



12-2016

Neurally Plausible Model of Robot Reaching Inspired by Infant Motor Babbling

Zahra Mahoor

University of Tennessee, Knoxville, zmahoor@vols.utk.edu

Recommended Citation

Mahoor, Zahra, "Neurally Plausible Model of Robot Reaching Inspired by Infant Motor Babbling. " PhD diss., University of Tennessee, 2016.

https://trace.tennessee.edu/utk_graddiss/4149

This Dissertation is brought to you for free and open access by the Graduate School at Trace: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of Trace: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a dissertation written by Zahra Mahoor entitled "Neurally Plausible Model of Robot Reaching Inspired by Infant Motor Babbling." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Bruce MacLennan, Major Professor

We have read this dissertation and recommend its acceptance:

Daniela Corbetta, Lynne Parker, James Plank

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Neurally Plausible Model of Robot Reaching Inspired by Infant Motor Babbling

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Zahra Mahoor

December 2016

© by Zahra Mahoor, 2016
All Rights Reserved.

To funny Lonnie and fluffy Ling Ling for their love and support.

Acknowledgements

First of all, I would like to thank my Ph.D. advisor and mentor, Dr. Bruce MacLennan for his support, guidance, and patience throughout the course of my study. This dissertation would not have been completed without his invaluable teachings and inputs during our weekly meetings. I appreciate that he would always answer my questions and correct my mistakes without judgment.

Also, I would like to thank the members of my Ph.D. committee, Dr. Daniela Corbetta, Dr. James Plank and Dr. Lynne Parker, for their valuable and constructive feedback on this dissertation. In particular, I must thank Dr. Parker for allowing me to participate in her weekly group meetings. Her group meetings broadened my knowledge of artificial intelligence and taught me critical thinking. Additionally, I would like to thank the Department of Electrical Engineering and Computer Science at the University of Tennessee for providing me with a teaching assistant position and the National Science Foundation for funding the humanoid robot used in my experiments.

I would like to thank my wonderful friends at the University of Tennessee who have helped me on both personal and professional levels: Nastaran Simarasl, Sarah Mousavi, Misagh and Amanda Mansouri, Shima Mohebbi, Casey Miller, Allen McBride, Katie Schuman, Meg Drouhard, Nicole Pennington, Sadika Amreen, Denise Gosnell, Nicholas Lineback, Stephanie Rivera, Kathy Thompson, Melanie Reese, Dana Bryson, and the Syssters at the University of Tennessee.

I would like to thank my family for their love and support, especially my mother who lives on the other side of the world. Her encouraging voice telling me to pursue my dreams is always with me. Most of all I would like to thank my loving, supportive, encouraging, and funny husband, Lonnie Yu, who supported me in every possible way while I was writing this dissertation. Finally, I would like to thank my beautiful, fluffy cat, Ling Ling, whose presence in my life provides me with an enormous amount of hope and joy.

Whoever wants to reach a distant goal must take small steps.—Saul Bellow

Abstract

In this dissertation, we present an abstract model of infant reaching that is neurally-plausible. This model is grounded in embodied artificial intelligence, which emphasizes the importance of the sensorimotor interaction of an agent and the world. It includes both learning sensorimotor correlations through motor babbling and also arm motion planning using spreading activation. We introduce a mechanism called bundle formation as a way to generalize motions during the motor babbling stage.

We then offer a neural model for the abstract model, which is composed of three layers of neural maps with parallel structures representing the same sensorimotor space. The motor babbling period shapes the structure of the three neural maps as well as the connections within and between them; these connections encode trajectory bundles in the neural maps.

We then investigate an implementation of the neural model using a reaching task on a humanoid robot. Through a set of experiments, we were able to find the best way to implement different components of this model such as motor babbling, neural representation of sensorimotor space, dimension reduction, path planning, and path execution.

After the proper implementation had been found, we conducted another set of experiments to analyze the model and evaluate the planned motions. We evaluated unseen reaching motions using jerk, end effector error, and overshooting. In these experiments, we studied the effect of different dimensionalities of the reduced

sensorimotor space, different bundle widths, and different bundle structures on the quality of arm motions.

We hypothesized a larger bundle width would allow the model to generalize better. The results confirmed that the larger bundles lead to a smaller error of end-effector position for testing targets. An experiment with the resolution of neural maps showed that a neural map with a coarse resolution produces less smooth motions compared to a neural map with a fine resolution. We also compared the unseen reaching motions under different dimensionalities of the reduced sensorimotor space. The results showed that a smaller dimension leads to less smooth and accurate movements.

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Approach and Uniqueness	1
1.3	Contributions	4
2	Background in Infant Development	6
2.1	Introduction	6
2.2	Developmental Trends	6
2.2.1	Rhythmical Stereotypies	6
2.2.2	Mass to Specific Movement	7
2.2.3	Hierarchical Integration	7
2.2.4	Developmental Direction	7
2.3	Reflexes	8
2.3.1	Palmar Grasping Reflex	9
2.3.2	Influence of Reflexes on Voluntary Movement	9
2.4	Manual Control	10
2.4.1	Pre-reaching	10
2.4.2	Goal-Directed Reaching	11
2.4.3	Grasping	11
2.4.4	Releasing	12
2.5	Brain Development	13

3	Literature Review on Developmental Robotics	14
4	Methodology and Implementation	31
4.1	Conceptual Model	31
4.1.1	Motor-sensory Phase-space and Trajectory Bundles	31
4.1.2	Conceptual Trajectory Bundles Formation	32
4.1.3	Conceptual Trajectory Planning	33
4.1.4	Conceptual Trajectory Execution	36
4.2	Neural Model	37
4.2.1	Formation of Neurally-Plausible Trajectory Bundles	38
4.2.2	Neurally-Plausible Trajectory Planning and Execution	41
4.3	Model Implementation	42
4.3.1	Implementation of Trajectory Bundle Formation	44
4.3.2	Implementation of Path Planning and Execution	44
4.3.3	Dimension Reduction	45
4.3.4	Representation of Motor-sensory Space	49
5	Experiments and Results	63
5.1	Humanoid Robot	64
5.2	Motor Babbling	65
5.3	Random Start Points and End Points	66
5.3.1	Dimension Reduction	70
5.3.2	Self-Organizing Maps	74
5.3.3	Manually Constructing a Neural Map	78
5.4	Evaluation Metrics of Planned Motions	84
5.5	Fixed Start Point	86
5.5.1	Autoencoder as Dimension Reduction	88
5.5.2	Construction of the Neural Map using Cartesian Product of Features	89
5.5.3	Results of Varying Dimension of Reduced Space	90

5.5.4	Results of Varying Bundle Width	94
5.5.5	Results of Varying Training Size	98
5.5.6	Diffusion-Based Path Planning versus Breadth First Search . .	102
5.5.7	Initial Bundle Formation	102
5.6	Multiple Fixed Start Points	106
5.6.1	Results of Varying Bundle Width	107
5.6.2	Results of Modifying Bundle Formation	109
5.6.3	Results of Varying the Resolution of the Neural Map	114
5.7	Discussion	117
6	Conclusion	124
6.1	Future Research	127
	Bibliography	130
	Vita	140

List of Tables

2.1	Developmental sequence for the appearance and inhibition of selected primitive reflex behaviors (Gallahue, 1982)	9
3.1	Components of Development adapted from (Lungarella et al., 2003). .	16
5.1	Range of joint positions of the left arm.	66
5.2	Motor babbling strategies.	67
5.3	Sensor values.	68
5.4	Motor values.	68
5.5	Motors and sensors during the motor babbling creates a motor-sensory trajectory.	69
5.6	Sequence of neurons fired for two sample trajectories in a 200×300 neural map.	76
5.7	Sequence of neurons fired for sample trajectories in a 350×400 neural map.	78
5.8	Left arm's configuration of the fixed start.	86
5.9	Path Planning Parameters.	88
5.10	Accuracy of autoencoders with different bottleneck size.	92
5.11	Accuracy of autoencoders with different training size.	101
5.12	Left arm's configuration for the starts.	107

List of Figures

2.1	Examples of spontaneous (stereotypic) movements of the arms, hands, and fingers: (a) arm waving with object, (b) arm banging against a surface, (c) banging both arms together with object, (d) arm sway, (e) and (f) finger flexion, (g) hand rotation, (h) hand flexion (Gabbard, 2004).	8
2.2	(a) Basic grasping technique and (b) Changes in object positioning (Gabbard, 2004).	12
3.1	Partial motor development sequence for the iCub humanoid robot. Shaded regions relate to periods of development of each ability as observed in infants. Darker shading indicates more advanced abilities (Law et al., 2011a).	22
3.2	Partial vision development sequence for the iCub robot (Law et al., 2011a).	22
3.3	The (a) reaching and (b) grasping architecture (Caligiore et al., 2008).	23
3.4	The spiking neural network framework during training and test (Bouganis and Shanahan, 2010).	26
3.5	The NOCH framework (DeWolf and Eliasmith, 2011).	29
4.1	A conceptual bundle with three trajectories in which ϕ represents the width of trajectories to fuzzy the dynamic.	35

4.2	An overview of our reaching model with three neural maps at its core and dimension reduction and dimension expansion modules.	38
4.3	Forward and backward connections between neurons i and j	40
4.4	A simplified neural representation of a trajectory bundle with gray nodes inside the bundle and color of the edges representing strength of the connections among the neurons in the bundle.	40
4.5	Neural architecture for implementing the path planning and execution process. Activity in map B spreads from the goal state γ , and the current state r in map F excites nearby neurons r' and r'' in map C . Competition among excited neurons in map C leads to firing of neuron r' , which represents the new motor-sensory state.	43
4.6	A two-layer neural network is in the core of the GHA algorithm. In this example, the output layer that captures the principal components can have up to 4 nodes. The output nodes with dashed line could be the two least significant components.	47
4.7	A general framework of an autoencoder. The green arrows and orange arrows respectively represent the weights of encoder and the decoder. The smallest box in the middle represents the bottleneck layer.	49
4.8	a) A small self-organizing map where each node of input is connected to all the nodes of the map. One node is colored in brown as an example along with its neighborhood in different colors. Connections are omitted to avoid clutter in this image. b) Neighborhoods of size of 1 and 2 are marked dash lines for a center node located in $(2,2)$ in a hexagonal topology. Six nodes with indices of $(1,1), (1,2), (2,1), (2,3), (3,2), (3,3)$ are in distance 1 of this node. Twelve nodes with indices of $(0,0), (0,1), (0,2), (1,0), (1,3), (2,0), (2,4), (3,1), (3,4), (4,2), (4,3), (4,4)$ are within distance 2 of this node. . . .	52

4.9	Creating a 3D neural map using Cartesian product of the three features of space \mathcal{A}' . Each side of the cube represents one of the three features that are arranged from min_i to max_i with step size of res_i	55
4.10	Creating a 2D neural map by arranging the first two dimensions of the space \mathcal{A}' . Each patch or small square is divided according to the samples within the boundaries of that square. s_i represents a sample of the training set.	56
5.1	Rosie, Humanoid Robot.	64
5.2	Random trajectory babbling inside a box (safe area) in front of Rosie.	68
5.3	(a) The first and (b) the second eigenvalues resulting from PCA and GHA.	71
5.4	(a) The y axis shows the difference between the (a) first and (b) second eigenvectors calculated by PCA and the ones calculated by GHA.	72
5.5	Variance versus number of eigenvalues after applying PCA to (a) a training set of 5,000 babbling trajectories with 41 features and (b) a training set of 10,000 babbling trajectories with 41 features.	73
5.6	Variance versus number of eigenvalues after applying PCA to (a) a training set of 5,000 babbling trajectories with 20 features and (b) a training set of 10,000 babbling trajectories with 20 features.	74
5.7	Variance versus number of eigenvalues after applying PCA to (a) a training set of 5,000 babbling trajectories with 10 features and (b) a training set of 10,000 babbling trajectories with 10 features.	75
5.8	(a) A 2D histogram for neural map of 200×300 neurons (a) for the entire babbling data set and (b) for one babbling trajectory.	77
5.9	A 2D histogram for a 150×150 neural map.	79

5.10	(a) A sequence of the (a) first and (b) second joint of the shoulder's positions where various seen arm trajectories are chained one after another. The green line represents the original seen trajectory and the blue line represents the same trajectory after the process of $data \rightarrow PCA \rightarrow PCA^{-1} \rightarrow reconstructed_data$	81
5.11	(a) A sequence of the (a) first and (b) second joint of the shoulder's positions where various seen arm trajectories are chained one after another. The green line represents the original seen trajectory and the blue line represents the same trajectory after the process of $data \rightarrow PCA \rightarrow neural_map \rightarrow PCA^{-1} \rightarrow reconstructed_data$	82
5.12	(a) A sequence of the (a) first and (b) second joint of the shoulder's positions where various unseen arm trajectories are chained one after another. The green line represents the original unseen trajectory and the blue line represents the same trajectory after the process of $data \rightarrow PCA \rightarrow neural_map \rightarrow PCA^{-1} \rightarrow reconstructed_data$	83
5.13	Experimental settings: (a) Meka Robotics M3 mobile humanoid robot "Rosie." (b) The humanoid robot randomly explores an arc in front of its left arm with a fixed start pose. The initial and final positions of babbling trajectories are on this arc, but there is no constraint on points between the initial and final positions.	87
5.14	A babbling trajectory along with its reconstruction using " $data \rightarrow encoder \rightarrow decoder \rightarrow reconstructed_data$ ". The solid and dashed lines represent the reconstructed trajectories and the original ones respectively.	89
5.15	(a) A sequence of the (a) first and (b) second joint of the shoulder's positions where various unseen trajectories are chained one after another, mapping the original ones and after the reconstruction by " $data \rightarrow encoder \rightarrow neural_map \rightarrow decoder \rightarrow reconstructed_data$ ". . .	91

5.16	Evaluation of planned motions for three trials, where $ \phi = 1$ is for the trials and the training data is 70%: (1) $ \mathcal{A}' = 3$, (2) $ \mathcal{A}' = 4$, and (3) $ \mathcal{A}' = 5$. (a) Smoothness of planned trajectories based on norm jerk metric across three trials. (b) Euclidean distance error of end effector position across three trials. (c) Smoothness of planned trajectories based on norm jerk metric across the y value of the final position in robot's coordinate system. (d) Mean Euclidean distance error of end effector position across the y value of the final position in robot's coordinate system.	95
5.17	Evaluation of planned motions for three trials, where $ \phi = 1$ is for the trials and the training data is 70%: (1) $ \mathcal{A}' = 3$, (2) $ \mathcal{A}' = 4$, and (3) $ \mathcal{A}' = 5$. (a) Smoothness of planned trajectories based on overshooting index across three trials. (b) Smoothness of planned trajectories based on overshooting index across y value of the final position robot's coordinate system.	96
5.18	Evaluation of planned motions for four trials, where $ \mathcal{A}' = 5$ is for the trials and the training data is 70%: (1) $ \phi = 1$, (2) $ \phi = 3$, (3) $ \phi = 6$, and $ \phi = 10$. (a) Smoothness of planned trajectories based on norm jerk metric across four trials. (b) Euclidean distance error of end effector position across four trials. (c) Smoothness of planned trajectories based on norm jerk metric versus y value of the final position the robot's coordinate system. (d) Mean Euclidean distance error of end effector position across the y values of the final position.	99
5.19	Evaluation of planned motions for four trials, where $ \mathcal{A}' = 5$ is for the trials and the training data is 70%: (1) $ \phi = 1$, (2) $ \phi = 3$, (3) $ \phi = 6$, and $ \phi = 10$. (a) Smoothness of planned trajectories based on overshooting index across three trials. (b) Smoothness of planned trajectories based on overshooting index across y axis of robot's coordinate system.	100

