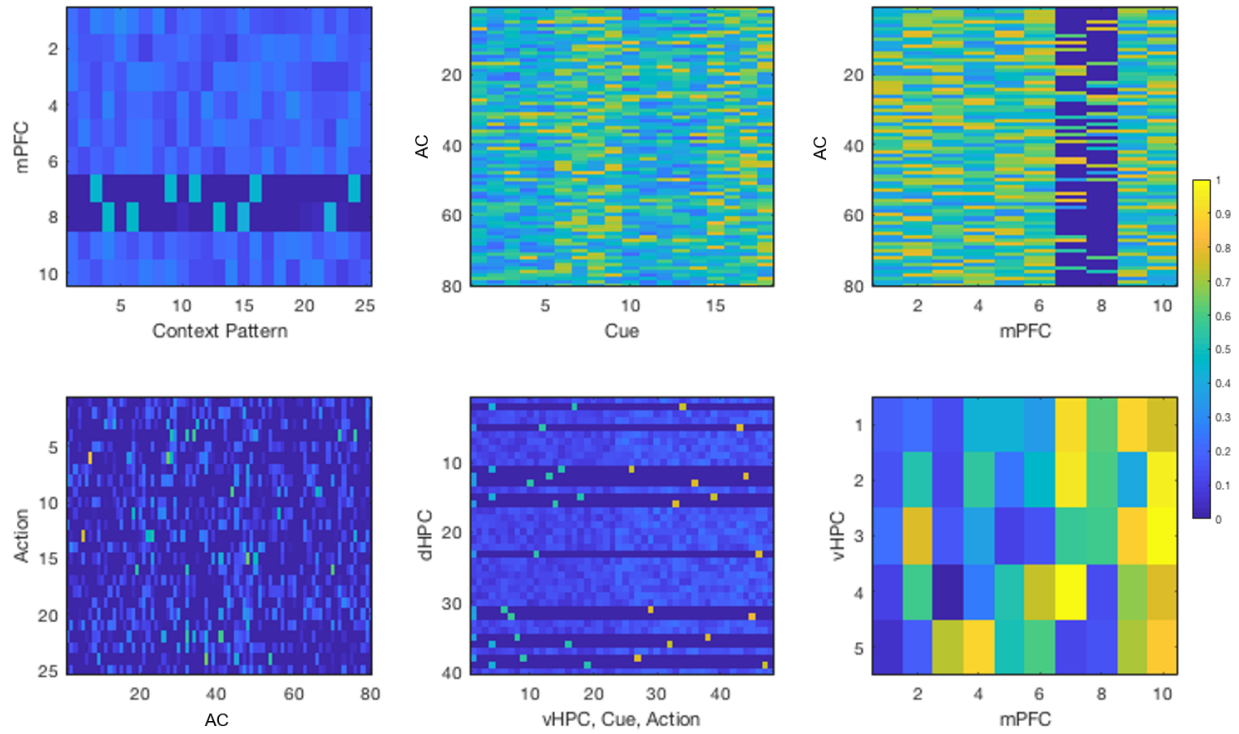

Figure 4.7: Weight matrices of the control network after simulating all of the second experiment. Rows represent post-synaptic layers and columns represent presynaptic layers. Weights from the context pattern to the mPFC now show two distinct schemas, with stronger patterns than seen in the first experiment. Weights from mPFC to the AC show two sets of gating patterns. Weights to the dHPC now show twice the amount of triplets as before, reflecting that triplets from two schemas are being encoded.

together increase and decrease, as they are all affected by the rising and falling activity levels of mPFC. dHPC works similarly, as seen in Figure 4.8, although different winners are chosen at every epoch. For the dHPC we display neural traces for the first 100 epochs in the first trial epochs, due to the frequent switching of winners.

To show the effects of schema familiarity and novelty on neuromodulator activity, Figure 4.9 displays the activities of the familiarity module, novelty module, and neuromodulator over the first 21 trials of Experiment 1. Due to the Hebbian learning of connections from the mPFC to the familiarity module, familiarity increases over time. Since it uses a sigmoidal transfer function, the increase is step-like, increasing from 0 to 1 swiftly. Novelty starts high when Schema A is introduced, and quickly drops to 0 due to the anti-Hebbian learning rule.

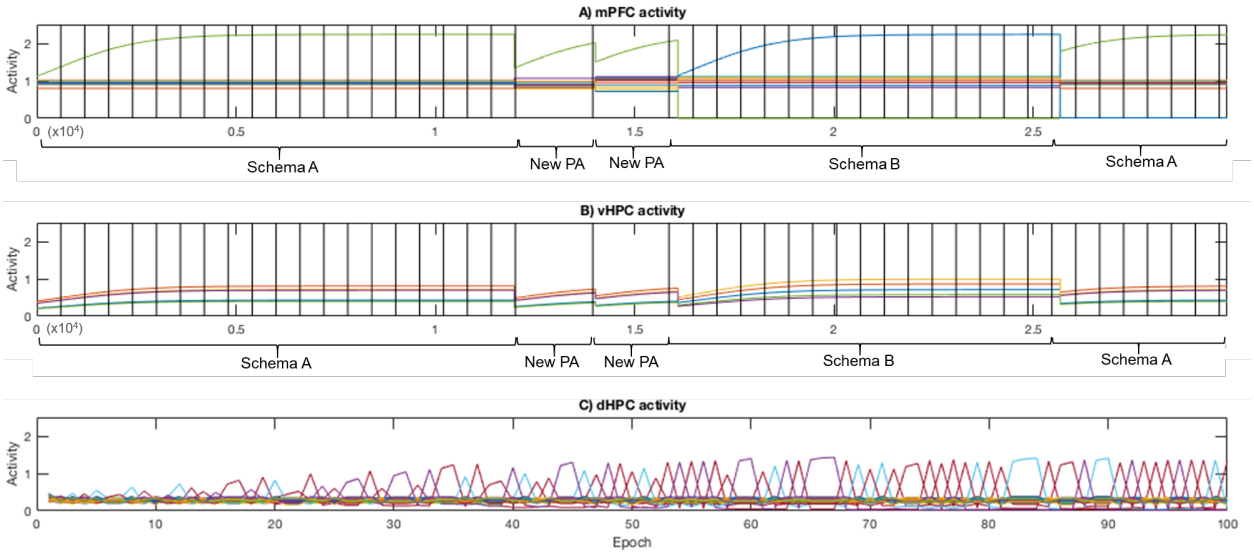

Figure 4.8: Neural activity for the network of a single simulated rat while performing the first and second experiments combined. Each colored line represents the activity of one neuron. Each vertical black line represents the separation of epochs into trials. Activity is measured before winner-take-all is applied. A) The mPFC activities for the first and second experiments are shown in sequence, with the training of Schema A and two instances of new PAs for the first experiment, and the training of Schema B and return of Schema A for the second experiment. When training on Schema A, one mPFC neuron is consistently chosen as winner, as its weight values increase over time. When new PAs are introduced, the winner remains the same, but decreases in activity. At the start of Experiment 2, Schema B is introduced and a new mPFC neuron wins. The number of epochs greatly increases due to the novelty. When Schema A is retrained, the neuron previously associated with schema A becomes active again. B) The neurons in vHPC follow the same general trend as the mPFC neurons. However, all neurons increase and decrease together, as they all take input from the winning mPFC neuron. C) Activity of the dHPC neurons for the first 100 epochs of the first trial is shown. As the density of switching of dHPC winners occurs at every epoch rather than at every trial, it is necessary to display the activity at the epoch level. At each epoch, a different dHPC neuron is selected to have its weights increase, as a different triplet is present each time. The activities of winning neurons gradually increase for 40 epochs, remaining stable afterwards.

When two new PAs are introduced at Trial 21, novelty returns to a high state. Taking the product of novelty and familiarity, the activity of the neuromodulator increases only when familiarity and novelty are high. Since the activity of the neuromodulator is proportional to the number of training epochs in a trial, this leads to the desired behavior of increased learning when new PAs are introduced to an existing schema.

We also tracked the activity of the neuromodulator for all three experiments. Figure 4.10A shows the number of training epochs in each trial of the first and second experiments. As a reminder, each epoch within a trial consists of an Indexing Phase, a Free Phase, and a Clamped Phase. The first 21 trials of the first experiment are on the left of the black vertical line, and the remaining trials to the right of the vertical line are for the second experiment. Each individual colored line represents the neuromodulator activity of a single run, with 20 runs total. When new PAs were introduced on trial 21 of the first experiment, the familiarity of Schema A multiplied by the novelty of the new stimuli caused a sharp spike in neuromodulator activity, increasing the number of epochs for those trials. The neuromodulator activity for the second experiment remained low, as there were no new PAs introduced. Figure 4.10A shows the index of the winning mPFC neuron in each trial, with a different colored line for each individual run. Every time the schema changes, the index of the winning neuron changes accordingly. Figures 4.10C and 4.10D show similar effects, with large increases in neuromodulator activity when there is a new PA and switching of active schema neurons whenever the schema changes from consistent to inconsistent and vice versa. The white regions represent consistent schema trials while the gray shaded regions represent inconsistent schema trials.