5.20	Evaluation of planned motions for five trials, where $ \phi = 1$ and $ \mathcal{A}' = 5$ are for the five trials: (1) $train = 10\%$, (2) $train = 20\%$, (3) $train = 30\%$, (4) $train = 50\%$, and (5) $train = 70\%$. (a) Smoothness of planned trajectories based on norm jerk metric across five trials. (b) Euclidean distance error of end effector position across five trials. (c) Smoothness of planned trajectories based on norm jerk metric across y value of final position of the arm in robot's coordinate system. (d) Mean Euclidean distance error of end effector position across y value of the final position.	103
5.21	Evaluation of planned motions for five trials, where $ \phi = 1$ and $ \mathcal{A}' = 5$ are for the five trials: (1) $train = 10\%$, (2) $train = 20\%$, (3) $train = 30\%$, (4) $train = 50\%$, and (5) $train = 70\%$. (a) Smoothness of planned trajectories based on overshooting index across five trials. (b) Smoothness of planned trajectories based on overshooting index across y value of final position of the arm in robot's coordinate system. . . .	104
5.22	(a) The dashed and solid lines are planned joint positions and velocities using breadth first search and the spreading activity and (b) a planned trajectory where the bundle formation stage was combined with Alg. 2.	105
5.23	Experimental settings: (a) Meka Robotics M3 mobile humanoid robot "Rosie." (b) The humanoid robot randomly explores an arc in front of its left arm from 8 different start poses. The initial and final positions of babbling trajectories are on this arc, but there is no constraint on points between the initial and final positions.	107

5.24	Evaluation of planned motions for three trials, where $ \mathcal{A}' = 6$ is set for the trials and training is performed for 8 different start points: (1) $ \phi = 1$, (2) $ \phi = 3$, (3) $ \phi = 6$. (a) Euclidean distance error of end effector position of planned trajectories across start position's y in the robot's coordinate system for 8 start poses. (b) Euclidean distance error of end effector position of planned trajectories across absolute change in y position in the robot's coordinate system. (c) Euclidean distance error of end effector position across three trials with ϕ	110
5.25	Evaluation of planned motions for four trials, where $ \mathcal{A}' = 6$ is set for the trials and training is performed for various fixed start points: (1) $ \phi = 1$, (2) $ \phi = 3$, (3) $ \phi = 6$. (a) Smoothness of planned trajectories based on overshooting index across start position's y in the robot's coordinate system for 8 start poses. (b) Smoothness of planned trajectories based on overshooting index across absolute change in y position in the robot's coordinate system. (c) Smoothness of planned trajectories based on overshooting index across three trials with ϕ . . .	111
5.26	Evaluation of planned motions for four trials, where $ \mathcal{A}' = 6$ is set for the trials and training is performed for various fixed start points: (1) $ \phi = 1$, (2) $ \phi = 3$, (3) $ \phi = 6$. (a) Smoothness of planned trajectories based on norm jerk metric across start position's y in the robot's coordinate system. (b) Smoothness of planned trajectories based on norm jerk metric across absolute change in y position in the robot's coordinate system. (c) Smoothness of planned trajectories based on norm jerk metric across three trials with ϕ	112

5.27	Evaluation of planned motions for three trials, where $ \phi = 3$ and $ \mathcal{A}' = 6$ are for the three trials called: <i>lnrConnections</i> , <i>fixConnections</i> , and <i>parConnections</i> . (a) Smoothness of planned trajectories based on the norm jerk metric across four trials. (b) Smoothness of planned trajectories based on the overshooting index across four trials. (c) Smoothness of planned trajectories based on the norm jerk metric across absolute difference in y position of end-effector in the robot's coordinate system. (d) Smoothness of planned trajectories based on the overshooting index across absolute difference in the y component of end effector position in the robot's coordinate system.	115
5.28	Evaluation of planned motions for three trials, where $ \phi = 3$ and $ \mathcal{A}' = 6$: (1) the neural map's resolution was the median of difference between features, (2) $2\times$ median of features, and (3) $3\times$ median of difference between features in terms of (a) Smoothness of planned trajectories based on norm jerk metric across three trials. (b) Smoothness of planned trajectories based on overshooting index across three trials. (c) Smoothness of planned trajectories based on norm jerk metric across absolute difference in y position of end-effector in the robot's coordinate system. (d) Smoothness of planned trajectories based on overshooting index across absolute difference in y position of the end-effector in the robot's coordinate system.	118
5.29	Evaluation of planned motions for four trials, where $ \phi = 3$ and $ \mathcal{A}' = 6$ are for the three trials: (1) the neural map's resolution was median of features (2) the neural map's resolution was $2\times$ median of features. (3) the neural map's resolution was $3\times$ median of features in terms of (a) Euclidean distance error of end effector position of planned trajectories across three trials. (b) Euclidean distance error of end effector position of planned trajectories across the absolute change in y position in the robot's coordinate system.	119

Chapter 1

Introduction

1.1 Motivation

A genuinely independent and autonomous robot must be active in adapting to unseen circumstances and it must have the capacity to learn from past encounters. Robots must be versatile and persistently capable of adaptation in their surroundings and be motivated to investigate their body and their surrounding environment. These necessities demonstrate a rundown of issues in the commonplace robotic frameworks that are expressly customized for a particular assignment ([Law et al., 2011a](#)).

In contrast, infants learn to control their body and to interact competently with their physical environment. They can learn about their world in a short period and to coordinate their limbs within a few months. Children can acquire new skills and generalize those physical skills to a broader context while they are growing and are being exposed to a new environment.

1.2 Approach and Uniqueness

In this dissertation, we aim to model knowledge representation and learning mechanisms used by infants and to transform development of reaching skill for robots by applying mechanisms from developmental psychology studies in infants.

We hope that by mimicking the known learning process in infants, autonomous robots will be more intelligent, adaptable, and useful than traditional robots. Our objective is to build a neurally-plausible computational model of human infant reaching; this model is based on embodied artificial intelligence that emphasizes the importance of motor-sensory interaction of an agent and the world (Iida et al., 2004; Pfeifer et al., 2007; Pfeifer and Iida, 2004).

Developmental psychologist Jean Piaget believed that knowledge is acquired through physical or mental actions. Motor-sensory interaction with the world is the first of four stages of infant development and happens from birth to two years of age. Infants lack an internal representation of their bodies when they are born and by actively exploring at this stage, they are able to interpret their world and self (Piek, 2006).

A *motor-sensory space* defines the interface between the agent’s internal control processes (e.g., its nervous system) and the body-environment system. This space is defined by all the sensory inputs and all the motor outputs of the nervous system or corresponding artificial control system. For all but the simplest animals this is a very high-dimensional space. The term “motor-sensory” is an intentional inversion of the usual “sensory-motor,” since the latter can inadequately suggest an input-compute-output information-processing model of cognition. Rather, perception is an active process, which typically depends on the motion of the agent in its environment (Gibson, 1979). This is critical because the motor activity or exploration directs the gathering of perception of the environment in a more fruitful way (Sporns and Pegors, 2004). “Motor-sensory” reminds us that motor activity is fundamentally prior to sensation and perception, although of course the two occur in a tight loop established by the agent’s continuous physical engagement with its environment. The physical constraints of body-environment interactions define manifolds of possible trajectories in this motor-sensory space (Kuniyoshi et al., 2004).

In an analogy of vocal babbling in infants (Vihman et al., 1985), this motor exploratory stage is called *body babbling* or *motor babbling* by Meltzoff and Moore

(1997). This means that through this process infants learn the dynamics of their body and how to control it. In a similar developmental fashion, we wish to build a computational model that is: (1) *embodied*, (2) *developmental*, and (3) *neurally plausible*, to mimic infant development.

This dissertation falls within the area of developmental robotics. *Developmental robotics* is the combination of robotics and developmental science, mostly developmental psychology and developmental neuroscience. This research area is very similar to *epigenetic robotics*. The main important difference is that epigenetic robotics focuses primarily on cognitive and social development and on motor-sensory environmental interaction, while developmental robotics covers more issues such as motor skill learning and morphological development (Lungarella et al., 2003; Lungarella and Gomez, 2009).

We describe the advantages of each aspect of our model. We want our model to be embodied for the following reasons:

- Embodied motor-sensory coordination would allow us to exploit the dynamical and natural properties of the body and its environment (Brooks, 1997; Lungarella and Berthouze, 2002a, 2004; MacLennan, 2011).
- Embodied motor-sensory coordination enables us to build an implicit model of dynamics and kinematics for a complicated body-environment interaction.
- Trajectories in motor-sensory space, “trajectory bundles”, can be defined by the physical constraints of body-environment interactions (Kuniyoshi et al., 2004).
- We believe that the morphology and physical properties of an agent are important factors in shaping an agent’s behavior and cognition.

We want our model to be developmental because an infant is born into the world without having any knowledge about its body and the environment. The infant obtains the knowledge throughout the interaction with the world and body in developmental stages. Mimicking infant’s development is a promising approach to

building a control mechanism. This control mechanism can be adapted to varying robot arms and tasks in the environment.

And finally we want to have a neurally-plausible model which is intended to account for infant development. Nervous systems are the best mechanisms we know for controlling competent reaching, grasping, and shared object manipulation. A neurally plausible model in this dissertation means that it is not inconsistent with the brain’s functionalities. However, our model can not be directly mapped to different parts of the brain. We are particularly interested in a neurally-plausible model for the following characteristics:

- It is common to see high-dimensional sensory and motor areas in the brain.
- In the sensory and motor systems, neural connections are arranged such that adjacent areas in the periphery are represented by adjacent neurons in the brain (MacLennan, 1997, 2009). These neurons represent the information using population codes (Georgopoulos, 1996).
- It has been shown that neurons in the sensory-specific cortices are activated by different types of sensors. These connections across different modalities facilitate the process of the sensory integration (Neville, 2002).
- Hebbian and competitive learning are common mechanisms in the brain.

This model will ultimately be implemented and tested in a robotic system.

1.3 Contributions

The contributions of this dissertation will be

- To better understand the control of dynamically complex motor-sensory behavior in both humans and robots.

- To build an embodied, developmental, and neurally plausible computational model for the motor-sensory control of reaching.
- To investigate multiple dimension reduction techniques for reducing the abstract motor-sensory space.
- To investigate multiple neural representation techniques for the motor-sensory space.
- To implement spreading activity for path planning and path execution in the abstract motor-sensory space.

The following chapters are organized as the following: chapter 2 covers related concepts in infant development that help to understand the literature in developmental robotics. Chapter 3 reviews the research work in developmental robotics that serves as a basis for this dissertation. Chapter 4 introduces our conceptual model of infant reaching and the neural representation of this model and an implementation of this model. Chapter 5 shows some experiments performed regarding our model along with the results and discussion of the important factors. Finally chapter 6 summarizes the model and our findings and offers some recommendations for the future research.

Chapter 2

Background in Infant Development

2.1 Introduction

In this chapter, we will explain some aspects of infants' development that are related to this dissertation, such as brain development and manual control.

2.2 Developmental Trends

2.2.1 Rhythmical Stereotypies

Rhythmical Stereotypies are repetitive movements that infants engage in without the presence of any known stimuli. Unlike rudimentary behaviors, these transitional movements are not defined as voluntary or goal-oriented. Infants present these behaviors when they are capable of controlling their body parts in some degrees but not capable of performing voluntary or goal-oriented behaviors. It is known that these non-goal oriented movements happen in order and can be predicted (Gabbard, 2004, p. 259-261).

Some examples of this behavior are rocking on hands and knees, head banging, kicking, and scratching shown in figure 2.1. It is believed that infants are provided at

birth with these abilities to get ready for later more coordinated movements (Snow, 1989).

2.2.2 Mass to Specific Movement

General or Mass Movements develop in children before the Specific ones. For example, infants usually move their legs and arms at the same time (Mass Movement). Later on, as the result of body growth, infants will be able to move their arms without moving their legs and eventually move one arm without moving the other one (Specific Movements). Fine movements, like finger movements, will happen after that point (Snow, 1989).

2.2.3 Hierarchical Integration

Infants adopt a hierarchical style in learning skills. First, they learn simple and rudimentary skills, and those skills are used gradually as the building blocks of more complex movements. For example, grasping a raisin using the thumb and index finger and putting it inside a bottle demands the four core competencies of reaching, grasping, placing, and releasing to be integrated (Snow, 1989).

2.2.4 Developmental Direction

Motor control develops in a cephalocaudal-proximodistal trend in infants. That means, the baby's head and neck develop and function first, then the shoulder and upper trunk and later, the lower trunk and legs. The development also proceeds in a proximodistally order, starting from the shoulders to the elbows and then to the fingers (Snow, 1989; Gabbard, 2004).

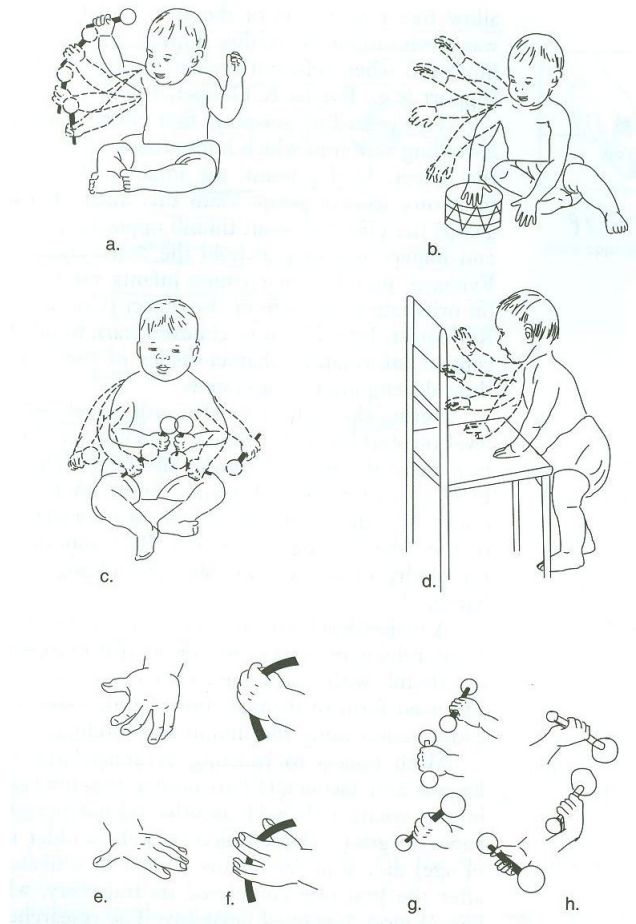


Figure 2.1: Examples of spontaneous (stereotypic) movements of the arms, hands, and fingers: (a) arm waving with object, (b) arm banging against a surface, (c) banging both arms together with object, (d) arm sway, (e) and (f) finger flexion, (g) hand rotation, (h) hand flexion (Gabbard, 2004).

2.3 Reflexes

One early movement in newborn infants is the reflex activity, which is defined as an involuntary response to a particular stimulus such as touch, light, sound, and change in pressure. The lower brain and spinal cord are mostly responsible for control of these activities, which will be replaced by voluntary movements as the upper part of the brain grows. These reflexes help an infant learn more about its environment and body, and they serve as a basis for the phases of motor development. Some of the primitive reflexes in the infant are shown in Table 2.1 (Snow, 1989; Gallahue, 1982).

Table 2.1: Developmental sequence for the appearance and inhibition of selected primitive reflex behaviors (Gallahue, 1982)

Reflex	Months	0	1	2	3	4	5	6	7	8	9	10	11	12
Moro	✓	✓	✓	✓	✓	✓	✓	✓						
Startle									✓	✓	✓	✓	✓	✓
Search	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Sucking	✓	✓	✓	✓										
Palmar-mental	✓	✓	✓											
Palmar-mandibular	✓	✓	✓	✓										
Palmar grasp	✓	✓	✓	✓										
Babinski	✓	✓	✓	✓										
Plantar grasp						✓	✓	✓	✓	✓	✓	✓	✓	✓
Tonic neck	✓	✓	✓	✓	✓	✓	✓	✓						

2.3.1 Palmar Grasping Reflex

As soon as a small object is put in the palm of an infant's hand, this reflex is triggered, and the hand is closed tightly around the objects without using the thumb. In the beginning the grasp is tight, and after the first month it becomes weaker and eventually is replaced by voluntary grasping by the age of 3 or 4 months. It is believed that palmar grasping plays a significant role in learning later voluntary grasping (Gallahue, 1982, p. 143).

2.3.2 Influence of Reflexes on Voluntary Movement

There are two views regarding this influence. In one view, reflexes form the foundation for all voluntary motor activities. In the other view, the reflexes stop the voluntary movements from emerging. It's hard to establish an exact relationship between

voluntary movements and primitive reflexes. It seems that some of the primitive reflexes gradually evolve into voluntary movements (Snow, 1989, p. 104).

Gallahue (1982) divides the reflexive phase of development into two stages:

1. **Information Encoding Stage:** At this time, the involuntary movements in infants are the sole noticeable activities. Also, since the motor cortex is less developed compared to the lower brain center, the lower brain is responsible for controlling the fetal and neonatal movements. This brain center triggers involuntary reactions to a variety of stimuli and information is gathered during this time.
2. **Information Decoding Stage:** This stage starts when reflexes are inhibited, and higher brain areas continue to develop. The motor cortex gradually controls the infants' voluntary movement, and the infants demonstrate more complex perceptual movements rather than simple sensorimotor activities. At this stage, a baby can process sensory inputs.

2.4 Manual Control

The term *manual control* encompasses two forms of behavior: prehension and manipulation. *Prehension* describes the initial voluntary use of hand during basic grasping and *manipulation* describes a child's skilled voluntary use of its hand during the ages of 5-8 years. Three building blocks of manual control are reaching, grasping, and releasing (Gabbard, 2004).

2.4.1 Pre-reaching

Pre-reaching is an essential behavior observed in infants before the age of 4 months; after that voluntary reaching emerges. If newborns are provided with full support of the head and trunk, they can extend their arms and hands toward an object. However, infants cannot successfully touch the object. This behavior is called pre-reaching

because it is not clear whether it is visually guided or visually elicited (Smitsman and Corbetta, 2010).

2.4.2 Goal-Directed Reaching

Most infants start the initial *goal-oriented* efforts in reaching and grasping by the age of 4 or 5 months. At this age, the trajectories of the initial movements are jerky, but after a couple of months of experience, they become smoother and follow a straight line. Infants can correct their reaching trajectories based on visual information by the age of 7 months, and they can adjust their reaching trajectories based on their previous *experiences*. There are other factors involved in the development of goal-oriented reaching such as posture control, head stabilization and movement speed control (Gabbard, 2004, p. 274-276).

2.4.3 Grasping

The first grasping attempt is usually a corralling action where both arms and hands are used to pull an object. At this stage, the infants can pick up objects using an immature palmar grasp. Naive palmar grasp is described by using thumb and fingers as a whole to hold an object without making a thumb opposition. Later, infants are capable of adjusting their grip size to match the size of the object and its orientation (Gabbard, 2004).

At age 5 or 6 months, infants can grasp objects with one hand using *pseudo thumb opposition* shown in figure 2.2. By the age of 8 months, they are capable of holding two objects with both hands. At the age of 9-10 months, major progress in grasping happens when an infant grasps a small object with a *pincer grasp* or actual opposition. Figure 2.2 depicts the changes in object positioning during development (Gabbard, 2004).

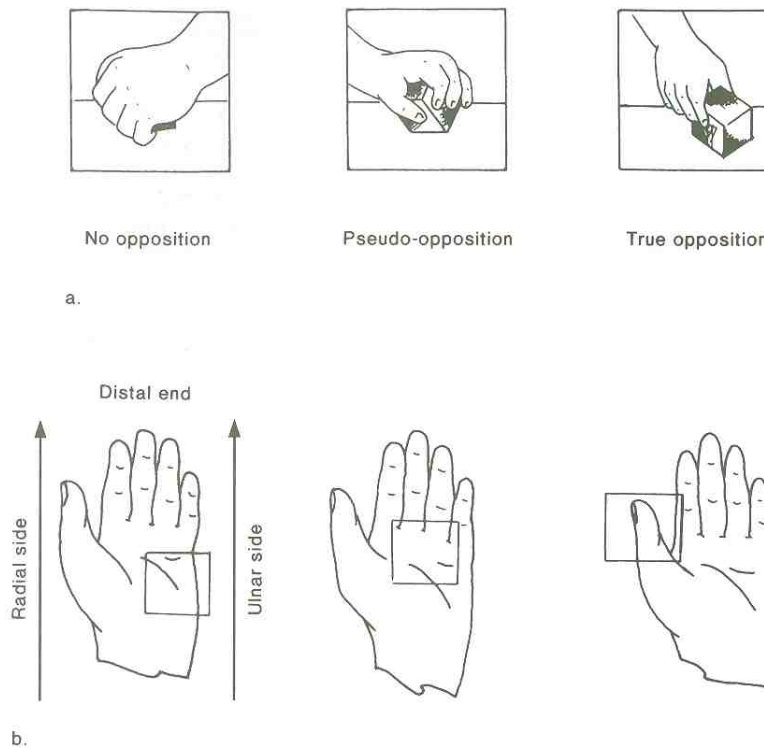


Figure 2.2: (a) Basic grasping technique and (b) Changes in object positioning (Gabbard, 2004).

2.4.4 Releasing

Releasing is another form of manual control where infants must accurately control and relax the arm and hand muscles to release an object. Since it is a rather difficult task, this component of manual control happens after the other ones are learned. An infant can release objects at age eight months by opening its hands and dropping them. An infant is not able to perform an accurate release until 18 months of age (Snow, 1989).

The primary factor in this behavior is the ability to perceive the object's weight and control the amount of applied force. It seems that infants use similar power in the arm and grasping action regardless of the object weight. By month 9, a child can adjust the force with the object's weight right after the initial grip. Eventually at the

age of 18 months, they can predict the object's weight and use the right amount of force (Snow, 1989).

2.5 Brain Development

There are two primary processes in the brain that control the brain development. The first one, *synaptogenesis*, is responsible for generating the neural connections. The second one, *apoptosis*, is in charge of the neurons' death (Piek, 2006).

Synaptogenesis starts early in the infant's development and due to this process, the brain grows rapidly during the first few years of childhood. Studies on the cerebral cortex of children have shown that from the ages of 24 to 72 months, there was a 60% to 78% growth in the total number of cortical neurons compared with the number at birth (Piek, 2006).

Apoptosis results in cell death, and greater or lesser degrees of apoptosis are responsible for both decrease and increase in some cortical neurons. This process is especially in charge of removing overproduced cells in the embryonic period. It is not clear if an infant is provided at birth with this process or not. The survival of new cortical cells is dependent on their use and the activity of appropriate pathways (Piek, 2006).

After looking at these two processes, we should mention that different parts of the brain develop at different rates. The bottom of the brain that is responsible for involuntary early movements, reflexes, and some core functions form and become functional first. The upper part, which controls voluntary movements, speech, and cognition, develops later. By the age of 3 months, the brain has developed enough to control the upper parts of the body. From age 6 to 15 months, the part of the brain that controls thought and learning processes develops quickly (Snow, 1989).

Chapter 3

Literature Review on Developmental Robotics

We previously introduced Developmental Robotics in chapter 1. This field stresses the importance of environmental and intrinsic components in the formation of human behaviors. Remarkable changes and sequences of behaviors are the main characteristics of human development. Particularly, these properties are more noticeable because during infancy one pattern of behavior is replaced by another one. We can identify the periods of growth and consolidation in the human infant. Piaget, a pioneer in the field of developmental learning, emphasizes the significance of sensory-motor interaction, skill learning through multiple stages, and a constructivist approach (Asada et al., 2009).

In this chapter, we begin with the important components of development that are used in Robotics (seen in Table 3.1). This table was compiled by Lungarella et al. (2003) and we used these aspects as a way to organize the related studies in this chapter. It is recommended that these aspects be considered during the design and construction of a system that simulates infants' development. A single study might cover multiple components, but not necessarily all of these components. Some of the important features of our model are that is neurally plausible, uses motor babbling

to learn inverse dynamics for arm trajectories, and is tested on a real robot. Here we review related studies that serve as a basis for this dissertation. Each of these studies shares some, but not all, of the qualities listed above that make this dissertation an original contribution to the field.

Bernstein (1967) first hypothesized a three-stage solution for the problem of abundant degrees of freedom. In the first step, the peripheral degrees of freedom are minimized to a small number. In the following stage, the disabled degrees of freedom are incrementally added back into the system while the agent trains at each increment. In the final step, reactive phenomena such as gravity and passive dynamics are added into the system, and the agent exploits the new system. The freeing and freezing approach was not implemented in this dissertation but could be used in the future work to structure the babbling stage.

Lungarella and Berthouze (2002b) offered a model for a robot to learn to swing by reducing the number of available degrees of freedom, which helps to support the interaction between the environment and the neural model. The focus of this work is not reaching and grasping; nevertheless, it is a good example of managing the excess of degrees of freedom.

Nagai et al. (2003)’s work is inspired by the developmental process of infant shared attention, which goes through different stages: the ecological stage at 6 to 9 months old, the geometric stage at 12 months old, and the representational stage at 18 months old. In this paper, a constructive model is built to reproduce these three steps, where a robot learns the shared attention skill by interacting with a human. This work is an example of the social interaction aspect of developmental robotics. The foundation of this paper is similar to our approach; we used the sensorimotor correlation to facilitate motor skill learning in a robot.

Table 3.1: Components of Development adapted from (Lungarella et al., 2003).

Component	Synopsis
Incremental process	Prior structures and functions are necessary to bootstrap later structures and functions
Importance of constraints	Early constraints can lead to an increase of the adaptivity of a developing organism
Self-organizing process	Development and learning are not determined by innate mechanisms alone
Degrees of freedom	Constraining the movement space may be beneficial for the emergence of well co-ordinated and precise movements
Self-exploration	Self-acquired control of body dynamics
Spontaneous activity	Spontaneous exploratory movements are important precursors of motor control in early infancy
Prospective control, early abilities	Predictive control is a basic early competency on top of which human cognition is built
Categorization, sensorimotor co-ordination	Categorization is a fundamental ability and can be conceptualized as a sensorimotor interaction with the environment
Value systems	Value systems mediate environmental saliency and modulate learning in a self-supervised and self-organized manner
Social interaction	Interaction with adults and peers is very important for cognitive development

Berthouze and Lungarella (2004) introduced a nonlinear coupling between the environment and the robotic system by connecting a rubber band from a supportive frame to a humanoid robot at hip-level. The robot in this study learned to swing, a repetitive action that emerges during the first year of human life. They showed with this situation that only one stage of freezing and freeing of degrees of freedom is not adequate to produce a good level of performance in a swinging task. Instead, alternating freezing and releasing of degrees of freedom produced an optimal performance. Although reaching and grasping are not repetitive activities, the freezing and freeing and freezing idea used in this work could be useful for the future work of this dissertation.

Gomez et al. (2004) and Gomez and Hotz (2004) mimicked infant development in robot reaching in three concurrent stages. Their method first gradually increases the resolution of the visual and tactile systems. Second, it freezes and releases degrees of freedom in the motor system. Third, it gradually adds neurons to the neural control architecture. The robot in their experiments has to learn to bring a colored object in the robot's hand from the periphery to the center of the visual field. This work is similar to our approach, since it falls into the category of embodied artificial intelligence and developmental robotics, but it is not neurally-plausible.

The objective of Oztop and Arbib (2001) and (Oztop et al., 2004) is to emulate the process of grasp development in infants based on motor babbling. The system is composed of a hybrid neural control circuit and a 19 DOF arm, which uses forward kinematics to simulate the motion of the arm and hand. The neural circuit learns the grasping task using the Hebbian learning and utilizes the inverse kinematics for reaching. Their model is able to generate grasps without visual feedback. This means that the model makes a motor plan in response to sensory stimuli. This aspect of the model is inspired by the fact that infants are able to touch glowing and loud objects in a dark environment. This work does not use any dynamics at the implementation level and does not consider the haptic and proprioceptive feedback from the hand to

the F5 area via somatosensory cortex in its modeling. The reaching model also solves inverse kinematics, but we are trying to solve inverse dynamics.

Kuniyoshi et al. (2004) studied the emergence of symbolic behaviors and communication from physical dynamics of a robot and its sensorimotor interactions with the real world. The behaviors and interactions emerged through self-exploratory learning of body schemas. In one simulation, a baby body is attached with several tactile sensors and is surrounded by water. The spatiotemporal correlation is computed for all pairs of sensing points. In another simulation, a robot explores its behavior without having any predefined behavioral primitives. This simple experiment consists of a ball attached to a stick with tactile sensors spread out all over the ball and the stick. A simple two-layered neural network is trained for sensorimotor correlation. Interestingly, the system learns some patterns such as lifting the bar and swinging the bar. Relating to the self-exploratory idea, in the last experiment a visuomotor neural learning system is designed. The robot learns the temporal sequence of the sensorimotor vectors by generating random arm movements and watching them. The inputs of the neural network are the visual motions and proprioceptive data. The learning trajectory bundles in this work are very similar to ours, and we used this work as a basis for trajectory learning in our model.

Steels (2004) proposed a general system where a complicated organism itself could control the set of skills that needs to gain and knowledge about its body and environment. The agent doesn't require someone to change the environment, or to set the reward functions, or to introduce resources progressively on-line in a maturation plan. The main idea is inspired by humanistic psychology where people are involved in some activities for the sake of doing it without receiving direct rewards. These activities are called "autotelic," which emphasizes that the motivation originates from the individual itself rather than from an outside source. These activities are different from the ones that are directly pleasurable; the activity must itself be challenging. Also, there must be a steady progression in level of challenge and a balance between the complexity of a task and the required skill to perform.