4.4.4 Effects of neuromodulation

To study how the neuromodulator influences learning, we removed connections to and from the novelty and familiarity modules in the network. Rather than boosting the number of epochs using neuromodulator activity, we used a constant number of epochs for each trial. We repeated the first part of Experiment 1 with the same number of epochs for every trial, trying different values for the constant number of epochs. As before, the network was trained on Schema A for 20 trials and training a new PA was introduced on the 21st trial. As seen in Figure 4.11, training with more epochs per trial leads to faster learning and

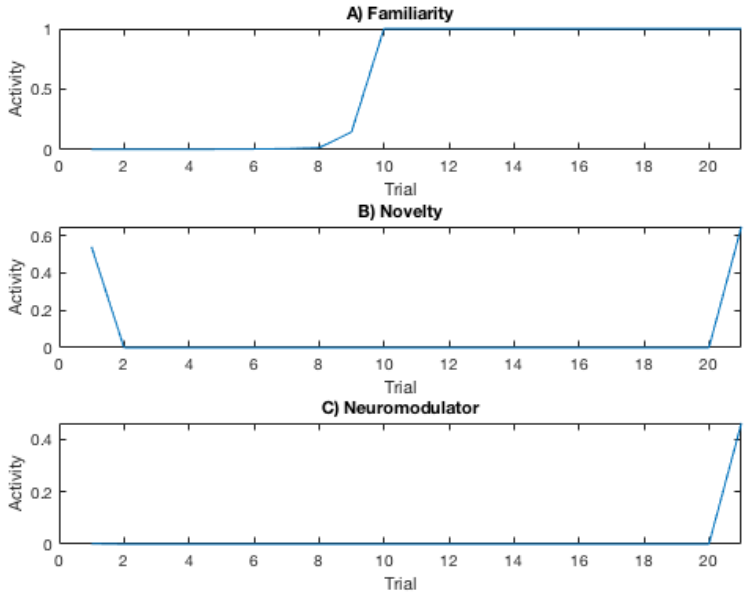

Figure 4.9: Combining familiarity and novelty for neuromodulation. A) Familiarity increases as the network is trained on a schema. Due to the sigmoidal transfer function, familiarity activity is step-like, going from unfamiliar to familiar over some training. B) Novelty starts at a high value whenever there are new PAs, which occurs on the first trial when the schema is introduced, and on the 21st trial when two PAs are replaced. C) At each trial, the product of familiarity and novelty lead to an increase of activity when schema familiarity is high and novelty is high.

better performance on the new PA. It is therefore not required to have a neuromodulator for rapid learning of new information. However, the number of total epochs over all trials can be greatly reduced if the model is able to detect novelty within the schema and adjust the learning accordingly. The smaller number of epochs reduces the overall learning time of the experiment. Compared to the conditions with a flat number of epochs per trial, we found that our original network with the neuromodulator had better performance on the new PAs than the other successful conditions, despite having fewer total epochs. It was therefore able to conserve network training time overall.

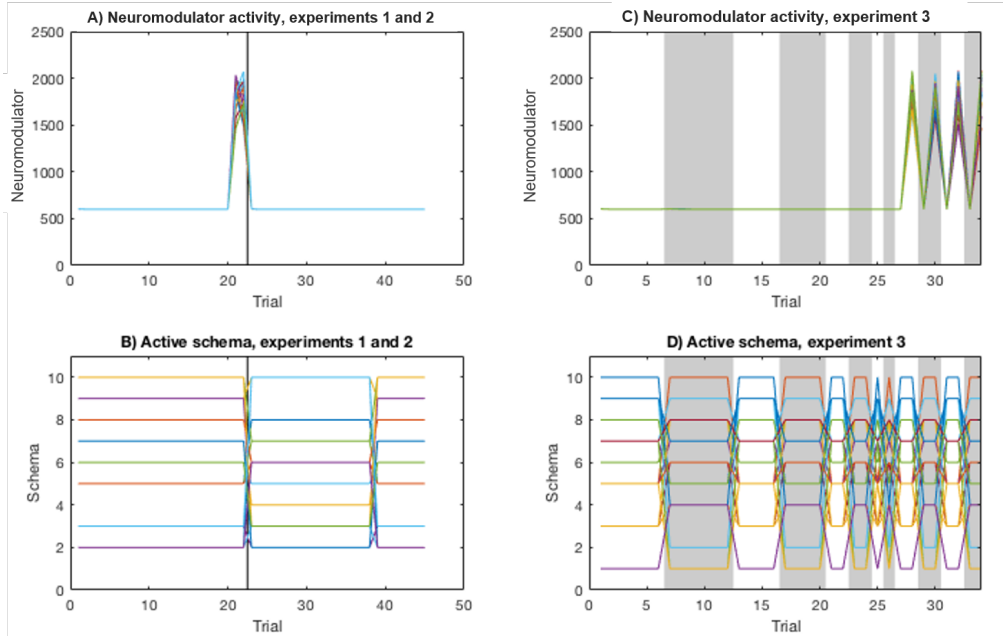

Figure 4.10: Neuromodulator activity and winning schemas. Each colored line represents the activity of the neuromodulator from an individual network out of 20. A) Neuromodulator activity expressed by the number of epochs trained per trial. The vertical black line separates experiments 1 and 2. On trial 21, the number of epochs rises sharply due to the combination of familiar schema and novel stimulus. B) Index of winning mPFC neuron in each trial of experiments 1 and 2. The index switches clearly each time the schema switches.

4.5 Discussion

Confirming the results of Tse et al. (2007), we showed that our biologically plausible neural network was able to learn schemas over time and quickly assimilate new information if it was consistent with a prior schema. The learning was done through an indexing stream in which the mPFC and HPC projected context-dependent patterns onto the representation stream, and the rapid consolidation was done by enhancing replay activity of novel and familiar information. The network was also able to learn multiple schemas without catastrophic forgetting, by maintaining separate sets of AC neurons for tasks within different schemas.

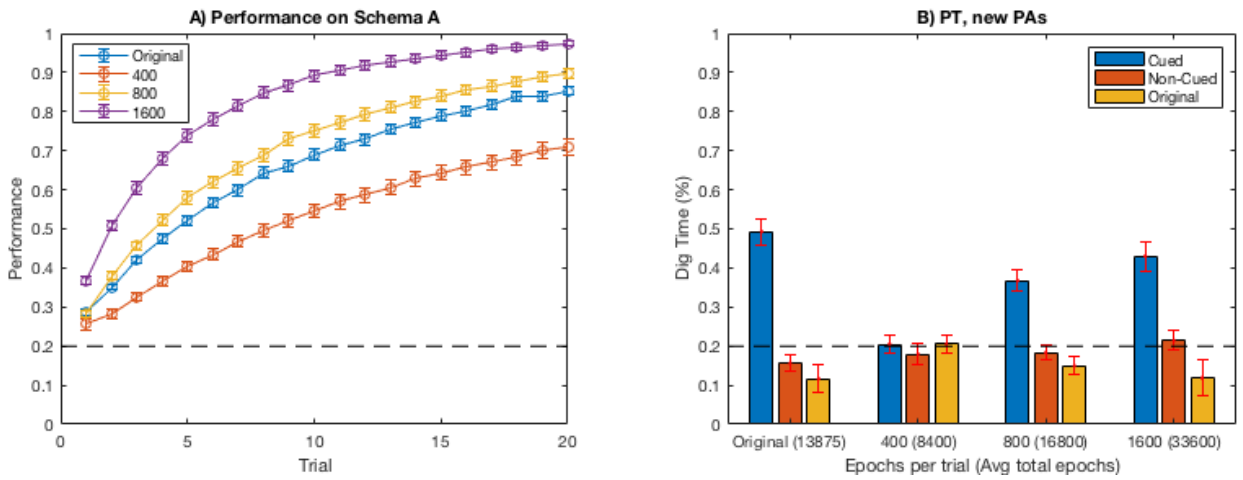

Figure 4.11: A) Performances of each condition. The blue line represents the original network and the remaining lines use a flat number of epochs as indicated in the legend. All conditions are able to learn the schema, but with different learning rates. B) Probe tests of each condition after introducing a new PA. The average number of epochs for each trial is displayed in parentheses. The performance of the probe test increases as more epochs are trained per trial. However, the original network can get a performance equivalent to the other conditions, but with far fewer epochs of training.

4.5.1 Hippocampal Indexing

The network highlighted diverse roles of indexing by the HPC. Eichenbaum (2017) proposed that specific memories are represented in the dHPC whereas contextual information is represented in the vHPC. Combined with indexing theory, our model showed that indexing separates representations of objects, spatial layouts and tasks by the contexts in which they belong. This modularity makes it less likely that learning new information in new contexts would overwrite previously learned information in old contexts. The indexing of the dorsal HPC is necessary for driving the consolidation processes that transfer information to long term storage.

The indexing behavior is comparable to a recent approach to avoiding catastrophic forgetting by Nakano and Hattori (2017), in which the intermediate layers of a deep neural network are gated by patterns that differ by context. A paper by Masse et al. (2018) has the similar idea of using CHL as a plausible deep representation of information, and applying "pseudopatterns"

alongside their regular training patterns for better separation. Our experiments explain in more depth how these patterns are formed and employed throughout different stages of the learning process. The central location of the hippocampus makes it a likely candidate for effecting context-dependent gating within the network. Its connections to the mPFC allow the formation of context-dependent indices, and its wide connectivity to the whole neocortex gives it the ability to gate information at many levels of representation.