Asuni et al. (2005) build a neural model for visuomotor coordination of a robotic manipulator in a reaching task. This model is based on a self-organizing neural network that learns the correlation of motor actions and sensory feedback. This system maps between the position of the arm in 3D Cartesian space and its joint space. This model was tested through four different experiments: reaching in the usual condition, reaching with clamped joints, reaching with a tool, and blind reaching which aims to show that the model is adaptive under perturbation. The mapping in this model solves the inverse kinematics problem while we are trying to solve the inverse dynamics problem. The set of experiments in this study was outside the scope of this dissertation but they can be used for the future experiments.

The system introduced by Demiris and Dearden (2005) learns multiple forward models without any prior information by means of motor babbling and a Bayesian belief network. This model was tested on both real and simulated robots. In this system, which is inspired by human hand movement, an association between motor commands and the position of the moving gripper is learned. The main focus of this work is not reaching, but it is nevertheless a good example of motor babbling in imitation.

Josh Bongard (2006) used actuation-sensation relationships for a four-legged robot to indirectly infer its structure and then generate forward locomotion using that assumed structure. Also, this robot can recover from an unexpected damage autonomously through continuous self-modeling. The robot uses an active process to conclude its structure by engaging in an exploration phase that is directed by the robot itself. First, the robot executes a random motor action and observes/stores the sensor values. Then, the “model synthesis component” creates a set of 15 potential models for the robot using stochastic optimization. This stage aims to explain the sensorimotor correlation. The “action synthesis component” then uses these self-created models to identify a new action that will lead the most information from the robot.

A new action is found by looking for a motor command that would produce the most difference among the predicted sensors resulting from the different models. After the new motor action is determined, it is submitted to the robot for the execution. Meanwhile, the “model synthesis component” repeat the synthesis process while it has more data to evaluate the model. The entire process continues for 16 iterations and finally a model with the least amount of error is selected. The “behavior synthesis component” uses this final model to produce behaviors that can be sent to the robot for execution. Even though the task here is not reaching, the idea of self-modeling could be useful for sensorimotor prediction in our future models.

Lee and Meng (2005) introduced a robotic system that learns the association between proprioceptive and motor space by taking advantage of natural constraints with a common learning approach. Those natural constraints are active and idle sensors, the presence of objects, and resolution of sensors. The robot of this experiment consists of two manipulator arms with 2 DOF and a visual sensor that serves as an “eye.” Each arm is attached to a two-finger gripper with tactile sensor pads all over it.

Lee et al. (2007a) and Lee et al. (2007b) offered the LCAS (Lift-Constraint, Act, Saturate) algorithm to learn hand/eye coordination. At the beginning of the LCAS cycle, all or almost all constraints are imposed, and there is a small opportunity for any complex activity. In each cycle, the system gradually removes a restriction and explores (Act) all the possible new experiences until the learning saturates. The computational framework of this work is based on a two-dimensional map, where the map consists of circular overlapping and regularly spaced receptive fields. Each field contains a set of variables to keep track of the state and properties of the map. In this work, a correlation map between motor and sensor space is built, and it doesn’t focus on trajectory planning and reaching.

In this work, the proprioceptive sensors are calculated as “joint angle encoding”, “shoulder encoding”, “body-centered encoding”, and “Cartesian encoding”. These

encodings are good examples of mimicking proprioceptive sensors where we didn't implement them in the current research but can be used in the future work.

To automatically indicate when to stop an adaptive phase or when a map has become saturated, two global variables are used. The *global excitation* is a cumulative excitation, which is the sum of excitation levels of all those fields with the excitation level above a given threshold. The *global conservancy* (a normalized value) is the inverse of the summation of the frequency levels and shows the “familiarity” of the map. Global excitation illustrates the strength of the current activity, and global conservancy measures the novelty of the fields being experienced. These global indicators are used to remove constraints in two different ways: sensory maps are created with finer resolution when we have an overall high familiarity, and the motor actions are more spontaneous when the global excitation is small. A basic “reflex” is provided to start the system when the total excitation levels are below a given threshold.

Lee et al. (2007a) identified a trade-off between required time for exploration and accuracy of the motor actions by changing the sensory resolutions. When small fields are used, the specification of sensory space is fine and reaching motions to a given location are more accurate, but at the same time, much more exploration is needed to generate those mappings. On the other hand, when large fields are used, more sensory space can be covered and, therefore the mapping is acquired much quicker.

The model offered by Law et al. (2011a) is an extension of the work by Lee et al. (2007a) and is built upon the same constraints and the same LCAS algorithm which guides sensorimotor learning in an iCub humanoid robot. The primary goal of hand/eye system is to find two correlation maps: one between retina and gaze space, and one between reach and gaze space. Also, they investigated how biological processes, motor, and vision development can be transferred to the iCub humanoid robot. The former transfer is shown in figure 3.1 and the latter transfer in figure 3.2. The approach used by Law et al. (2011a) is built upon shaping and constraints; the structured motor babbling offered here can be useful for the future work since the

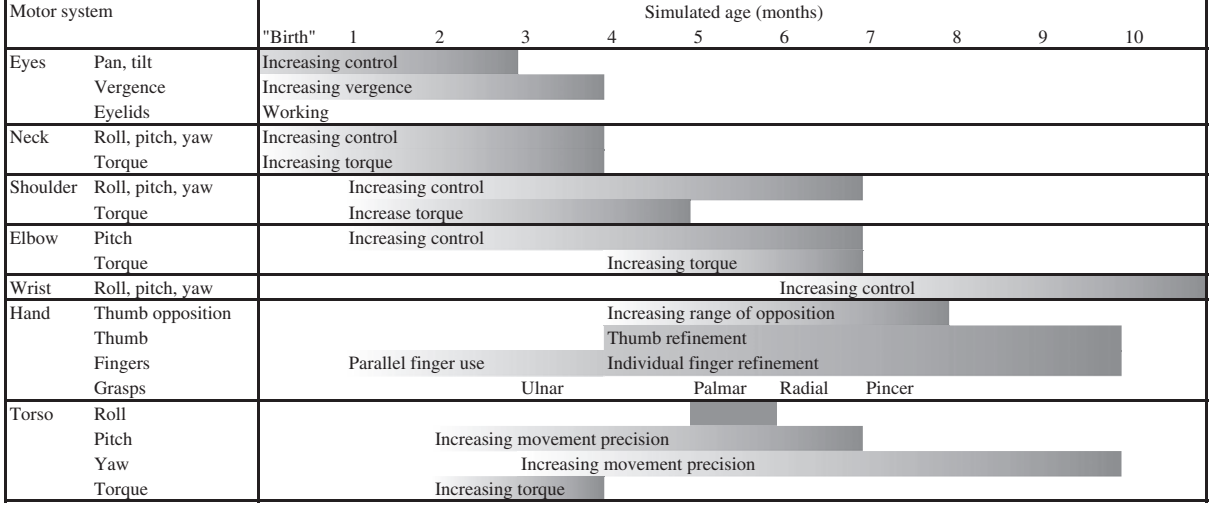


Figure 3.1: Partial motor development sequence for the iCub humanoid robot. Shaded regions relate to periods of development of each ability as observed in infants. Darker shading indicates more advanced abilities (Law et al., 2011a).

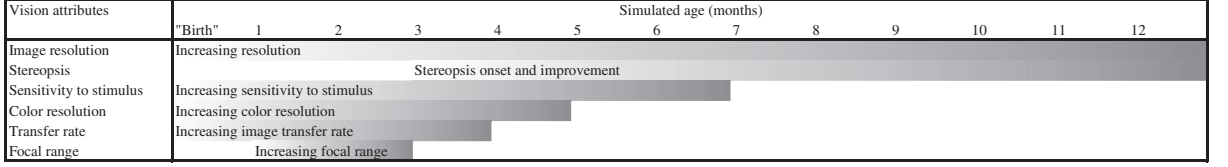


Figure 3.2: Partial vision development sequence for the iCub robot (Law et al., 2011a).

random motor babbling didn't perform well in the current study. In this work (Law et al. (2011a)), the stages of eye-saccade learning have been implemented on iCub while the reaching stages are in progress.

Caligiore et al. (2008) used primary circular reaction hypothesis which focuses on motor babbling in front of the eyes. During the motor babbling, Hebbian learning rules are used to form correlations between actions and perceived consequences for two rather complex skills of reaching in the presence of an obstacle and grasping.

When the hand touches the objects, the grasping movements are started; this is inspired by the *enclosure reflex* in young infants. Fig. 3.3a and Fig. 3.3b show the reaching and grasping architectures. Their computational framework consists of two 2D maps of neurons with population codes that encode signals for input

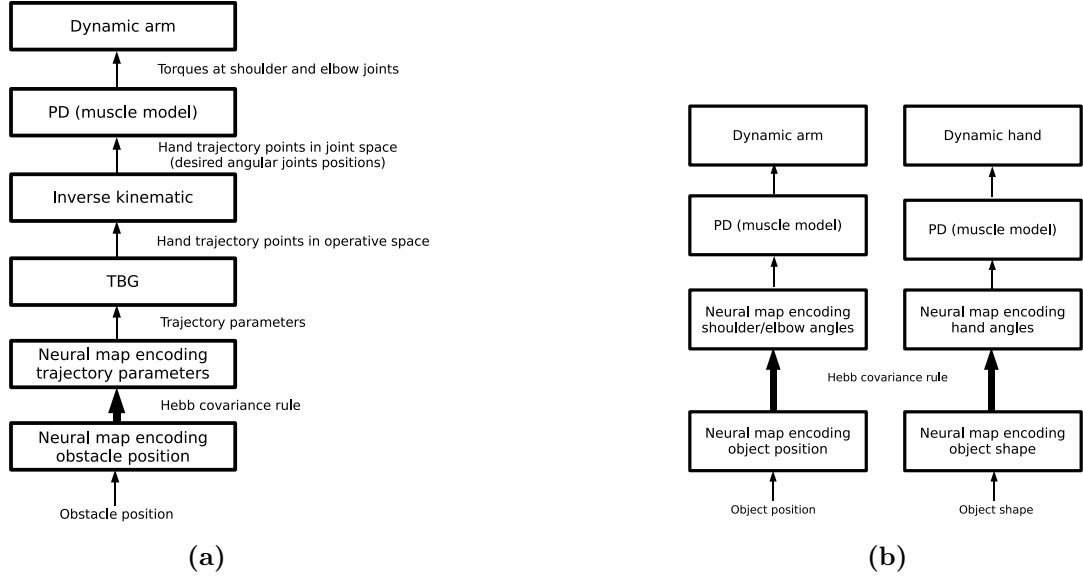


Figure 3.3: The (a) reaching and (b) grasping architecture (Caligiore et al., 2008).

and outputs. In reaching, the input map encodes the position of an obstacle in “eye-centered coordinates”, while the eye fixated on the target before and during the movement. The output map encodes parameters used by a “pattern generator” to create the hand trajectories. During the learning stage, random parameters are created to perform motor babbling. At this stage, parameters that produce illegal motions are filtered out, such as the ones that lead to a collision with objects or any angle violation of joints. This work was not tested on any real robotic system, and they employed a 3D simulated dynamic eye-arm-hand instead. They are mainly trying to solve the inverse kinematics problem with the reaching rather than the inverse dynamics.

Laschi et al. (2008) offered a predictive sensorimotor coordination system for robot reaching inspired by infant development, as well as pre-shaping fingers for a grasp configuration, using neuro-fuzzy networks. Reaching controls the final position and orientation of the arm end effector but not the arm’s trajectory.

Saegusa et al. (2008a) designed a system that learns to predict future sensor values by using current sensor values and motor commands. The sensory-motor

learning procedure has two stage of exploration and learning. The system alternates between these two stages until the given performance is attained. The sensorimotor prediction was evaluated by a function called *confidence*. The principal idea is to use confidence acquired from the learning happened in the past to explore and collect new information.

The exploration strategy is improved by [Saegusa et al. \(2009a\)](#) and [Saegusa et al. \(2009b\)](#). In an experiment with a humanoid robot named James, the sensors determine the position of robot’s hand and the motor commands determine the joints’ positions of the upper-arm and the shoulder. [Saegusa et al. \(2008a\)](#) performed the same experiment with James except that the motor commands are velocity commands for the upper-arm and shoulder joint.

[Saegusa et al. \(2008b\)](#) conducted two experiments that focus on learning to predict sensor values such as somatosensory and visual. The somatosensory prediction, which is the prediction of encoder values, was conducted using the left arm of James. The visual prediction learning uses the images captured by the left eye as the sensory inputs. In these experiments, the output data are motor commands that are transferred to the actuators of the left arm and the head.

In another experiment by [Saegusa et al. \(2009b\)](#) with the iCub simulator, two types of movements such as object fixation and reaching are learned. In the object fixation learning, the sensor inputs are horizontal and vertical positions of the robot’s hand on both the left and right eyes. The motor commands correspond with “horizontal”, “vertical”, and “vergence” movements of the both left and right eyes. Regarding the reaching movement, the sensor state is a 3D input consist of the horizontal, vertical, and vergence position of both eyes. The motor commands correspond to the roll of the upper-arm, and shoulder and elbow pitch. Fixation learning is done prior to learning reaching.

[Saegusa et al. \(2010\)](#) tackled a different aspect of development and embodiment, which is body discovery. In their model, a robot discovers its body based on the correlation between two separate sensory feedbacks of vision and proprioception. A

robot can identify that a moving object is part of its body when there is a high correlation between these two sensors. At this time, some important features of the moving object and the body itself are stored in a visuomotor memory. This memory can further assist the robot to define its body without having any previous knowledge.

Saegusa et al. (2012) preformed three experiments of body discovery: without human interference, with interference from a moving object, and by modifying the arm by putting a plastic glove over the arm. In all of these experiments, the system works without prior knowledge of body kinematics, appearances, dynamics, or motor pattern. Here, the sensorimotor correlation was used solely for sensory prediction while we used sensorimotor correlation for both trajectory planning and sensory prediction.

Bouganis and Shanahan (2010) trained a neural network that controls a robot’s arm with four DOFs. This network consists of individual spiking neurons and employs a learning technique that is biologically plausible called “Spike Timing Dependent Plasticity” to modify the strength of connections. This model finds motor commands that cause the end-effector to move to a given spatial location (inverse kinematics); in contrast, we are trying to solve the inverse dynamics problem.

The neural network by Bouganis and Shanahan (2010) is a feed-forward type network with its input layer encode both the end-effector’s spatial location at the next time step and the proprioceptive sensory information (fig. 3.4). The firing patterns in the proprioception group indicate the angle of the respective joints located in the shoulder and the elbow of the arm. The input layers are all connected to the output layer, where the neurons in the output layer encode the motor commands which are sent to joints. Each node of the input layer is attached with both an excitatory and an inhibitory connection to each node of the output layer. In the training stage, an “Endogenous Random Generator” randomly simulates motor neurons in the range of $[-5^\circ, 5^\circ]$, and the produced motor commands would cause the end-effector to move in a particular spatial direction (based on forward kinematic equations), which is stored in the neural network.

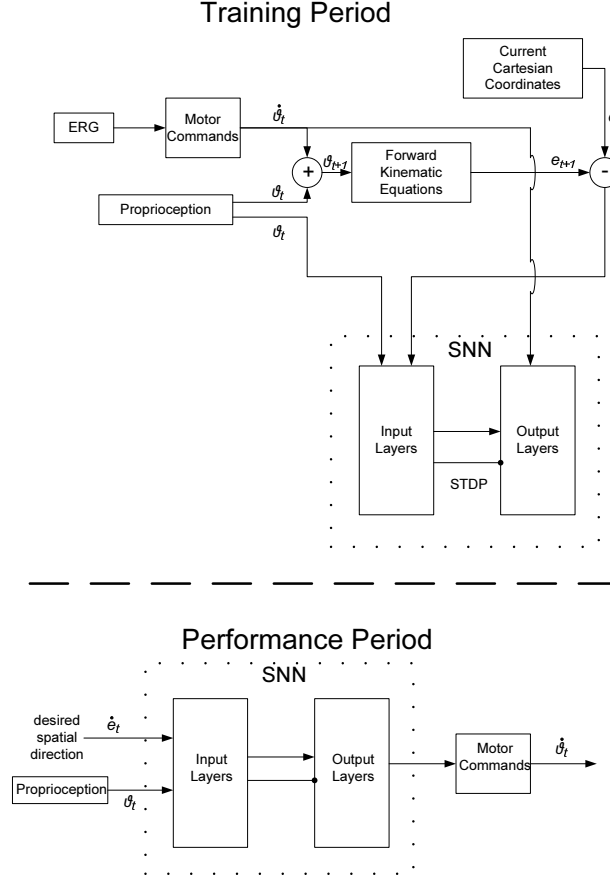


Figure 3.4: The spiking neural network framework during training and test (Bouganis and Shanahan, 2010).

Husle et al. (2010) suggested a framework that allows an active vision system with a robotic arm to conduct reaching that is guided by visual cues. The introduced framework consists of three different spaces, the “retinotopic reference frame”, the “gaze space”, and the “reach space”. The visual information captured by the active vision system are represented and calculated in the retinotopic frame of reference. The gaze space is defined by the range of motor positions in the system. The reach space is specified by the robot arm coordinate system that determines the reachable points in this space. Two mappings in the framework that link these three domains are the sensory-motor mapping for “eye saccades” and the mapping between reach and gaze space. The structure is embodied and based on infant development and

partially based on brain research. In our work, however, the reaching task is not visually guided.

Husle et al. (2011) mimicked some fundamental characteristics of infant development such as active vision, visual attention, coordination between hand and eye, and manipulation of simple objects.

Kraft et al. (2010) introduced a framework for visuomotor coordination where an agent is exposed to a minimum knowledge of its body, the machinery for feature extraction, structural knowledge, some innate behavioral patterns, and the physical world. The developmental process consists of three stages of learning the concept of objects, acquiring particular grasping knowledge for objects, and performing object manipulations. This system mimics the visuomotor learning of a six-month-old infant by using an initial premature grasping behavior, and it also uses the visual data to trigger “reaching and grasping” and “development of object representations.”

However, there are some differences between human development and this system. First, these developmental stages are not in a clear order in human development. Second, during play, more knowledge such as movement fine-tuning, body alignment, and sophisticated grasping affordance are learned besides learning to grasp a particular object. Finally, planning develops continuously in humans, where the skill sets and their combinations gradually become more advanced.

Thill and Ziemke (2010) used self-organizing maps to learn motor primitives, inspired by the mirror neuron system, but their experiments were based on abstract simulations of limbs, not on realistic dynamical simulations or real robotic arms.

Sauser and Billard (2006) used a neural field approach to model the dynamic integration of motor and sensory information. Their approach is very compatible with our own field approach, and they present a valuable mathematical analysis, but their investigation extended only so far as numerical simulation of some general effects.

Schaal et al. (2007) unified self-organizing dynamical systems approaches, like ours, with traditional optimal control; this incorporates earlier relevant work on

dynamic motion primitives by [Ijspeert et al. \(2003\)](#); this is very valuable work, but it is more focused on imitation than autonomous exploration.

[Rolf et al. \(2010a\)](#) used goal babbling (unstructured motor-exploration) as an exploration strategy to learn inverse kinematics. The focus of this method is to explore the surroundings of a sub-space with a low dimension. Since motor babbling’s goal is to explore the entire joint space, they believe that goal-directed babbling is more feasible for many DOFs. A path-based sampling approach is used to introduce targets to the model. Training data are generated along the paths, which are the results of execution of currently learned estimated model along a desired path toward goals. In this work, the main focus is to solve inverse kinematics using sensorimotor correlation while we are trying to solve inverse dynamic. This approach is not neurally plausible but is developmentally feasible. The motor babbling introduced here can be useful for the future work of our research.

[Rolf et al. \(2010b\)](#) addressed the challenge of body growth using the goal-directed method suggested by [Rolf et al. \(2010a\)](#). This model was tested with experiments on various patterns of growth (un-proportional or proportional growth) on a simulated robot arm and a simulated growth on a humanoid robot. In both cases, learning to reach was the central goal. An online version of goal babbling for bootstrapping the sensorimotor coordination was introduced by [Rolf et al. \(2011\)](#). The implemented technique can rapidly solve the inverse model for very high dimensional domains.

[Lee \(2011\)](#) introduced motor babbling as a form of play behavior. Playing has been known to be an essential activity for children to develop healthy regarding cognitive abilities. The four types of play that are particularly relevant to robotics are attunement (early adjustment of sensorimotor parameters), body (motor babbling with any of the limbs), object (manipulations and actions on objects), and social (interaction with other people) plays.

[DeWolf and Eliasmith \(2011\)](#) offered a hierarchical model of a reaching controller inspired by optimal feedback control and motor babbling. This model of motor control is based on the main brain areas is offered (figure 3.5) and each of the function of

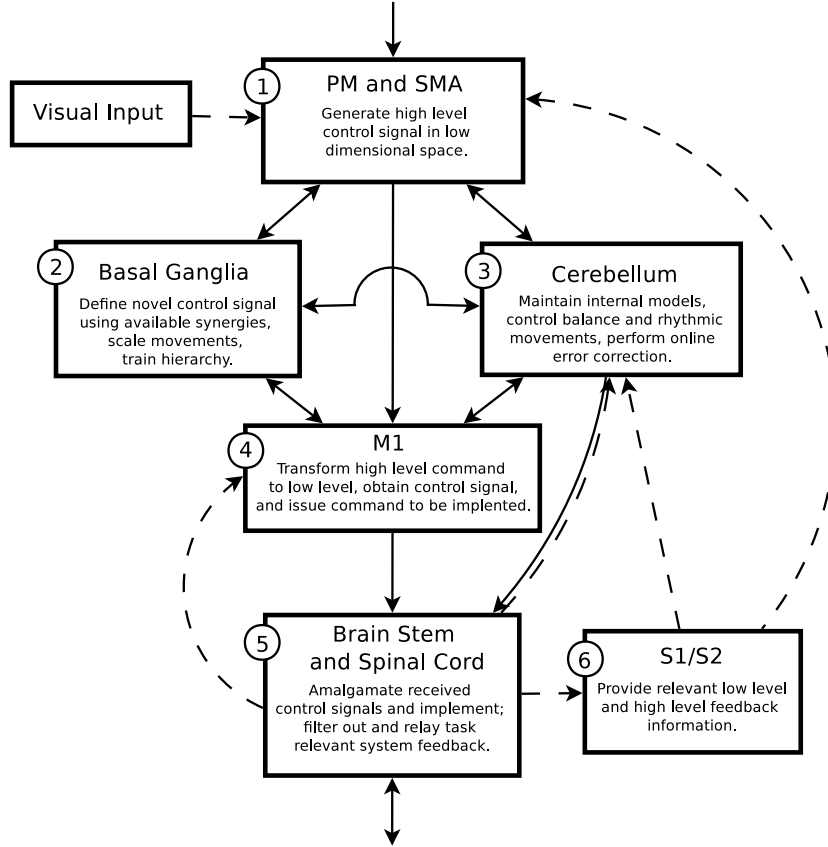


Figure 3.5: The NOCH framework (DeWolf and Eliasmith, 2011).

this model is mapped to the steps for generating optimal control signals. Functions of this model are also mapped to primary parts of the brain known to be involved in reaching. In an experiment, normal human reaching trajectories along with velocity profiles were produced by means of a hierarchical control system with two levels that included basal ganglia, motor cortices, and the cerebellum. Later in this experiment, some functionalities of the system were impaired, the results were compared with data from clinical studies of patients where their similar neural parts had substantial problems. This paper is a great example of a neurally-plausible approach for reaching, but this approach is not tested on any real robotic system, and the model of reaching is not inspired by infant development.

Law et al. (2014b) proposed a reaching model through simulating infant development on iCub. They focus on learning through developmental stages and motor

babbling. Simulated sensorimotor spaces are represented by overlapping maps of fields that resemble topographic maps in the brain. The work by [Law et al. \(2014b\)](#) is an extension of previous works by [Law et al. \(2011b\)](#), [Law et al. \(2011a\)](#), and [Husle et al. \(2011\)](#) and covers the acquisition of saccade, gaze, control of torso, and reaching and grasping that are visually-elicited in 3D space. This work is an excellent example of a longitudinal approach to development that starts from motor babbling and continues to the reaching and grasping stage. In this work, the effect of the torso is investigated on the robot’s representation of the space. Here, the LWPR algorithm was used to learn the relative rotation of torso and tilt that is needed to move an object from one location to another one within the gaze frame of reference. This algorithm is suitable for learning incrementally from sparse high dimensional data. The relationship between infant development and this longitudinal experiment was fully investigated by [Law et al. \(2014a\)](#).

We mentioned earlier that, at the core of this model, there are overlapping maps of fields with linkings that store the correlation in sensory-motor learning. [Earland et al. \(2014\)](#) investigated the effect of overlapping fields in sensory-motor representation.

[Caligiore et al. \(2014\)](#) introduced a computational model for development of reaching by integrating “reinforcement learning”, “equilibrium points”, and “minimum variance”. The model was tested with a simulated 2 DOF arm. The model can reproduce several known characteristics of an infant’s reaching, including the kinematics and dynamics of reaching trajectories in infants, the bell-shape velocity profile, the evolution of sub-movements and the control of degrees of freedom in reaching. The focus of this work is capturing the essential features of reaching and not the neural-plausibility of the model.

Chapter 4

Methodology and Implementation

In the following section, we describe our proposed conceptual model of reaching. In section 4.2, we explain a neural model for the proposed abstract model. In section 4.3 we examine an implementation of this model.

4.1 Conceptual Model

4.1.1 Motor-sensory Phase-space and Trajectory Bundles

We take our inspiration from the embodied development of the human motor-sensory system, in which an infant must learn the dynamic relationship between its body and environment. Focusing on the arm, we introduce a model for learning the correlation between motor action and consequent sensation. Let \mathcal{S} be the space consisting of all possible states of the agent’s sensory inputs, and let \mathcal{M} be the space of all possible states of the motor output system. We are interested in trajectories in the motor-sensory space $\mathcal{A} = \mathcal{M} \times \mathcal{S}$, which has the dimension $n = ms$.

If we consider s sensory inputs for the agent and the activity of each input is normalized to $\mathcal{I} = [-1, 1]$, then $\mathcal{S} = \mathcal{I}^s$. We can formally represent this space using $\hat{\mathcal{S}}$ disjoint sub-spaces: $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_{\hat{\mathcal{S}}}$. Each of these sub-spaces has

the dimension of $s_1, s_2, \dots, s_{\hat{\mathcal{S}}}$, respectively. Likewise the motor space is shown by $\mathcal{M} = \mathcal{I}^m$, which consists of $\hat{\mathcal{M}}$ disjoint sub-spaces $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2 \times \dots \times \mathcal{M}_{\hat{\mathcal{M}}}$.

To learn this correlation, we explore the dynamical properties of the arm, mimicking motor babbling in an infant. This exploration allows the agent’s neural control mechanisms to extract lower dimensional representations of the arm’s dynamics. During motor babbling, we record motor-sensory trajectories. Collections of these trajectories cluster as *trajectory bundles* in the abstract motor-sensory space \mathcal{A} . These trajectory bundles represent regions that are more dynamically feasible than the surrounding space.

Space \mathcal{A} has very high dimension, and direct neural implementation of this correlation learning could be computationally impractical. In our model, we use dimension reduction to create a more computationally tractable space \mathcal{A}' for learning correlations. There is evidence of dimension reduction mechanism in several brain areas, *e.g.* cerebellum and other motor-sensory systems.

4.1.2 Conceptual Trajectory Bundles Formation

Our goal in learning motor-sensory correlation is to construct trajectory bundles or dynamically feasible regions of \mathcal{A} . The correlations between motor actions and consequent sensations can be described by a scalar field $D(\mathbf{a})$, for $\mathbf{a} \in \mathcal{A}$. We are interested in regions of \mathcal{A} , where $D \approx 1$, versus the ones that are not dynamically feasible, $D \approx 0$. Let $\boldsymbol{\alpha}(t) \in \mathcal{A}$ be a motor-sensory trajectory, and let $\gamma_{\boldsymbol{\alpha}} : \mathcal{A} \rightarrow \mathcal{I}$ be an n -dimensional Gaussian distribution centered at $\boldsymbol{\alpha}$ with a suitable standard deviation. We use a Gaussian shape for the trajectories to emphasize that the borders of feasible trajectory tunnels are fuzzy and the fact that the dynamics are continuous. Fig. 4.1 shows a conceptual bundle of three trajectories in which parameter ϕ is used to show the fuzziness of trajectories.

The D field can be carved by either the following process [4.1](#)

$$\dot{D}(\mathbf{a}, t) = \eta_D[1 - D(\mathbf{a}, t)]\gamma_{\alpha(t)}(\mathbf{a}), \quad (4.1)$$

or in another form:

$$\dot{D} = \eta_D(1 - D)\gamma_{\alpha}. \quad (4.2)$$

Here η_D is the adaptation rate and D is the region of feasible motor-sensory correlation. One extension to this model is to allow the D field to adapt to changes in body dynamics by adding a slow decay term such as:

$$\dot{D}(\mathbf{a}, t) = \eta_D(1 - D(\mathbf{a}, t))\gamma_{\alpha(t)}(\mathbf{a}) - D(\mathbf{a}, t)/\tau_D. \quad (4.3)$$

We need another refinement because the feasibility of a phase-space trajectory may depend on the direction with which it passes through regions of \mathcal{A} . To accommodate this fact, we can construct a tensor field $\mathbf{D} : \mathcal{A} \rightarrow \mathcal{I}^n$ that encodes the facility of passing through each point in each possible direction. One way to construct this field is as follows:

$$\dot{\mathbf{D}} = \eta_D(1 - \|\mathbf{D}\|)\gamma_{\alpha}\dot{\alpha}. \quad (4.4)$$

4.1.3 Conceptual Trajectory Planning

In this stage, we aim to construct a trajectory/path from a dynamical starting point to the goal through the abstract motor-sensory phase space \mathcal{A} . The goal and the start points are both represented in this motor-sensory phase space, while the goal is shown by an “image of completion” $G : \mathcal{A} \rightarrow \mathcal{I}$. This function is a map from the points in this abstract space to a range that stands for the point’s attraction.

When an infant sees an interesting object located in its visual field, the image of completion will be triggered. Therefore, the image of completion is the combination of visual information (the location and features of the object) and infant’s feeling

when it holds the object. The image of completion for infant reaching is defined as the feeling of holding the object in its hand.