4.5.2 Neuromodulation and Novelty Detection

The gating of patterns by the HPC is not employed evenly at all times, but depends heavily on external factors such as novelty, uncertainty, and reward. Therefore, our model predicts that gating of context information is controlled by neuromodulatory areas. This could explain how the brain controls phases of learning in such a flexible manner. Our model suggests that schema familiarity could be detected by the mPFC and novelty could be detected by the HPC. We propose that neuromodulation monitors these signals and amplifies learning and consolidation of new (i.e. novel or unfamiliar) information while sparing old information. Our simulated neuromodulator may have biological correlates in the locus coeruleus, as the LC reacts to sudden changes in schemas and causes changes in theta oscillations within the HPC. However, the combination of novelty and familiarity is also reminiscent of the basal forebrain functionality suggested by Yu and Dayan (2005), in that the level of uncertainty is framed within a specific context. Other areas such as the dopaminergic ventral tegmental area (VTA) may be involved in neuromodulatory gating as well, as it has inbound and outbound connections with the hippocampus that control the speed of learning according to reward and novelty (Otmakhova et al., 2013). It is important to note that the speed of learning in our model is not defined by the number of epochs used for training, but by the number of epochs that occur within a day of consolidation. This is meant to approximate how hippocampal replays may increase for salient information.

4.5.3 Interactions Between the Medial Prefrontal Cortex and Hippocampus

There are differing opinions on how interaction between the mPFC and HPC is involved in context-dependent tasks. The SLIMM model suggests a competitive relationship, with activation of the mPFC inhibiting the HPC when a stimulus is congruent with a prior schema. On the other hand, Preston and Eichenbaum (2013) suggest a more cooperative interaction, with the mPFC drawing specific memories from the vHPC, and in turn influencing the dHPC via entorhinal inputs. As the mPFC is also known to mediate attention shifting in context-dependent tasks (Birrell and Brown, 2000), it is likely that shifts in schemas cause the mPFC to change the activity of the HPC. In our experiment, the presence of new PAs within an existing schema should require both the mPFC and HPC to express the familiarity of the schema and novelty of the new PAs. In our model, the main roles of the mPFC are to provide contextual input to the representation stream and apply top down control of the HPC to change which specific neurons are active in the HPC to effectively separate tasks by schema. To further align with mPFC functions, future work should consider a distributed encoding in the mPFC, which could better represent overlapping information between schemas. It is also important to note that no direct anatomical connections exist from the mPFC and HPC, and instead are routed through the thalamus. Future models including the thalamus could further test the theory proposed by Eichenbaum (2017) that the thalamus controls information flow between the mPFC and HPC.

By uniting the theories of hippocampal indexing and interactions between the mPFC and HPC, we generate new and testable hypotheses that can be validated experimentally. By deactivating the LC or BF, we can test the effects of neuromodulation on the time it takes to learn the Tse et al. (2007) task, and discover which specific areas are applying the neuromodulation. We also expect that severing connections from the mPFC to the HPC would cause catastrophic forgetting of the tasks, as intermediate representations would not be properly

gated. It may also be possible to lesion the HPC and project artificial gating patterns on the neocortical areas storing intermediate representations to see if this prevents catastrophic forgetting.

4.5.4 Relevance to Complex Spatial Navigation

Our model builds spatial maps of the environment that incorporate contextual information. As place cells are highly sensitive to context (Smith and Mizumori, 2006), we demonstrate how associations of context and place change navigational behavior. Furthermore, experimental literature shows that the spatial selectivity of place cells decreases along the dorsal-ventral axis of the HPC (Jung et al., 1994). By extending the role of the dorsal-ventral axis as a hierarchical indexing area, we see how navigational decisions are affected by different schemas.

In addition, the use of the neuromodulatory area to increase the training of novel information matches observations that neuromodulation shapes and prioritizes replay events during consolidation (Hasselmo, 1999; Atherton et al., 2015). This consolidation is important for decision making in complex spatial navigation tasks, and is reflected in the model results. By combining the familiarity of context and novelty of index neurons in the hippocampus, the model quickly learns the locations associated with the cues for better navigation.

Chapter 5

Applying a Neurobiological Model of Schemas and Memory Consolidation to Contextual Awareness in Robotics

5.1 Introduction

We now discuss a robotic demonstration of the schemas and memory consolidation model to show how schemas impact the navigation trajectories of agents performing tasks in the real world. When operating in varied environments, robots must learn the appropriate tasks to perform within a context. This requires mental representations that are flexible enough to learn tasks in new contexts and yet stable enough to retrieve and maintain tasks in old contexts. Similarly in neuroscience, the stability-plasticity dilemma asks how the brain is plastic enough to acquire new memories quickly and yet stable enough to recall memories over

a lifetime (Mermillod et al., 2013; Abraham and Robins, 2005). Using ideas and theories from memory consolidation in neuroscience, we aim to improve contextual awareness in robotics.

One theory of how the brain balances stability and plasticity is that information is stored in schemas, which are defined as items bound together by common contexts (van Kesteren et al., 2012). Tse and colleagues demonstrated this by training rats on different schemas, which were collections of associations between different foods and their locations in a square arena (Tse et al., 2007). They found that the rats were able to learn new information quickly if it fit within a familiar schema. Additionally, the rats were able to learn new schemas without forgetting previous ones. The hippocampus (HPC) was necessary for learning schemas and any new information matching a schema, and a followup study in Tse et al. (2011) showed increased plasticity in the medial prefrontal cortex (mPFC) when information was consistent with a familiar schema. In the previous chapter, we introduced a neural network model showing how the mPFC develops representations of schemas and modulates indexing patterns in the hippocampus to form schema-specific tasks representations. With the addition of neuromodulatory areas, the model learned rapidly when information was consistent with a familiar schema.

In this chapter, we apply our model of schemas and memory consolidation to a robot task to explore its effects in an embodied setting and to test how these concepts might improve contextual awareness in real-world settings. In addition to providing insight on how the physical constraints affect the neurobiological mechanisms, our experiment demonstrates how research in memory consolidation could be applied to robotic functions.

5.2 Methods

5.2.1 Neural Network Model

The neural network model is exactly the same as presented in the previous chapter. However, the software was reimplemented in Python for better compatibility with the robot. The network parameters used in the network for this experiment are similar to the previous chapter, but with a few changes to match the different environment. The parameter values can be found in Table 5.1.

Population Sizes		Learning Parameters	
N_{cue}	18	T_s	5
N_{mPFC}	10	$\eta_{indexing}$	.1
$N_{multimodal}$	40	$\eta_{pattern}$	.0001
$N_{context}$	25	η_{CHL}	.001
N_{vHPC}	5	g	.001
N_{dHPC}	40	e_{boost}	1000
		$e_{default}$	600
		e_{settle}	20
		w_{min}	0.3
		w_{max}	0.8
		w_{inh}	-10
		w_{fam}	.0001
		$w_{novelty}$	1
		s	200
		x_{shift}	.03
		P	0.3

Table 5.1: Parameters used in robot experiment.

5.2.2 Robot Interface

We test our model by training it on the task of finding and retrieving objects in an indoor space. We use the Toyota Human Support Robot (HSR) and its associated Robot Operating System (ROS) packages to carry out the tasks (Yamamoto et al., 2018; Quigley

Figure 5.1: The Toyota Human Support Robot. A scanning lidar at the base allows for SLAM mapping of the environment. A combination of height, arm, and gripper controls allows for objects to be picked up and put down on various surfaces. An RGBD camera allows for object segmentation, identification, and localization.

et al., 2009). This includes Hector SLAM for building and using maps of the environment, the ROS navigation stack for autonomous navigation, as well as packages created by Toyota Motor Corporation to read sensor data and control the individual joints of the robot. To relay commands to the robot, we create a graphical user interface in Python, depicted in Figure 5.2. Images from the camera are input to the object segmentation and recognition package, YOLOv3 (Redmon and Farhadi, 2018), which is implemented in PyTorch (https://github.com/marvis/pytorch-yolo3). The YOLOv3 network uses pretrained weights trained on the COCO image dataset (Lin et al., 2014).

In a separate window, the neural network activity can be monitored via a separate graphical user interface. Each neuron in the context pattern layer represents an available object in the COCO dataset, and each neuron in the cue layer represents an object in the COCO dataset that is graspable by the robot. The activation of the each input neuron is set to 1 if the

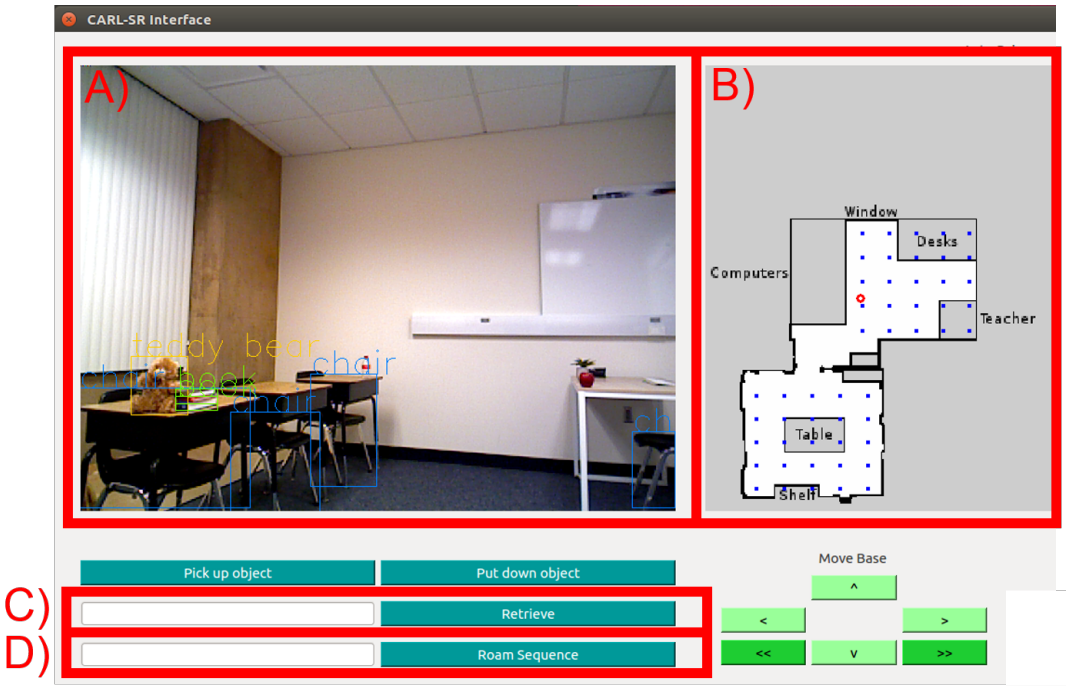

Figure 5.2: The graphical user interface for controlling the HSR includes buttons for manual control of the robot as well as the following components for automated behavior: A) The head camera input of the HSR with object segmentation and detection. B) A map of the roamable area is displayed here. The red circle shows the current location of the HSR. Blue dots represent the locations of corresponding neurons in the action layer of the neural network. C) Retrieval commands are sent to the robot by entering the name of the object to be retrieved. D) A roam sequence is initiated by clicking this button, causing the HSR to explore the current room.

object was detected within the last 60 seconds. Each neuron in the action layer represents a grid coordinate in the physical space.

For one trial of training, the robot is provided with a set of 5 physical locations in the room, which are hand-selected for maximum coverage of the area. The robot visits each of these locations in random order, stopping at each point to perform a full head rotation, scanning to identify objects. Using the infrared depth sensor on the HSR, the locations and identities of all objects seen during the scan are recorded. The locations of the objects are set to the closest grid coordinate. After visiting each location, the robot then trains on the set of objects and locations, with the set of all objects seen in the past 60 seconds input to the context pattern layer, and each graspable object seen during exploration input to the cue

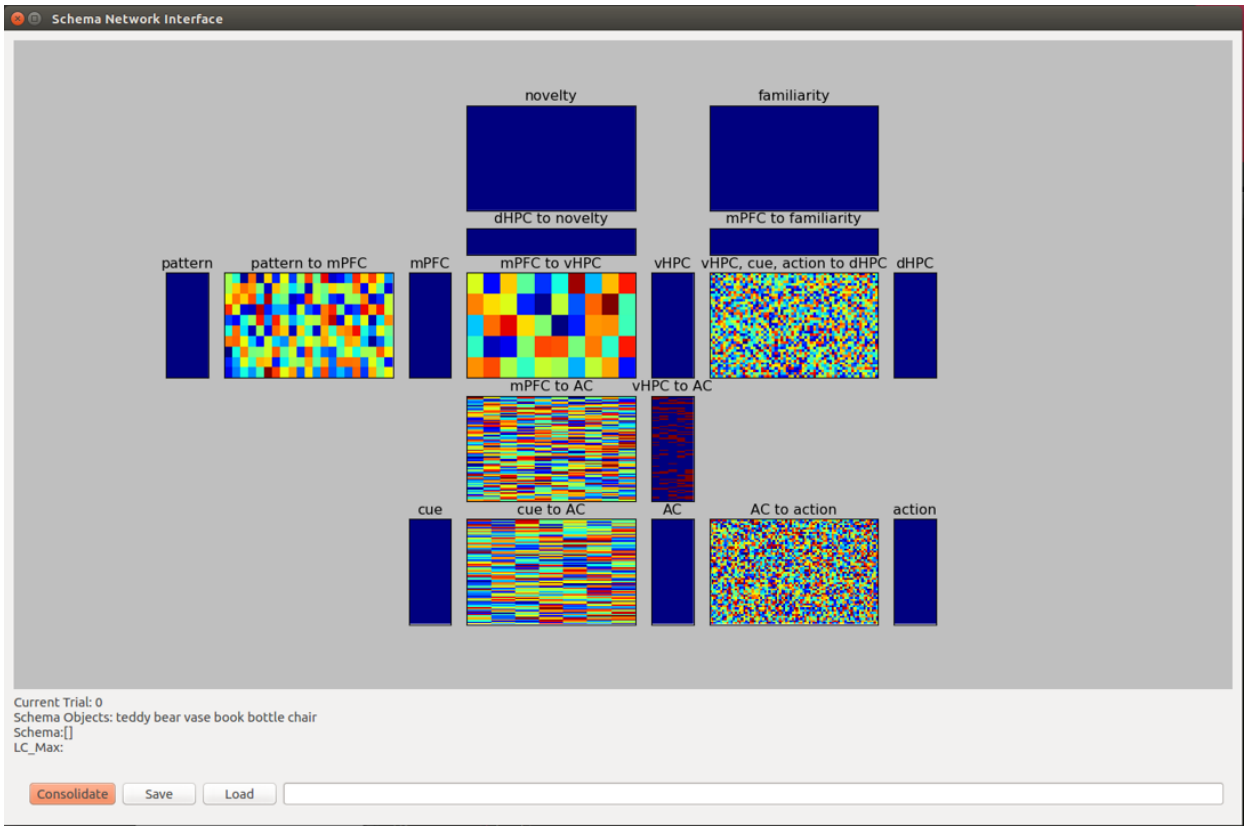

Figure 5.3: A graphical user interface for observing neural network activity. Network weights from training trials can be saved and loaded. The training of the current schema is initiated using a button. The current schema, visible contextual objects, and novelty activity are also displayed here.

layer, one per epoch.

The performance of the robot is tested by typing the name of a graspable object into the graphical user interface and requesting the robot to retrieve the item. The neural network then receives input of all objects seen in the last 60 seconds into the context pattern, and the requested object as the cue. Running the network in Test Phase, the output layer produces different levels of activation corresponding with the belief of where the cued object was located. The activities of the 5 output neurons with highest activation are normalized into a probability distribution. The robot then visits each of these 5 points by sampling from the probability distribution, removing that point from the list, and re-normalizing the probabilities. At each point, the robot performs a full head scan. If the cued object is

seen, the robot then aborts the searching process and picks up the object. The movement sequence for picking up the object is set to navigate to a specific distance and angle from the object, raising the height of the robot to match the object, lowering the arm, and closing the gripper in a pre-programmed fashion. The graspable objects in our environment are small and generally spherical, such that the same basic grasping routine is applied to all objects.

5.2.3 Experiment Design

In a series of experiments, we trained the robot to retrieve objects in two adjacent rooms, a classroom and a breakroom (Figure 5.4).

Experiment 1 - Learning and updating a single schema

The first experiment was to train the robot on a single schema (Experiment 1a). The robot was placed in a classroom setup, with typical classroom items as seen in Figures 5.4A-B. The graspable items were an apple, a bottle, and a teddy bear, placed in different locations around the room. After training and testing for 5 trials on the classroom, the teddy bear was replaced by a computer mouse (Experiment 1b). The robot then went through one trial of training, and was tested on its ability to retrieve the novel object. In all testing trials, the robot started at the same location in front of the computers, and brought the retrieved items to a neutral location near the door separating the two rooms.

Experiment 2 - Maintaining multiple schemas

The second experiment was to train the robot on a second schema consisting of a breakroom as seen in Figures 5.4C-D. The graspable items were an apple, a cup, and a wine glass.

After training and testing for 5 trials, the classroom schema was tested again to see if the robot was able to maintain performance of prior tasks. As the apple was present in both contexts, we wanted to test whether the addition of schemas would separate the tasks even with an overlap. For all testing trials, the robot started by facing the table and microwave, and brought objects to the same neutral location between the two rooms. When choosing locations to explore, only neurons corresponding to the breakroom were used.

Experiment 3 - Schema prompting

The third experiment was to test whether schemas could help the robot retrieve items it was never explicitly trained to retrieve. Starting from the neutral drop-off location, the robot was shown a banana for context, and nothing else. Then, it was cued to retrieve the banana. Since the banana was part of the breakroom schema, we predicted that the robot would search for the banana in the breakroom first as opposed to the classroom. Figure 5.5 shows the sequence of actions when retrieving the banana. We repeated the same procedure with a small graspable book in the classroom to ensure that schema prompting would work on both schemas.

5.3 Results

We completed all experiments with a population of 5 individuals. For each individual, the weights were randomly initialized at the beginning and used throughout all experiments. Figure 5.6 shows the performance in Experiments 1 and 2. Performance was calculated analytically by cueing each graspable item in the room to the network, running it in Test Phase, and calculating the percentage of activation level of the action neuron corresponding to the location of the object. Behavioral performance in terms of retrieval time was also

tested on Trial 0 (prior to any training), and Trial 4. The performance of an individual in any particular trial was averaged over each of the graspable objects in the room. In Experiment 1a, performance improved over the trials. In Experiment 1b, the retrieval time was quick despite having only one training trial. Figure 5.7 shows example trajectories taken by the robot after training in each experiment. From the starting location, the robot goes directly to the location of the item, picks it up, and returns to drop off the item at a neutral location.