For example, in the case of an infant reaching for and grasping an object, the image of completion is the perception of grasping the object (tactile, proprioceptive, visual, etc.). Such an image of completion might be elicited by the sight of an interesting object at a particular place in the infant’s visual field. The visual information provided about the object’s location and properties (size, material, etc.) combines with the infant’s goal (holding it) to generate the image of completion. For example, the desire to grasp the object might generate goal haptic inputs in \mathcal{S}_1 , and the perceived location of the object would generate goal activity defined over the proprioceptive and visual fields (e.g., \mathcal{S}_2 and \mathcal{S}_3).

The purpose of the path-planning process, then, is to find an abstract trajectory from the current motor-sensory state into the goal region. In order to describe this process, one must imagine a D field which defines feasible “paths” through \mathcal{A} as ant trails in a high-dimensional phase space rather than a 2D space. In another words, we can think of $1 - D$ as representing infeasible regions that are outside of “trajectory tunnels” or “passible regions.” With this assumption, the diffusion of signal from the goal state to the current state shapes the path planning process. In this process, path planning and execution happens at the same time. The diffusion process is defined as

$$\dot{P} = \eta_P \nabla^2 D P - P/\tau_P + G. \quad (4.5)$$

Here, $P(\mathbf{a})$ is the amplitude of the path signal at $\mathbf{a} \in \mathcal{A}$. G is the goal motor-sensory state. To cover the cases that image of completion changes before the reaching finishes, we can add a rapid decay term, τ_P , so the potential paths can adjust quickly on the fly.

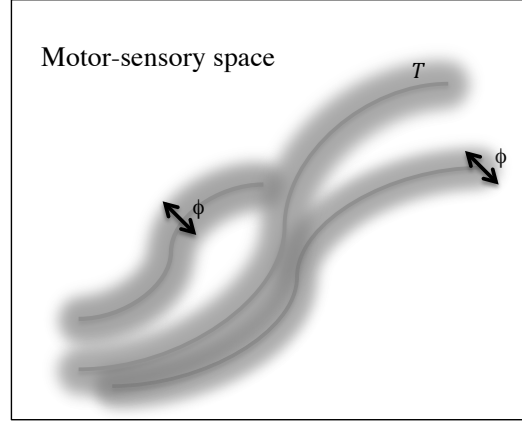


Figure 4.1: A conceptual bundle with three trajectories in which ϕ represents the width of trajectories to fuzzy the dynamic.

If we use a tensor field \mathbf{D} instead of a scalar field D , the path planning process can take into account the differing facilities of change along different dimensions:

$$\dot{P} = \eta_P \nabla^2 \mathbf{D} P - P/\tau_P + G. \quad (4.6)$$

Therefore, the signal diffuses most rapidly in the directions in which \mathbf{D} is greatest.

The diffusion-based path planning imposes conservation throughout the process, meaning the activities that are introduced via the source to the system equals the activities absorbed in the sink or dissipated in the system. This conservation law presents a set of problems for motion planning, for instance, for a start state located far from the image of completion, a signal might die before reaching to the start state. In contrast, start states in the vicinity of the image of completion can quickly be saturated with the activation, so the path planning fails.

We switched to spreading activation which is more appropriate to conceptualize and implement path planning in this dissertation. Since this process doesn't require conservation of signals in the system, the signal can be amplified or diminished in a neural network to handle short or long distance paths better. The process of spreading activation is a search method first introduced in cognitive psychology to

model retrieving concepts from memory that forms a semantic network. This process also has been used for document retrieval in a network of documents as well as a search mechanism in the artificial neural networks.

The search process begins with the source nodes in the neural network (e.g. the image of completion) and spreads out the activity to the other nodes; the source's activity propagates in the network according to the connections among nodes. This process iteratively updates the activation for all the nodes in the neural network. The nodes with an activation level higher than a given threshold are fired until the activity reaches the start state.

4.1.4 Conceptual Trajectory Execution

The path execution process can be performed by following the gradient of the path signal/diffusion or activation that we introduced in the previous section along the trajectory tunnels, $\dot{\alpha} = \eta_{\alpha} \nabla P$. We need to add a threshold θ to this process to assure that motor-sensory states change if the path signal is above the threshold; otherwise the state would change for any insignificant changes. Here is the complete path execution process:

$$\dot{\alpha} = \eta_{\alpha} \nabla [P - \theta]^+. \quad (4.7)$$

Here, $[x]^+$ is defined as, $[x]^+ = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$

This mechanism for path planning and execution has several characteristics that work best to our benefit:

1. the state of the motor-sensory system will not change until activation from the goal reaches the start state. At this point, a feasible dynamical path from the goal to the start has been determined, so the system can seek the goal.
2. since the path planning process is determined by following the gradient of the path signal, the path is likely near optimal and accessible.

3. since there is a slow decay term in the process of path diffusion, if the goal state changes the system is able to look for the new goal quickly.
4. in case the predicted sensory input doesn't match the actual sensory input in presence of any perturbations, the system automatically will follow the path planning from its new state.

In order to approximate the path planning and execution process in terms of the gradient, we use neural representation of trajectory tunnels. This approximation is especially helpful when there are two or more equally attractive paths. In this case, inherent stochastic mechanisms will break the symmetry, and one of the paths will be picked.

4.2 Neural Model

We take our inspiration from the embodied development of the human motor-sensory system, in which an infant must learn the dynamic relationship between its body and environment. Again, focusing on the arm, we introduce a model for learning the correlation between motor action and consequent sensation. The central feature of our model is the encoding of trajectory bundles in three maps of neurons with a parallel structure representing the same motor-sensory space. We refer to them as the *backward*, *forward* and *competition* maps. A particular motor-sensory state is represented by localized activity over the maps, and trajectories are defined by shifting activities among neurons with overlapping receptive fields. Fig. 4.2 shows an overview of this model with the three neural maps in the center and the dimension reduction and dimension expansion modules on the sides.

The connections between successively activated neurons encode both reverse-time correlations for path planning and forward-time correlations for path execution. The backward map \mathbf{B} represents connections from neurons activated at time $t + 1$ to neurons activated at time t . Connections between the forward map \mathbf{F} and the

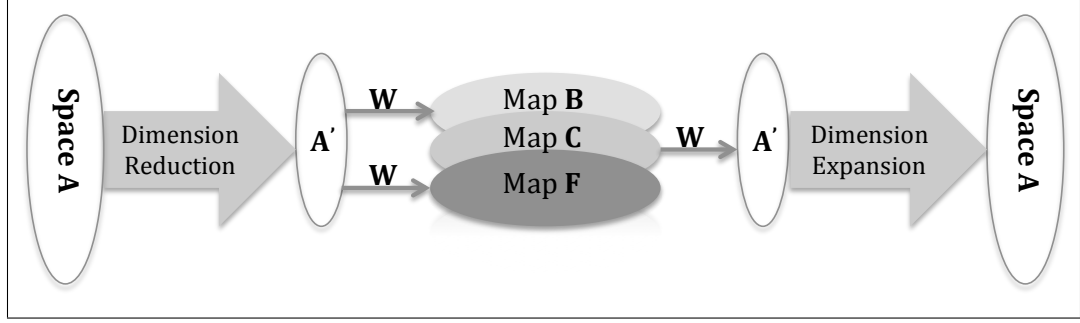


Figure 4.2: An overview of our reaching model with three neural maps at its core and dimension reduction and dimension expansion modules.

competition map **C** represent forward connections for path execution from neurons activated at time t to neurons activated at time $t+1$. Finally, in the competition map **C**, mutually inhibitory connections between nearby neurons implement a competitive network. Neurons in these maps are Radial Basis Functions (RBF) in which centers are determined by a neural weight matrix \mathbf{W} . Connections in \mathbf{W} represent the receptive fields of neurons in maps **B** and **F** from space \mathcal{A}' and are normalized ($\|\mathbf{W}_i\| = 1$). In addition, normalized connections \mathbf{W} also represent the projection fields of neurons in map **C** to space \mathcal{A}' . Because the vectors comprising \mathbf{W} are normalized, activity levels of neurons in the neural maps are inversely proportional to the Euclidean distance between the centers of the neurons and a given point in space \mathcal{A}' . We consider the weights \mathbf{W} , as well as the underlying topology of the neural maps, to represent the result of both the motor babbling itself and the prior development as determined by evolution or other developmental processes.

4.2.1 Formation of Neurally-Plausible Trajectory Bundles

As we mentioned previously, changing patterns of activity over the neurons in the neural map will represent the phase-space trajectories. A particular motor-sensory state will be represented by localized activity over these neurons, and the trajectories will be defined by shifting activities between neurons with overlapping receptive fields.

The trajectory planning happens in this new reduced space; each point in this space represents different states of these neurons.

Consider an adjacency matrix \mathbf{A} so that $A_{ij} = 1$ if neurons i and j have significantly overlapping receptive fields and otherwise $A_{ii} = 0$. We explain later the process of constructing this matrix.

The connections between successively activated neurons can encode both reverse-time correlations for path planning and forward-time correlations for path execution. These synapses will encode the \mathbf{D} field which was introduced in section 4.1.2.

In one topographic map, the $R \times R$ matrix \mathbf{B} represents the backward correlations. R is the number of neurons in the backward map. The connection strength to neuron i from neuron j (Fig. 4.3) is updated by

$$\dot{B}_{ij} = \eta_D(1 - B_{ij})r_i(t)r_j(t + \delta t) - B_{ij}/\tau_D. \quad (4.8)$$

Here, r_k is the activity of the k -th neuron, which has a certain weight vector in the neural representation of the trajectory. The adaptation rate η_D is small so that trajectory bundles evolve slowly. Another topographic map represents the $R \times R$ forward correlation matrix \mathbf{F} . This map updates as follows:

$$\dot{F}_{ij} = \eta_D(1 - F_{ij})r_i(t + \delta t)r_j(t) - F_{ij}/\tau_D. \quad (4.9)$$

We can see that $\mathbf{F} = \mathbf{B}^T$. Fig. 4.3 demonstrates two neurons, i and j , and the forward and backward connections between them. Fig. 4.4 illustrates a simplified neural representation of map \mathbf{B} , with neurons and connections among those neurons in a bundle. The connections are stronger in the center of the bundle compared to the connections in the sides.

In the last topographic map, the $R \times R$ matrix \mathbf{C} encodes the the inhibitory connections between connected neurons. This matrix implements a competitive network and is defined as

$$\mathbf{C} = -k_C \mathbf{A}. \quad (4.10)$$

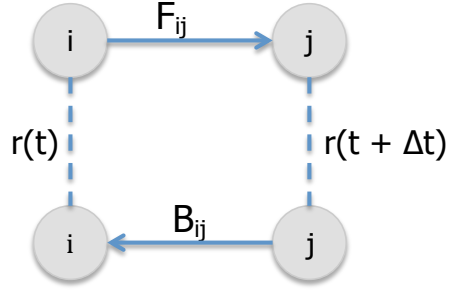


Figure 4.3: Forward and backward connections between neurons i and j .

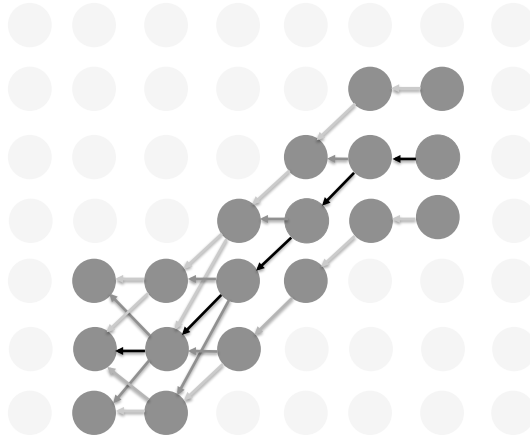


Figure 4.4: A simplified neural representation of a trajectory bundle with gray nodes inside the bundle and color of the edges representing strength of the connections among the neurons in the bundle.

4.2.2 Neurally-Plausible Trajectory Planning and Execution

After trajectory bundles are created, an agent can find a trajectory or path through the abstract motor-sensory phase space \mathcal{A} from a dynamic starting point to a goal. The goal or image of completion G initiates this process. The goal's activities spread through the network and the backward connections established in map **B**; this activation excites a subpopulation of the backward-connected neurons (see fig. 4.5.) Let γ be the activity of goal neurons and β be the backward spreading activation. Backward activation β will then be updated as

$$\dot{\beta} = \eta_B(\mathbf{B}\beta + \gamma)(1 - \beta) - \beta/\tau_B. \quad (4.11)$$

This update rule implements spreading activation, weighted according to the synapse weights in **B**. A decay term β/τ_B is included so that if the goal changes, the potential paths will quickly readjust. η_B is the rate of spread in this equation.

Path execution begins when neurons in map **C** receive input from neurons in map **B** (representing path planning) as well as neurons in map **F** (representing the current motor-sensory state). This backward activity activates corresponding forward-connected neurons to a degree $\varepsilon\beta$. At the same time, activities from the current neuron would activate those neurons to a certain degree. These neurons in map **C** compete with each other to define the next state of the planning; a neuron that was sufficiently excited by both the forward connection from the current state \mathbf{r} and the backward connection from the goal state is the winning neuron. The winning neuron would fire and define the next state of plan-execution in the forward connections. This process is formulated as

$$\dot{\chi} = \eta_C\sigma(\mathbf{C}\chi + \varepsilon\beta + \mathbf{F}\mathbf{r} - \theta_C) - \chi/\tau_C. \quad (4.12)$$

In this equation, θ_C is the activation threshold and η_C is the adaptation rate. Since we don't want the path planning to be sensitive to the slow changes of the neurons' weights, we assume $\eta_B, \eta_C \gg \eta_D$.

Path execution begins when neurons in map **C** receive input from neurons in map **B** (representing path planning) as well as neurons in map **F** (representing the current motor-sensory state). Activity in **B** activates corresponding neurons in map **C** to a degree of $\lambda\beta$. At the same time, activity of neuron r in **F** activates potential successor neurons r' and r'' in map **C**. Activated neurons in map **C** compete to define the next state of the planning; the neuron r' maximally excited by both the current state r and the backward connections from the goal state is the winning neuron. This neuron fires and defines the next motor-sensory state in \mathcal{A}' . This state is translated back from \mathcal{A}' to \mathcal{A} to generate both motor signals and sensor prediction. The winning neuron shifts to a refractory state for the rest of the planning and execution, which helps to avoid cycles in the path planning. Figure 4.5 shows the path planning and execution process.

Activity in the winning neuron is translated into motor signals by inverting the PC representations in order to project them back into \mathcal{A} . We could only project the motor subspace in the neural and PC inversion, but we want to investigate the sensory subspace prediction.

4.3 Model Implementation

We use trajectories resulting from motor babbling in three passes. The first pass trains a dimension reduction module. The second pass structures the three parallel neural maps **F**, **B**, and **C**. The third pass determines the weights of synapses within and among these neural maps. In the following sections we explain the implementation of our model, which includes multiple different techniques for the purpose of dimension reduction, different ways of neural representation of phase-space, trajectory bundle formation, and path planning and execution processes.