In Experiment 2, performance improved over time with this new context or schema (Figure 5.6). As in Experiment 1, a novel object that fit within the context was quickly learned. Furthermore, when the individual was prompted to retrieve an object in the previously learned schema (CR in Figure 5.6), the performance was not degraded, demonstrating the ability to hold two separate schemas in memory without forgetting either one.

In Experiment 3, the robot explored the room corresponding with the schema containing the prompted object, dropping the object in the appropriate location. Figure 5.8 shows that the action layer activation is high for object locations in the same room as the prompted item. Figure 5.9 shows selected sets of weights in the network after training on all experiments. Weights from the context pattern to mPFC show two rows containing distinct patterns of high and low weight values. This indicates that there is one mPFC neuron encoding each schema. Weights from the vHPC, cue and action to dHPC show rows with 3 high values. This means that some dHPC neurons are encoding triplets of cue, action, and schema. Weights to the novelty and familiarity areas show that novelty has decreased for some dHPC inputs and familiarity has increased for the two schema neurons. Weights from mPFC to AC show two columns representing the two schema neurons, each containing a different pattern of high weight values. This shows the gating effect of the vHPC on the AC, which separates task representations by schema. More of the gating is seen in weights from AC to action, as the action neurons corresponding to the locations of items have strong weights from different

sets of AC neurons.

5.4 Discussion

Neuroscience research on schemas inspired a memory model, which allowed the Toyota Human Support Robot to separate and maintain objects by context, quickly learn the locations of novel objects, and retrieve new objects by its knowledge of a schema. This relates to current approaches in improving contextual awareness when carrying out commands (Paul et al., 2016). For instance, as the HSR is designed to aid humans in household tasks, context is important when determining how to carry out an assigned task. Maintaining schemas of different rooms allows the robot to be aware of context when carrying out tasks. When asked to retrieve items at an unknown location, the robot explored the rooms associated with the appropriate schema. This led to faster retrieval times. The same principle can be applied to other tasks, such as tidying up, seeking individuals, or behaving appropriately given a context (e.g., quiet in the classroom, raising one's voice in a noisy breakroom).

There have been various approaches to conceptual understanding of space in robotics. For instance, Kostavelis and Gasteratos (2017) combines SLAM systems and object recognition with clustering and semantic knowledge for better navigation. Hierarchical conceptual representations of space have been explored by Zender et al. (2008). In future work, we hope to combine multiple levels of abstraction in the Indexing Stream by adding multiple layers in the HPC. By representing these levels in a fully connectionist way, our model could extend to end-to-end deep learning techniques for task learning, with sensory inputs trained to motor output. With the addition of the Indexing Stream to a basic task-learning network like the Representation Stream, a deep neural network can use contextual information to maintain separate representations of tasks. This may aid in the prevention of catastrophic forgetting if the robot must learn tasks in multiple settings over time. Furthermore, the neuromodulator

can detect novelty and familiarity in the physical environment, prompting an increase in training epochs to learn new task-relevant information. By only increasing learning when a schema is familiar and a paired association is novel, the robot avoids learning irrelevant information.

The use of robotics to explore hippocampal function has revealed insights into how spatial representation in the brain arises from navigation (Gaussier et al., 2002; Barrera and Weitzenfeld, 2008). Our experiment shows how context is tied to these spatial representations via interaction with the mPFC. By studying the neural network model in an embodied setting, we learned of various sources of uncertainty that must be addressed by the brain. Initial tests showed that inaccurate object detection and depth readings led to large errors in the network. Furthermore, the model was unaware of errors in grasping, attempting to complete the retrieval sequence even if the object had not been grasped. While it is possible to decrease these sources of uncertainty, the brain still must process remaining uncertainty throughout the network. For instance, rather than having activation inputs of 1 or 0, the activations could relate to the certainty of object detection from perceptual errors. Likewise, the strength of activations in the output layer may prompt the robot to allocate more resources to motion planning when the action to be performed is uncertain.

The work presented in this chapter demonstrates how ideas from memory models in the brain may improve robotic applications and issues in artificial intelligence, such as catastrophic forgetting and lifelong learning. It also shows how these mechanisms ultimately impact the navigational decisions made by agents as they carry out tasks in their environment.

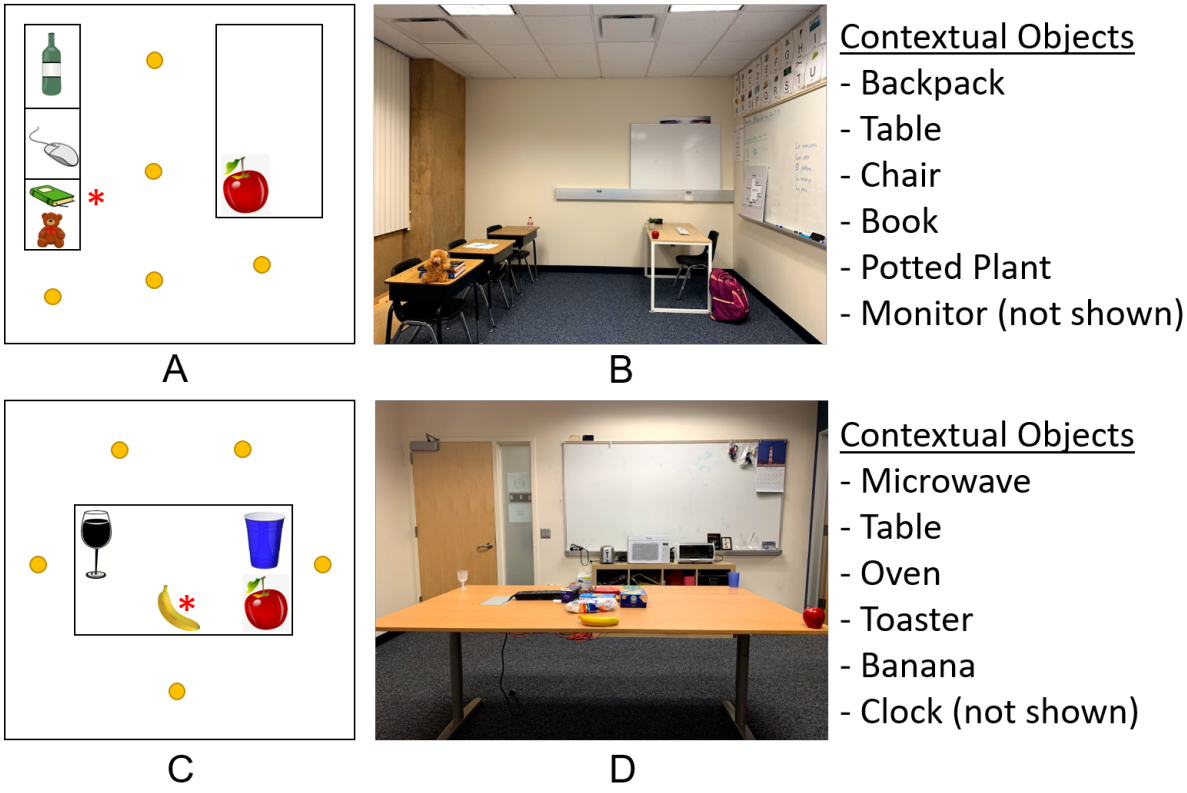

Figure 5.4: Experimental setup for the classroom and breakroom schemas. Yellow dots represent manually-picked destinations for the robot to roam and scan the room during training. A) Room layout and paired associations trained in the classroom. The bottle, teddy bear, and apple were laid out in the specified locations. After training on this layout, the teddy bear was exchanged for the mouse, introducing novelty. The robot then trained on this environment. The book (starred) was not trained as a paired association, but was included as a contextual item during training. B) Physical setup of classroom and contextual items. C) Room layout and paired associations trained in the breakroom, which is adjacent to the classroom. The wine glass, cup, and apple were laid out on the central table. The banana (starred) was not trained as a paired association, but was included as a contextual item during training. D) Physical setup of breakroom and contextual items.

Figure 5.5: Sequence of actions in Experiment 3. The HSR was shown a banana, with nothing else in its field of view. The experimenter then placed the banana on the breakroom table, outside of the HSR's view. The robot went to the breakroom to pick up the banana and navigated to the drop off location to deposit it.

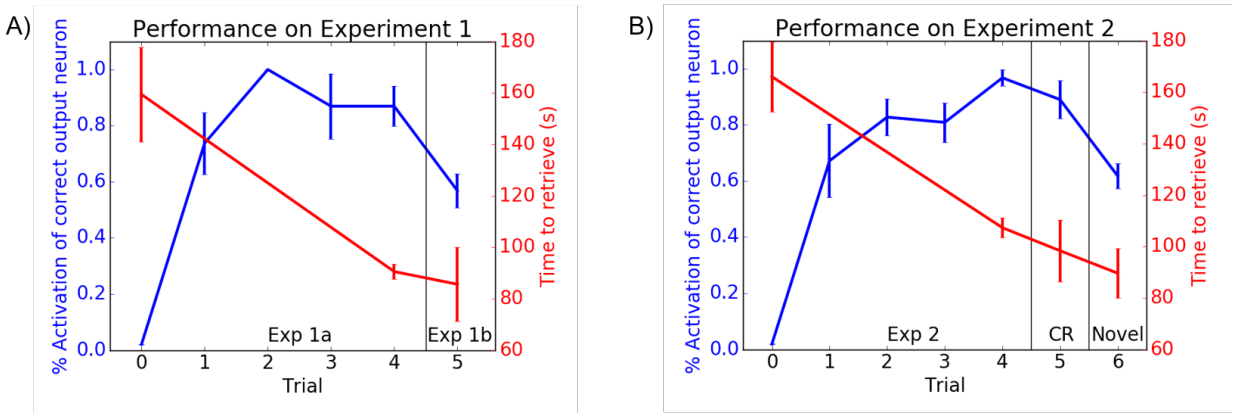

Figure 5.6: Performance on Experiments 1 and 2 on a population of n = 5. A) For Experiment 1, the activation of the correct location neuron of a cued object increased with training (blue line) and the retrieval time decreased (red line). When a novel object was introduced (Exp 1b), the location activation was still high and retrieval time was low, despite only having had one trial of training. B) Performance also improved over time when a novel schema was introduced in Experiment 2. When returning to the classroom schema (CR), performance of original objects and novel objects was retained. Points denote the mean performance and error bars denote the standard deviation of the population.

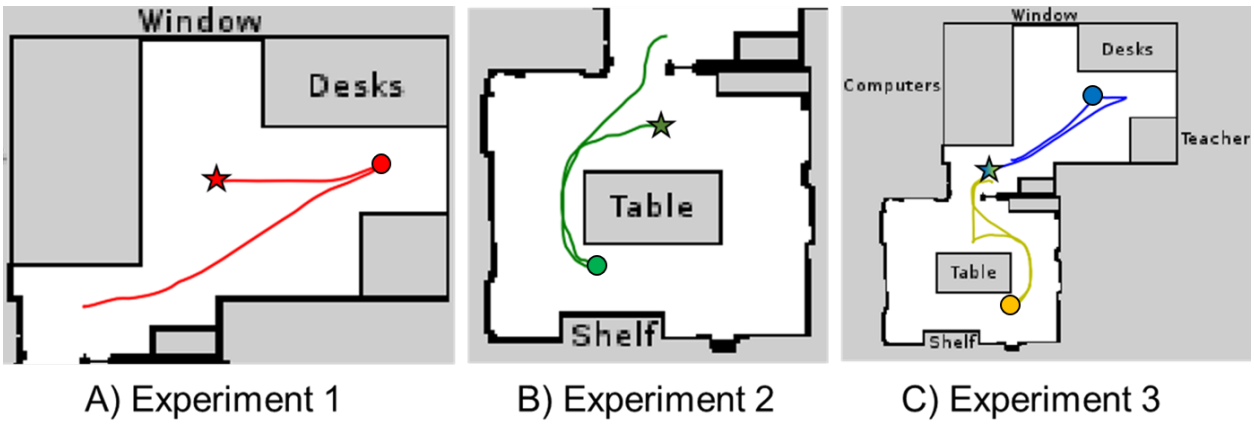

Figure 5.7: Trajectories of an individual robot at various stages of the experiments. Stars indicate the start of the trajectory, dots indicate the destination associated with the target object, and the tail end of the line indicates the final destination. A) The red line shows the trajectory of the robot when retrieving the bottle after Trial 4 of training in Experiment 1. B) The green line shows the the retrieval of an apple in Trial 4 of Experiment 2. C) The yellow line shows the trajectory of the robot retrieving the banana in Experiment 3. The blue line shows retrieval of the book. Although the entire area is accessible during retrieval of both objects, each object is associated with a prior schema, prompting the robot to search in the room corresponding to that schema.

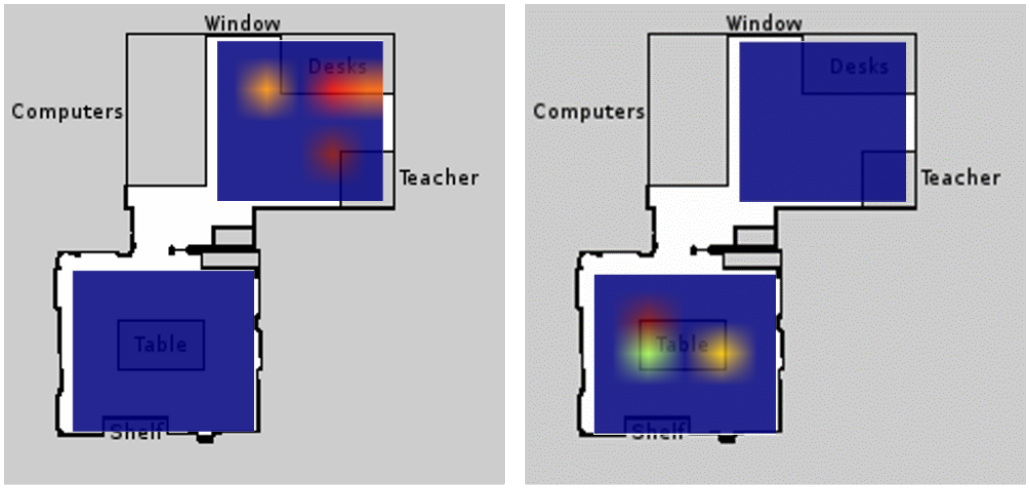

Figure 5.8: Heatmap of action layer during Experiment 3. A) When the robot was presented with a book and asked to retrieve it, the action layer showed high activity for objects in the classroom schema. B) When the robot was presented with a banana and asked to retrieve it, the action layer showed high activity for objects in the breakroom.

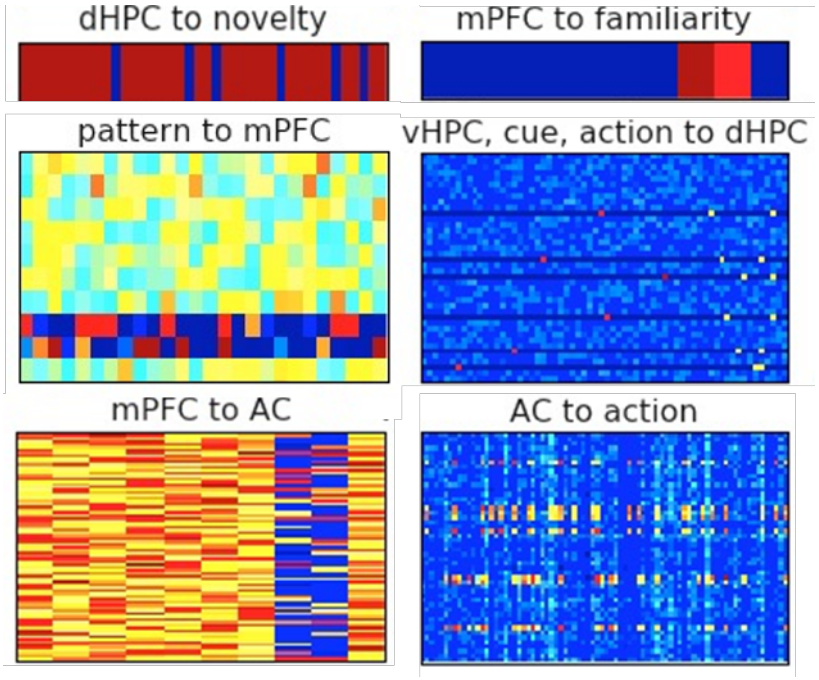

Figure 5.9: Selected weights on one individual after the experiment. Each set of weights is depicted as a heatmap, with warmer colors representing higher values, rows representing the post-synaptic layer neurons, and columns representing the pre-synaptic layer neurons.

Chapter 6

Conclusion

6.1 Future Directions

There are several ideas for expanding the presented work to scalable solutions in AI, robotics, and neuroscience research. The systems presented in Chapters 2 and 3 point towards a system of robotic navigation run entirely in neuromorphic hardware. In fact, this may even be extended to include the model of schemas and memory consolidation, as it can be implemented in a rate-coded neural network. A complete neuromorphic system for navigation could then consist of three tiers of spike-based controls. At the highest tier, tasks involving spatial navigation would be learned within multiple contexts as in Chapter 4. The mPFC would then be able to detect the current context and set environmental costs in the middle tier path planner. The action of the highest tier output could then select a destination in the middle tier. Once the chosen path is calculated, the lowest tier can run a deep spiking neural network that takes raw perception data and plans reactive controls for obstacle avoidance and path following. The energy savings of running a full neuromorphic navigation system would be especially beneficial on a mobile robotic platform.

Extending the model of schema and memory consolidation to represent schemas hierarchically would also create a flexible cognitive map based on item associations. Just as schema prompting was able to aid the retrieval of objects not explicitly trained, a hierarchical schema representation would be able to help a robot make intelligent guesses of which house, room, and location in a room an object is most likely to be located. Additionally, this would increase the scalability of the model by accessing only the lowest level of hierarchy needed to complete a navigation task. For instance, if an agent knows that an object is located in a particular room, it need not represent or retrieve information about irrelevant rooms.

Systems of navigation and memory are both dependent on similar brain areas such as the hippocampus and prefrontal cortex (Spiers, 2008; Eichenbaum and Cohen, 2014). There are also observations that memory retrieval patterns work similarly to animal foraging patterns (Abbott et al., 2015). This suggests that memory and navigation overlap in many ways. It would therefore be interesting to model both within the same network and observe which parts of the model can be shared across processes. Doing so may help us understand how higher order cognition evolved in an embodied interaction with the environment.