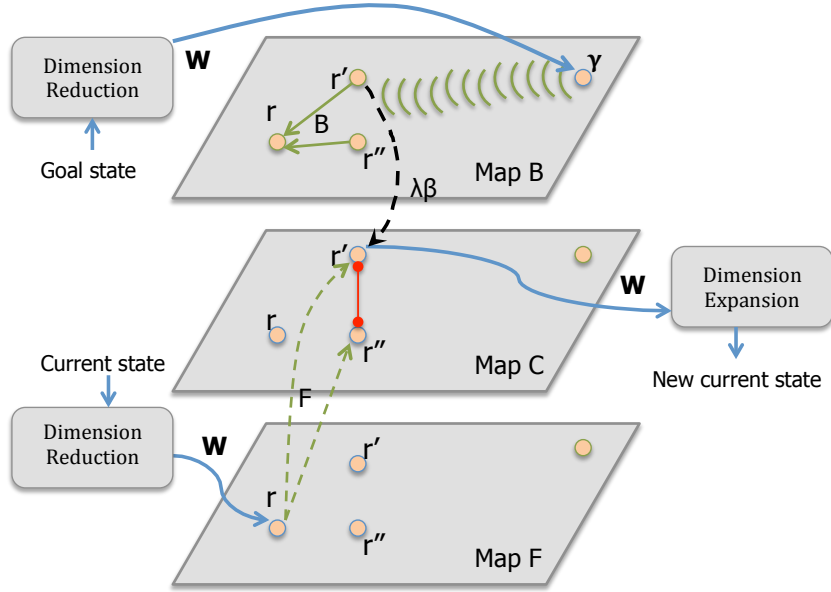


Figure 4.5: Neural architecture for implementing the path planning and execution process. Activity in map **B** spreads from the goal state γ , and the current state r in map **F** excites nearby neurons r' and r'' in map **C**. Competition among excited neurons in map **C** leads to firing of neuron r' , which represents the new motor-sensory state.

4.3.1 Implementation of Trajectory Bundle Formation

Alg. 1 describes the bundle formation process (following eq. 4.8 and eq. 4.9) which occurs in maps \mathbf{B} and \mathbf{F} through motor babbling. A babbling trajectory in the reduced space is passed as an input to this procedure. Points along the trajectory iteratively fire a set of neurons from the neural map \mathbf{B} , and reverse-time connections between firing neurons are strengthened. At iteration $i + 1$, fired neurons have their connections to the previously fired neurons from iteration i increased by weight w . In this procedure, ϕ stands for the width of bundles. By setting ϕ to a value larger than one, we can create synapses not only between the maximally firing neurons but also between neighboring neurons with a lesser level of activity. In Alg. 1, the function *calculate_weight* determines the strength of connections as a linearly decreasing function of distance from the middle of the bundles. The *update* function increases the old connection's strength by w and guarantees that the strength of connections is not above 1.0. After successfully creating map \mathbf{B} , we copy the connections from map \mathbf{B} into map \mathbf{F} with reversed direction.

One modification to the bundle formation approach to pursue further generalization is to assume there are initial connections among neurons before any learning happens. Alg. 2 shows this pre-learning stage. In this procedure, each neuron in the map is connected to d other nearby neurons based on the inverse of Euclidean distance with connections that weigh w . Here, for each neuron i in the neural map, the neighboring neurons within the threshold of d are found. Then a connection with a weight of w is set up between neuron i and a neighbor neuron j . After this initial map construction, trajectory bundles are added to the map using Alg. 1.

4.3.2 Implementation of Path Planning and Execution

Phase-space trajectories are represented by changing patterns of activity over the neurons in the neural maps. Trajectory planning occurs in the reduced space \mathcal{A}' . Alg. 3 describes the implementation of path planning and execution. In this

implementation, *start* and *goal* are single neurons, but we expect to expand to multiple neurons to represent the goal and start states. In this procedure, β initially is set to zero for all the neurons in map **B**. We iteratively update β until the end of path execution or for a certain number of steps, *max_step*. Meanwhile, χ is updated for the neighbors of current state r in map **C** where the first term, $\lambda\beta$, reflects the weight of connections from map **B**, and the second part, $\mathbf{F}[r, n]$, reflects the weight of forward connections from map **F**. The competition between nearby neurons in map **C** is computed by *argmax*. The next current state r' is added to the list *fired_neurons*, which keeps track of neurons that have been fired throughout path execution and are in their refractory state. The *transform* function projects the motor-sensory state represented by r from \mathcal{A}' back to \mathcal{A} , and from there motor commands can be sent to the arm for execution.

4.3.3 Dimension Reduction

In the following sections we will mention a few ways in that we have investigated which the spaces and processes may be represented in a way suitable for neural computation.

4.3.3.1 Principal Component Analysis

One simple way of reducing the high dimensionality is Principal Components Analysis (PCA) (Shlens, 2014); this simple non-parametric technique can extract relevant information from the dataset by identifying the most meaningful basis. PCA re-express the data as a linear combination of its basis vectors. For PCA to work properly, we need to subtract the mean from each of the data dimensions. After this normalization step, the produced data has an average of zero.

Let X be the data set with dimension $m \times n$. We calculate the *covariance* matrix for this dataset, and subsequently, *eigenvectors* and *eigenvalues* of this matrix. Now, we have n eigenvectors and n eigenvalues. The eigenvectors with the highest eigenvalues are the principal components of the dataset. In general, we sort all the

eigenvalues from highest to lowest to choose the first p eigenvectors which are the most significant. We keep the p eigenvectors and ignore the less important ones to derive the new dataset X' . Let matrix P be the most significant eigenvectors and, then the re-scaled data set is calculated by $X' = XP$.

4.3.3.2 Generalized Hebbian Algorithm

The *Generalized Hebbian Algorithm (GHA)* by Oja (1982) is a neurally-plausible technique for reducing dimension and extracting the first p principal components. This process gradually computes orthogonal basis vectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p$. Let $\boldsymbol{\alpha}(t)$ be the motor-sensory trajectories or the input samples and $Y_k = \mathbf{u}_k \cdot \boldsymbol{\alpha}$, for $k = 1, \dots, p$; where Y_k is the output of neuron k and \mathbf{u}_k is the eigenvector or the neurons' weight vector.

$$\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_p]^T, \quad (4.13)$$

$$\mathbf{Y} = \mathbf{U}\boldsymbol{\alpha}. \quad (4.14)$$

Eq. 4.15 describes the update rule for GHA where the first principal component can be discovered. In this equation, η_H is the learning rate.

$$\mathbf{u}(t+1) = \mathbf{u}(t) + \eta_H(\mathbf{Y}(t)\boldsymbol{\alpha}) \quad (4.15)$$

This equation was expanded by Sanger to discover the rest of the eigenvectors. You can see the GHA update rule in Eq. 4.16. Eq. 4.17 shows the update rule for the weight between neurons i and j .

$$\mathbf{u}_p(t+1) = \mathbf{u}_p(t) + \eta_H \mathbf{Y}(t) \left(\boldsymbol{\alpha} - \sum_{i < p} (Y_i - \boldsymbol{\alpha}) \right) \quad (4.16)$$

$$w_{ij} = w_{ij} + \eta_H (\boldsymbol{\alpha}_j y_i - y_i \sum_{k \leq i} (w_{kj} y_k)) \quad (4.17)$$

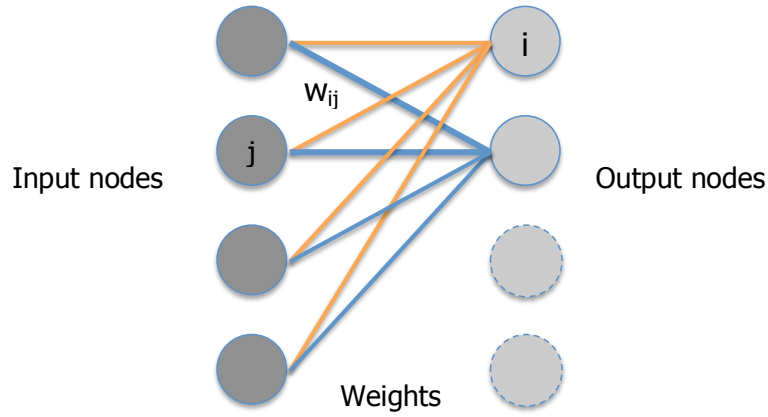


Figure 4.6: A two-layer neural network is in the core of the GHA algorithm. In this example, the output layer that captures the principal components can have up to 4 nodes. The output nodes with dashed line could be the two least significant components.

After several iterations, the \mathbf{U} matrix converges to the eigenvectors ordered by decreasing eigenvalue. We explained the process of transferring the input data into the reduced space in the section 4.3.3.1. Fig. 4.6 illustrates a small neural network with the two layers used in a GHA. In this example, the input data has 4 features, the output layer can therefore hold up to 4 nodes. In this example, the network's weights converge to the first two eigenvectors after several training passes. In general, the underlying network of a GHA consists of two layers: one input, and one output. But the size of the input layer is the same as the number of features in the dataset, and the size of the output layer is less than or equal to the number of features.

Alg. 4 shows the main function of a generalized Hebbian algorithm where the *Update* function refers to Alg. 5. Before we can find the eigenvectors using the neural network, we need to normalize the data by zeroing the mean. After normalization, samples are shuffled and passed to the *Update* function one at a time. This function updates the neural network's weights in a synchronized fashion, using the given sample starting from the weights of the first principle component. Weights of an output node are determined by the given example and all the previous output nodes. The process

of update happens multiple times for one sample throughout different iterations. After several epochs (training passes), the network has presumably converged, so we use the covariance matrix and the eigenvectors to calculate the eigenvalues. The transpose of the network's weight represents the eigenvectors. The order of output neurons represents the order of principal components in which their weights represent the corresponding eigenvector. The first neuron stands for the first principal component and so forth.

4.3.3.3 Autoencoders

Autoencoders are simple neural networks used to transform inputs into outputs with the least possible amount of loss. The goal is to make the output the same as the input in a network with a central bottleneck. Autoencoders use backpropagation to find synapse weights that encode the input in the middle layer (Hinton and Salakhutdinov, 2006). Backpropagation is a form of error-driven learning that can be implemented using a neurally plausible model as proposed by O'Reilly et al. (2012). Autoencoder is a nonlinear generalization of PCA the uses multiple layers of neural map to transform data. The data is transformed by a set of layers called the *encoder* and is reconstructed back by a another set of layers with the similar structure called the *decoder*. The learning happens by training the two networks to minimize the difference between the original data and reconstructed data. In our implementation, we train the network by starting with random weights and not from a pre-trained set of weights.

Both the inputs and outputs of the autoencoder are points in the motor-sensory space \mathcal{A} , and the bottleneck represents the same points in the reduced space \mathcal{A}' . After training, the front and back halves of the autoencoder serve as an encoder and a decoder respectively. The encoder module is used to project motor-sensory states from space \mathcal{A} to the reduced-dimensional space \mathcal{A}' . Similarly, the decoder is used to transform states from space \mathcal{A}' to space \mathcal{A} . Fig. 4.7 illustrates a general framework for an autoencoder. The input and output vector have the same number of nodes and the smallest box in the middle represents the bottleneck layer.

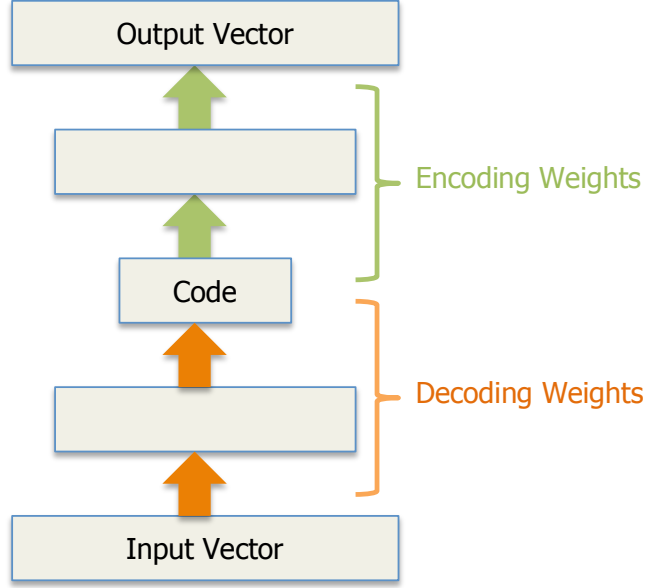


Figure 4.7: A general framework of an autoencoder. The green arrows and orange arrows respectively represent the weights of encoder and the decoder. The smallest box in the middle represents the bottleneck layer.

4.3.4 Representation of Motor-sensory Space

After transforming the babbling trajectories into a lower dimensional space, we create a set of *RBFs* that is the same for neural maps **F**, **B**, and **C**. Consider $T = \{\mathbf{a}_1, \dots, \mathbf{a}_i, \mathbf{a}_{i+1}, \dots, \mathbf{a}_f\}$, $\mathbf{a}_i \in \mathcal{A}$, a trajectory in the high-dimensional phase-space \mathcal{A} , and $T' = \{\mathbf{a}'_1, \dots, \mathbf{a}'_i, \mathbf{a}'_{i+1}, \dots, \mathbf{a}'_f\}$, $\mathbf{a}'_i \in \mathcal{A}'$, the same trajectory in the reduced space \mathcal{A}' . The j th feature of \mathbf{a}'_i is shown by a'_{ij} . We want to create a matrix **W** which serves as the receptive fields of neurons in both the neural maps **F** and **B** and the projection field of neurons in map **C**. To make each weight vector of **W** normalized ($\|\mathbf{W}_i\| = 1$), we calculate an additional pseudo-feature based on the features in \mathcal{A}' such that this pseudo-feature guarantees $\|\mathbf{W}_i\| = 1$. This means the extra feature called W_{i1} can be calculated simply by the other features as

$$W_{i1} = \sqrt{1 - \sum_{j=2}^{|\mathcal{A}'|} W_{ij}^2}. \quad (4.18)$$

We have investigated various techniques for creating such a neural map, which are explained in the following sections.

4.3.4.1 Self-Organizing Maps

The Self-Organizing Map (SOM) invented by Kohonen (2001) is used both to project a high-dimensional data space into a low-dimensional space and to cluster data so that similar data points will be mapped to nearby neurons. The SOM is used to represent the re-scaled data into a two-dimensional map. In addition, this technique creates a network that stores trajectories in such a way that any topological relationships within the motor babbling training set are maintained. It means that the SOM preserves the topology.

The SOM is a network, a 2D map of neurons where each neuron is fully connected to the input layer and holds a vector of weights with the same dimension as the input vector. The neurons are attached to nearby neurons by a neighborhood relation called the network topology. Neurons are connected to each other in a rectangular or hexagonal topology. Fig. 4.8a shows a small self-organizing map of size 5×6 neurons. Each neuron or node in this map is fully connected to the 4 nodes in the input layer. Therefore, the weight vector of each node has 4 features. In this project, we only focus on the hexagonal topology. Consider node p_1 with indices of (x_1, y_1) and another node p_2 with indices of (x_2, y_2) in a self-organizing map. The hexagonal distance between these two nodes is calculated as

$$dx = x_1 - x_2, dy = y_1 - y_2 \quad (4.19)$$

$$dist = \max(\text{abs}(dx), \text{abs}(dy)), \text{ if } \text{sign}(dx) = \text{sign}(dy) \quad (4.20)$$

$$dist = \text{abs}(dx) + \text{abs}(dy), \text{ if } \text{sign}(dx) \neq \text{sign}(dy). \quad (4.21)$$

In Eq. 4.20 and Eq. 4.21, $\text{sign}(x) = 1$ if $x \geq 0$ otherwise $\text{sign}(x) = -1$ and $\text{abs}(x)$ returns the absolute value of number x . Fig. 4.8b shows a map of 5×6 neurons with their corresponding indices. The node in location (2,2) is a center node and two

neighborhoods sizes of 1 and 2 are highlighted in colors according to the hexagonal topology and the preceding equations.