6.2 Summary

Navigation requires a hierarchy of skills ranging from basic to complex. This dissertation describes models and demonstrations that tackle spatial navigation along this spectrum of complexity, while simultaneously showing that the study of navigation in biological organisms can be successfully applied to navigation challenges in robotics and artificial intelligence. For more basic navigation skills such as reactionary motion planning and path planning, neuromorphic computing approaches were presented, which involved direct conversion of conventional artificial neural networks to spiking versions, as well as more tailored methods such as spike wave propagation for path planning. For higher order navigation skills such

as choosing destinations, a model of schemas and memory consolidation showed how spatial navigation tasks are learned flexibly without catastrophic forgetting. The model was then demonstrated to work on a robotic platform in the task of object retrieval. Through an interdisciplinary study of navigation, these models and demonstrations provide inspiration for improvement in industrial applications, and embodied demonstrations of navigation models shed light on how the brain helps the body move about the world.

Bibliography

Abbott, J. T., Austerweil, J. L., and Griffiths, T. L. (2015). Random walks on semantic networks can resemble optimal foraging. In *Neural Information Processing Systems Conference; A preliminary version of this work was presented at the aforementined conference.*, volume 122, page 558. American Psychological Association.

Abraham, W. C. and Robins, A. (2005). Memory retention–the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28(2):73–78.

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.-J., Taba, B., Beakes, M., Brezzo, B., Kuang, J., Manohar, R., Risk, W., Jackson, B., and Modha, D. (2015). Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557.

Aston-Jones, G. and Cohen, J. D. (2005a). Adaptive gain and the role of the locus coeruleus–norepinephrine system in optimal performance. *Journal of Comparative Neurology*, 493(1):99–110.

Aston-Jones, G. and Cohen, J. D. (2005b). An integrative theory of locus coeruleus-norepinephrine function: adaptive gain and optimal performance. *Annu Rev Neurosci*, 28:403–50.

Atherton, L. A., Dupret, D., and Mellor, J. R. (2015). Memory trace replay: the shaping of memory consolidation by neuromodulation. *Trends in neurosciences*, 38(9):560–570.

Backus, J. (1978). Can programming be liberated from the von neumann style?: a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641.

Barraquand, J., Langlois, B., and Latombe, J. C. (1992). Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224–241.

Barrera, A. and Weitzenfeld, A. (2008). Biologically-inspired robot spatial cognition based on rat neurophysiological studies. *Autonomous Robots*, 25(1-2):147–169.

Baxter, M. G. and Chiba, A. A. (1999). Cognitive functions of the basal forebrain. *Current opinion in neurobiology*, 9(2):178–183.

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90.

Benucci, A., Frazor, R. A., and Carandini, M. (2007). Standing waves and traveling waves distinguish two circuits in visual cortex. *Neuron*, 55(1):103–17.

Berridge, C. W. and Foote, S. L. (1991). Effects of locus coeruleus activation on electroencephalographic activity in neocortex and hippocampus. *Journal of Neuroscience*, 11(10):3135–3145.

Beyeler, M., Dutt, N. D., and Krichmar, J. L. (2013). Categorization and decision-making in a neurobiologically plausible spiking network using a stdp-like learning rule. *Neural Networks*, 48:109–124.

Birrell, J. M. and Brown, V. J. (2000). Medial frontal cortex mediates perceptual attentional set shifting in the rat. *Journal of Neuroscience*, 20(11):4320–4324.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66.

Chen, Z., Jacobson, A., Erdem, U. M., Hasselmo, M. E., and Milford, M. (2014). Multi-scale bio-inspired place recognition. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 1895–1901. IEEE.

Chou, T.-S., Bucci, L. D., and Krichmar, J. L. (2015). Learning touch preferences with a tactile robot using dopamine modulated stdp in a model of insular cortex. *Frontiers in Neurorobotics*, 9.

Conradt, J., Galluppi, F., and Stewart, T. C. (2015). Trainable sensorimotor mapping in a neuromorphic robot. *Robotics and Autonomous Systems*, 71:60–68.

Cox, B. R. and Krichmar, J. L. (2009). Neuromodulation as a robot controller: A brain inspired design strategy for controlling autonomous robots. *IEEE Robotics & Automation Magazine*, 16(3):72–80.

Cruz-Albrecht, J. M., Yung, M. W., and Srinivasa, N. (2012). Energy-efficient neuron, synapse and stdp integrated circuits. *Biomedical Circuits and Systems, IEEE Transactions on*, 6(3):246–256.

Danowitz, A., Kelley, K., Mao, J., Stevenson, J. P., and Horowitz, M. (2012). Cpu db: recording microprocessor history. *Communications of the ACM*, 55(4):55–63.

Detorakis, G., Bartley, T., and Neftci, E. (2018). Contrastive hebbian learning with random feedback weights. *arXiv preprint arXiv:1806.07406*.

Dragoi, G. and Tonegawa, S. (2011). Preplay of future place cell sequences by hippocampal cellular assemblies. *Nature*, 469(7330):397–401.

Dragoi, G. and Tonegawa, S. (2013). Distinct preplay of multiple novel spatial experiences in the rat. *Proc Natl Acad Sci U S A*, 110(22):9100–5.

Eichenbaum, H. (2017). Prefrontal–hippocampal interactions in episodic memory. *Nature Reviews Neuroscience*, 18(9):547.

Eichenbaum, H. and Cohen, N. J. (2014). Can we reconcile the declarative memory and spatial navigation views on hippocampal function? *Neuron*, 83(4):764–770.

Eiter, T. and Mannila, H. (1994). Computing discrete fréchet distance. Technical report, Citeseer.

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., Berg, D. J., McKinstry, J. L., Melano, T., Barch, D. R., di Nolfo, C., Datta, P., Amir, A., Taba, B., Flickner, M. D., and Modha, D. S. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446.

Fields, R. D. (2008). White matter in learning, cognition and psychiatric disorders. *Trends Neurosci*, 31(7):361–70.

Fields, R. D. (2015). A new mechanism of nervous system plasticity: activity-dependent myelination. *Nat Rev Neurosci*, 16(12):756–767.

Fischl, K. D., Fair, K., Tsai, W.-Y., Sampson, J., and Andreou, A. (2017). Path planning on the truenorth neurosynaptic system. In *2017 IEEE International Symposium on Circuits and Systems*. IEEE.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Gallistel, C. (1993). *The Organization of Learning*. MIT Pres.

Galluppi, F., Denk, C., Meiner, M. C., Stewart, T. C., Plana, L. A., Eliasmith, C., Furber, S., and Conradt, J. (2014). Event-based neural computing on an autonomous mobile platform. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2862–2867. IEEE.

Gaussier, P., Revel, A., Banquet, J.-P., and Babeau, V. (2002). From view cells and place cells to cognitive map learning: processing stages of the hippocampal system. *Biological cybernetics*, 86(1):15–28.

Han, F., Caporale, N., and Dan, Y. (2008). Reverberation of recent visual experience in spontaneous cortical waves. *Neuron*, 60(2):321–7.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107.

Hasselmo, M. E. (1999). Neuromodulation: acetylcholine and memory consolidation. *Trends in cognitive sciences*, 3(9):351–359.

Hawkins, J., Ahmad, S., and Cui, Y. (2017). A theory of how columns in the neocortex enable learning the structure of the world. *Frontiers in neural circuits*, 11:81.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., et al. (2015). An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*.

Hwu, T., Isbell, J., Oros, N., and Krichmar, J. (2016). A self-driving robot using deep convolutional neural networks on neuromorphic hardware. *arXiv preprint arXiv:1611.01235v1*.

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., et al. (2011). Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5:73.

Izhikevich, E. M. (2006). Polychronization: computation with spikes. *Neural Comput*, 18(2):245–82.

Izhikevich, E. M. and Edelman, G. M. (2008). Large-scale model of mammalian thalamocortical systems. *Proc Natl Acad Sci U S A*, 105(9):3593–8.

Izhikevich, E. M., Gally, J. A., and Edelman, G. M. (2004). Spike-timing dynamics of neuronal groups. *Cereb Cortex*, 14(8):933–44.

Jung, M. W., Wiener, S. I., and McNaughton, B. L. (1994). Comparison of spatial firing characteristics of units in dorsal and ventral hippocampus of the rat. *Journal of Neuroscience*, 14(12):7347–7356.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835.

Kostavelis, I. and Gasteratos, A. (2017). Semantic maps from multiple visual cues. *Expert Systems with Applications*, 68:45–57.

Koul, S. and Horiuchi, T. K. (2015). Path planning by spike propagation. In *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, pages 1–4.

Koziol, S., Brink, S., and Hasler, J. (2013). Path planning using a neuron array integrated circuit. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 663–666.

Koziol, S., Brink, S., and Hasler, J. (2014). A neuromorphic approach to path planning using a reconfigurable neuron array ic. *IEEE Trans. VLSI Syst.*, 22:2724–2737.

Koziol, S., Hasler, P., and Stilman, M. (2012). Robot path planning using field programmable analog arrays. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1747–1752.

Krichmar, J. (2016). Path planning using a spiking neuron algorithm with axonal delays. In *IEEE Congress on Evolutionary Computation*, pages 1219–1226.

Krichmar, J. L. (2008). The neuromodulatory system: a framework for survival and adaptive behavior in a challenging world. *Adaptive Behavior*, 16(6):385–399.

Krichmar, J. L., Coussy, P., and Dutt, N. (2015). Large-scale spiking neural networks using neuromorphic hardware compatible models. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 11(4):36.

Kumaran, D., Hassabis, D., and McClelland, J. L. (2016). What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534.

LaValle, S. M. (2011a). Motion planning. *Robotics & Automation Magazine, IEEE*, 18(1):79–89.

LaValle, S. M. (2011b). Motion planning. *Robotics & Automation Magazine, IEEE*, 18(2):108–118.

LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *arXiv preprint arXiv:1608.08782*.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., et al. (2011). Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Lubenov, E. V. and Siapas, A. G. (2009). Hippocampal theta oscillations are travelling waves. *Nature*, 459(7246):534–9.

Masse, N. Y., Grant, G. D., and Freedman, D. J. (2018). Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *arXiv preprint arXiv:1802.01569*.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

McKinstry, J. L., Edelman, G. M., and Krichmar, J. L. (2006). A cerebellar model for predictive motor control tested in a brain-based device. *Proceedings of the National Academy of Sciences of the United States of America*, 103(9):3387–3392.

Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636.

Mermillod, M., Bugaiska, A., and Bonin, P. (2013). The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673.

Movellan, J. R. (1991). Contrastive Hebbian learning in the continuous hopfield model. In *Connectionist Models*, pages 10–17. Elsevier.

Nakano, S. and Hattori, M. (2017). Reduction of catastrophic forgetting in multilayer neural networks trained by contrastive Hebbian learning with pseudorehearsal. In *Computational Intelligence and Applications (IWCIA), 2017 IEEE 10th International Workshop on*, pages 91–95. IEEE.

Ni, J., Wu, L., Fan, X., and Yang, S. X. (2016). Bioinspired intelligent algorithm and its applications for mobile robot control: a survey. *Computational intelligence and neuroscience*, 2016:1.

Ni, J. and Yang, S. X. (2011). Bioinspired neural network for real-time cooperative hunting by multirobots in unknown environments. *IEEE Transactions on Neural Networks*, 22(12):2062–2077.

O'Keefe, J. and Nadel, L. (1978). *The Hippocampus as a Cognitive Map*. Oxford University Press.

Oros, N. and Krichmar, J. L. (2013). Smartphone based robotics: Powerful, flexible and inexpensive robots for hobbyists, educators, students and researchers. Technical Report 13-16, Center for Embedded Computer Systems, University of California, Irvine, Irvine, California.

Otmakhova, N., Duzel, E., Deutch, A. Y., and Lisman, J. (2013). The hippocampal-vta loop: the role of novelty and motivation in controlling the entry of information into long-term memory. In *Intrinsically motivated learning in natural and artificial systems*, pages 235–254. Springer.

Ovtcharov, K., Ruwase, O., Kim, J.-Y., Fowers, J., Strauss, K., and Chung, E. S. (2015). Accelerating deep convolutional neural networks using specialized hardware. *Microsoft Research Whitepaper*, 2(11).

Palmer, J. H. and Gong, P. (2014). Associative learning of classical conditioning as an emergent property of spatially extended spiking neural circuits with synaptic plasticity. *Front Comput Neurosci*, 8:79.

Paul, R., Arkin, J., Roy, N., and M Howard, T. (2016). Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators. *The International Journal of Robotics Research*.

Pfeiffer, B. E. and Foster, D. J. (2013). Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497(7447):74–9.

Pfeiffer, B. E. and Foster, D. J. (2015). Place cells. autoassociative dynamics in the generation of sequences of hippocampal place cells. *Science*, 349(6244):180–3.

Pomerleau, D. A. (1989). Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313.

Ponulak, F. and Hopfield, J. (2013). Rapid, parallel path planning by propagating wavefronts of spiking neural activity. *Frontiers in Computational Neuroscience*, 7(98).

Preston, A. R. and Eichenbaum, H. (2013). Interplay of hippocampus and prefrontal cortex in memory. *Current Biology*, 23(17):R764–R773.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan.

Quoy, M., Laroque, P., and Gaussier, P. (2002). Learning and motivational couplings promote smarter behaviors of an animat in an unknown world. *Robotics and Autonomous Systems*, 38(3-4):149–156.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*.

Rubino, D., Robbins, K. A., and Hatsopoulos, N. G. (2006). Propagating waves mediate information transfer in the motor cortex. *Nat Neurosci*, 9(12):1549–57.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Samsonovich, A. V. and Ascoli, G. A. (2005). A simple neural network model of the hippocampus suggesting its pathfinding role in episodic memory retrieval. *Learn Mem*, 12(2):193–208.

Sato, T. K., Nauhaus, I., and Carandini, M. (2012). Traveling waves in visual cortex. *Neuron*, 75(2):218–29.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275(5306):1593–9.

Shim, V. A., Ranjit, C. S. N., Tian, B., Yuan, M., and Tang, H. (2015). A simplified cerebellar model with priority-based delayed eligibility trace learning for motor control. *IEEE Transactions on Autonomous Mental Development*, 7(1):26–38.

Silva, D., Feng, T., and Foster, D. J. (2015). Trajectory events across hippocampal place cells require previous experience. *Nat Neurosci*, 18(12):1772–9.

Smith, D. M. and Mizumori, S. J. (2006). Hippocampal place cells, context, and episodic memory. *Hippocampus*, 16(9):716–729.

Soltoggio, A., Stanley, K. O., and Risi, S. (2017). Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *arXiv preprint arXiv:1703.10371*.

Soulignac, M. (2011). Feasible and optimal path planning in strong current fields. *Robotics, IEEE Transactions on*, 27(1):89–98.

Spiers, H. J. (2008). Keeping the goal in mind: Prefrontal contributions to spatial navigation. *Neuropsychologia*, 46(7):2106.

Stentz, A. and Carnegle, I. (1993). Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, 10:89–100.

Teyler, T. J. and DiScenna, P. (1986). The hippocampal memory indexing theory. *Behavioral neuroscience*, 100(2):147.

Thorpe, S., Delorme, A., and Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural Netw*, 14(6-7):715–25.

Thrun, S. (2010). Toward robotic cars. *Communications of the ACM*, 53(4):99–106.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208.

Tse, D., Langston, R. F., Kakeyama, M., Bethus, I., Spooner, P. A., Wood, E. R., Witter, M. P., and Morris, R. G. (2007). Schemas and memory consolidation. *Science*, 316(5821):76–82.

Tse, D., Takeuchi, T., Kakeyama, M., Kajii, Y., Okuno, H., Tohyama, C., Bito, H., and Morris, R. G. (2011). Schema-dependent gene activation and memory encoding in neocortex. *Science*, 333(6044):891–895.

Van Kesteren, M. T., Beul, S. F., Takashima, A., Henson, R. N., Ruiter, D. J., and Fernández, G. (2013). Differential roles for medial prefrontal and medial temporal cortices in schema-dependent encoding: from congruent to incongruent. *Neuropsychologia*, 51(12):2352–2359.

van Kesteren, M. T., Ruiter, D. J., Fernández, G., and Henson, R. N. (2012). How schema and novelty augment memory formation. *Trends in neurosciences*, 35(4):211–219.

VanRullen, R., Guyonneau, R., and Thorpe, S. J. (2005). Spike times make sense. *Trends Neurosci*, 28(1):1–4.

Wagatsuma, A., Okuyama, T., Sun, C., Smith, L. M., Abe, K., and Tonegawa, S. (2018). Locus coeruleus input to hippocampal CA3 drives single-trial learning of a novel context. *Proceedings of the National Academy of Sciences*, 115(2):E310–E316.

Walling, S. G., Brown, R. A., Milway, J. S., Earle, A. G., and Harley, C. W. (2011). Selective tuning of hippocampal oscillations by phasic locus coeruleus activation in awake male rats. *Hippocampus*, 21(11):1250–1262.

Wang, R., Cohen, G., Stiefel, K. M., Hamilton, T. J., Tapson, J., and van Schaik, A. (2013). An fpga implementation of a polychronous spiking neural network with delay adaptation. *Front Neurosci*, 7:14.

Wang, R. M., Hamilton, T. J., Tapson, J. C., and van Schaik, A. (2014). A mixed-signal implementation of a polychronous spiking neural network with delay adaptation. *Front Neurosci*, 8:51.

Wang, R. M., Hamilton, T. J., Tapson, J. C., and van Schaik, A. (2015). A neuromorphic implementation of multiple spike-timing synaptic plasticity rules for large-scale neural networks. *Front Neurosci*, 9:180.

Wu, J. Y., Xiaoying, H., and Chuan, Z. (2008). Propagating waves of activity in the neocortex: what they are, what they do. *Neuroscientist*, 14(5):487–502.

Yamamoto, T., Nishino, T., Kajima, H., Ohta, M., and Ikeda, K. (2018). Human support robot (hsr). In *ACM SIGGRAPH 2018 Emerging Technologies*, page 11. ACM.

Yu, A. and Dayan, P. (2005). Uncertainty, neuromodulation, and attention. *Neuron*, 46(4):681–692.

Zanos, T. P., Mineault, P. J., Nasiotis, K. T., Guitton, D., and Pack, C. C. (2015). A sensorimotor role for traveling waves in primate visual cortex. *Neuron*, 85(3):615–27.

Zeiler, M. D. and Fergus, R. (2014). *Visualizing and Understanding Convolutional Networks*, pages 818–833. Springer International Publishing, Cham.

Zender, H., Mozos, O. M., Jensfelt, P., Kruijff, G.-J., and Burgard, W. (2008). Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502.

Zhou, Y. and Zeng, J. (2015). Massively parallel a* search on a gpu. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.