Training: In a training round of the SOM, samples are drawn randomly from the dataset and fed to the network. For each sample, the best neuron from the map will be chosen based on a similarity measure and that neuron's weight and its neighborhood will be adjusted. This process is repeated a number of times called an *epoch*.

Finding the Best Matching Unit: In this project, we use *Euclidean Distance* to find the best match for each sample. The easiest way to find the Best Matching Unit (BMU) is to calculate it with

$$Distance = \sqrt{\sum_{i=0}^{i=n} (T_i - W_i)^2}, \quad (4.22)$$

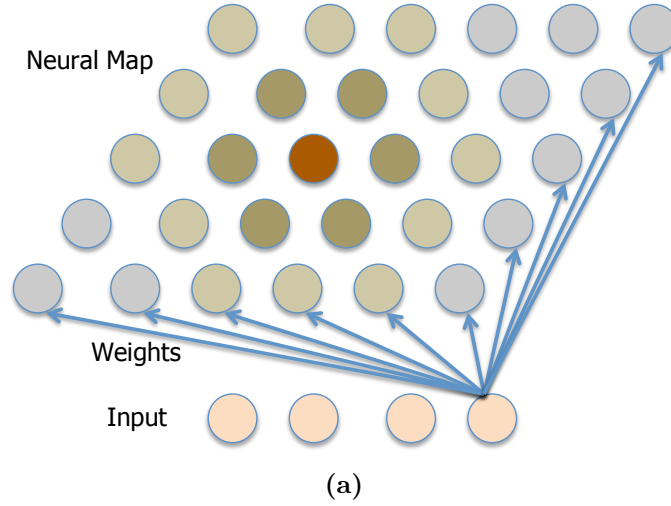
where W denotes the neuron's weight and T represents the current sample. The node with the smallest distance is the winner.

Finding neighbors: Initially, the SOM starts with a large neighborhood size or radius σ_0 . One important point of SOM is that the area of the neighborhood around the winning unit shrinks over the time. The neighborhood function must be a non-increasing function. Eq. 4.23 shows this function which is an exponential decay.

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right) \text{ for } t = 1 \text{ to } size(dataset) \quad (4.23)$$

In this equation, σ is the radius of neighborhood at time t , σ_0 is the radius of neighborhood at $t = 0$, and λ is a time constant defined in Eq. 4.24.

$$\lambda = \frac{epoch}{\sigma_0} \quad (4.24)$$



(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)

(b)

Figure 4.8: a) A small self-organizing map where each node of input is connected to all the nodes of the map. One node is colored in brown as an example along with its neighborhood in different colors. Connections are omitted to avoid clutter in this image. b) Neighborhoods of size of 1 and 2 are marked dash lines for a center node located in (2,2) in a hexagonal topology. Six nodes with indices of (1,1), (1,2), (2,1), (2,3), (3,2), (3,3) are in distance 1 of this node. Twelve nodes with indices of (0,0), (0,1), (0,2), (1,0), (1,3), (2,0), (2,4), (3,1), (3,4), (4,2), (4,3), (4,4) are within distance 2 of this node.

Updating weights: The SOM update rule for a neuron is as follows

$$W(t+1) = W(t) + \alpha(t) \times \eta(t) \times (T(t) - W(t)). \quad (4.25)$$

Where $\alpha(t)$ is the learning rate that, according to Eq. 4.26, decreases with time in an exponential fashion:

$$\alpha(t) = \alpha_0 \exp\left(-\frac{t}{\lambda}\right) \text{ for } t = 1 \text{ to } \text{size}(\text{dataset}). \quad (4.26)$$

Another important aspect of the SOM is how much the weights will be affected around the winning node. Here, we choose a Gaussian function which implies the effect of scaling is proportional to the distance of a node from the winning node (Eq. 4.27).

$$\eta(t) = \exp\left(-\frac{\text{distance}^2}{2\sigma^2(t)}\right) \text{ for } t = 1 \text{ to } \text{size}(\text{dataset}). \quad (4.27)$$

In this equation, *distance* is the distance of the Best Matching Unit (BMU) from the current neuron and σ is the radius of the neighborhood discussed earlier that shrinks over time.

Alg. 6 summarizes the self-organizing map. The learning happens through multiple passes over the training data set. In our implementation, the number of training passes or *epochs* is given as an input to the algorithm. Another convergence criterion looks at average quantization error over the input samples, defined as $E\{\|\mathbf{x} - \mathbf{m}_c(x)\|\}$ and to test whether this error is below a desired threshold value. Here, x is an input sample and m_c is a matching unit for this sample. Lastly, we can check whether the first and the second matching units for the input samples are neighbors; this test confirms that the topology was preserved.

4.3.4.2 Creating Neural Maps using Cartesian Product of Features

Another way to create neural representation of motor-sensory space is to produce the Cartesian product of the features in space \mathcal{A}' . To do so, we first calculate the

minimum, maximum and desired resolution for each feature of the motor-sensory states in the reduced space (Eq. 4.28, Eq. 4.29 and Eq. 4.30). We use these values to arrange each feature from the minimum to the maximum with the resolution as the step value (Eq. 4.31). Finally, the Cartesian product of all those ranges creates a set of weights \mathbf{W} (Eq. 4.32). Each ordered tuple of set \mathbf{W} encodes a neuron's weight or the center of an RBF; therefore the number of neurons in the neural maps is same as the number of tuples in this set. To make each tuple of \mathbf{W} normalized, ($\|\mathbf{W}_i\| = 1$), we calculate an additional feature based on the other features where this additional feature guarantees $\|\mathbf{W}_i\| = 1$.

$$min_j = \min_{i=1}^k(a'_{ij}); \text{ for } j = 1 : |\mathcal{A}'| \quad (4.28)$$

$$max_j = \max_{i=1}^k(a'_{ij}); \text{ for } j = 1 : |\mathcal{A}'| \quad (4.29)$$

$$res_j = \text{Resolution}(\{a'_{ij} | i = 1 : k\}), \text{ for } j = 1 : |\mathcal{A}'| \quad (4.30)$$

$$range_j = \{min_j, min_j + res_j, \dots, max_j\}; \text{ for } j = 1 : |\mathcal{A}'| \quad (4.31)$$

$$\mathbf{W} = range_1 \times range_2 \times \dots \times range_{|\mathcal{A}'|} \quad (4.32)$$

Fig. 4.9 shows a 3D map created by this approach. The size of reduced space in this example is 3. Each side of the cube represents one of the three features of the space \mathcal{A}' where they are arranged from min_i to max_i with the step size of res_i . There are $|range_1| \times |range_2| \times |range_3|$ small cubes in this cube where each sub cube represents a neuron in the neural map.

4.3.4.3 Creating Neural Maps using Babbling Trajectories

The approach in section 4.3.4.2 can become computationally expensive as the number of features in the reduced space increases. Furthermore, the Cartesian product creates neurons that might not ever be used in the bundle formation. Therefore, we proposed the following way to create the neural map. Assume we have a function *Resolution*

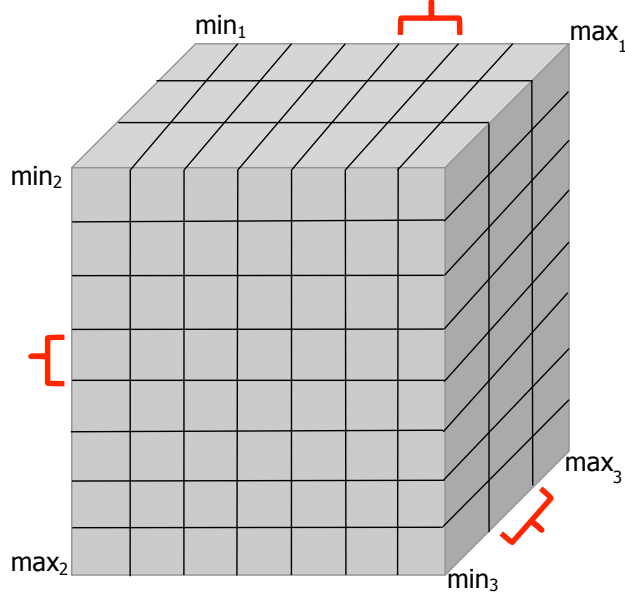


Figure 4.9: Creating a 3D neural map using Cartesian product of the three features of space \mathcal{A}' . Each side of the cube represents one of the three features that are arranged from min_i to max_i with step size of res_i .

that takes a vector of values for a given feature and returns a desired resolution for that feature, a set of intervals covering the feature's range. Then the resolution of each feature is given by

$$res_j = \text{Resolution}(\{a'_{ij} | i = 1 : k\}), \text{ for } j = 1 : |\mathcal{A}'|, \quad (4.33)$$

where k is the number of points in all babbling trajectories. Any function returning a set of intervals covering a feature's range can be used, and we discuss some examples in chapter 5.

High-resolution maps assure a smooth motion trajectory by activating different neurons for different motor-sensory points in T' . In order to efficiently store such fine-grained maps of motor-sensory space, we store only those neurons representing points in or near the trajectory bundles resulting from babbling. Each motor-sensory point \mathbf{a}'_i of the babbling data and its neighboring points as determined by function *Resolution* are assigned as the center of an RBF.

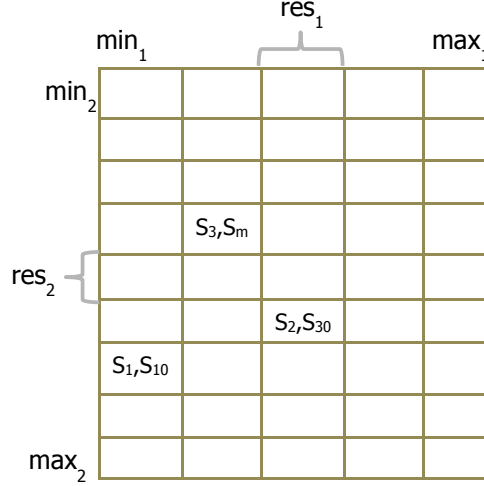


Figure 4.10: Creating a 2D neural map by arranging the first two dimensions of the space \mathcal{A}' . Each patch or small square is divided according to the samples within the boundaries of that square. s_i represents a sample of the training set.

4.3.4.4 Creating Neural Maps using Ordered Features

In the case of using PCA or GHA for dimension reduction, all the features in the space \mathcal{A}' are sorted in a descending order based on their eigenvalues. The first two features of the trajectories in the space \mathcal{A}' correspond with the two highest principal components. We create a 2D map such that two important features have the finest resolution in the map. To create a neural map, we arrange the first two features from the minimum (Eq. 4.28) to the maximum (Eq. 4.29) with a step size (Eq. 4.33). One of these ranges defines the x axis of the 2D map with m steps while the other one defines the y axis of the map with n steps. This arrangement creates a 2D map of $m \times n$ patches of RBF neurons. Each patch represents some points of the reduced space whose their first two features are within the ranges of the patch. To divide a patch into smaller patches or neurons, we iterate over all the training data in space \mathcal{A}' and locate samples that fall within the range of the patch. Using the features of the samples that are within the range a patch, we create a fixed number of neurons.

Fig. 4.10 shows a small 2D map where sides of the map represent the two important dimensions of space \mathcal{A}' . The rest of the dimensions, $|\mathcal{A}'| - 2$, are created

using the samples that are inside the patches. For example, in this map, samples s_2 , s_{30} fall into the same patch. This patch is divided to smaller patches to represent the statistics of these samples.

Algorithm 1 Trajectory bundle formation.

```

1: procedure BUNDLEFORMATION(neural_map,  $T'$ ,  $\phi$ ,  $\tau_D$ ,  $\eta_D$ )
2:    $n \leftarrow \text{find\_firing\_neurons}(T'[1], \phi)$ 
3:   for  $k \leftarrow 2$  to  $|T'|$  do  $\triangleright$  for all points along  $T'$ 
4:      $m \leftarrow \text{find\_firing\_neurons}(T'[k], \phi)$ 
5:     for  $i$  in  $n$  and  $j$  in  $m$  do
6:        $w \leftarrow \text{calculate\_weight}(i, j)$ 
7:        $\text{update}(\mathbf{B}[j, i], w, \eta_D, \tau_D)$ 
8:     end for
9:      $n \leftarrow m$ 
10:  end for
11:   $\mathbf{F} \leftarrow \mathbf{B}^T$ 
12:  return neural_map
13: end procedure

```

Algorithm 2 Initial connections in the neural map.

```
1: procedure INITIALCONNECTIONS(neural_map,  $w, d$ )
2:   for each  $i$  in neural_map do
3:      $m \leftarrow \text{find\_closest\_neurons}(i, d)$ 
4:     for  $j$  in  $m$  do
5:       if  $i = j$  then
6:         skip this node
7:       end if
8:       update( $\mathbf{B}[j, i], w$ )
9:       update( $\mathbf{B}[i, j], w$ )
10:    end for
11:  end for
12:   $\mathbf{F} \leftarrow \mathbf{B}^T$ 
13:  return neural_map
14: end procedure
```

Algorithm 3 Path planning and execution.

```
1: procedure PATHPLANNING(neural_map, start, goal,  $\eta_B, \tau_B, \lambda, \text{max\_step}$ )
2:    $\beta \leftarrow 0.0$  for all neurons in neural_map
3:    $\chi \leftarrow 0.0$  for all neurons in neural_map
4:    $r \leftarrow \text{start}$   $\triangleright$  current state
5:   fired_neurons  $\leftarrow \{r\}$ 
6:   step  $\leftarrow 0$ 
7:   while  $r \neq \text{goal}$  and step  $< \text{max\_step}$  do
8:     for each  $n$  in neural_map do
9:        $\beta \leftarrow \beta + \eta_B(\mathbf{B}\beta + \gamma)(1 - \beta) - \beta/\tau_B$ 
10:      if  $\beta > 0$  and  $\mathbf{F}[r, n] > 0$  then
11:        if  $n$  is not in fired_neurons then
12:           $\chi \leftarrow \lambda\beta + \mathbf{F}[r, n]$ 
13:        end if
14:      end if
15:    end for
16:    if  $\max(\chi) > 0$  then
17:       $r' \leftarrow \text{argmax}(\chi)$   $\triangleright$  winning neuron,  $r'$ 
18:       $r \leftarrow r'$   $\triangleright$  new current state
19:      (motor-sensory)  $\leftarrow \text{transform}(W[r])$ 
20:      add  $r$  to fired_neurons
21:    end if
22:    step  $\leftarrow \text{step} + 1$ 
23:     $\chi \leftarrow 0.0$  for all nodes
24:  end while
25: end procedure
```

Algorithm 4 Main algorithm for generalized Hebbian algorithm.

```
1: procedure GENERALIZEDHEBBIAN(data, epochs,  $\eta$ )
2:   rows, cols  $\leftarrow$  shape(data)
3:   weights  $\leftarrow$  random small values with shape(cols, cols)
4:   data  $\leftarrow$  data - mean(data)
5:   cov  $\leftarrow$  covariance(data)
6:   for  $i \leftarrow 1$  to epochs do
7:     shuffle data
8:     for  $j \leftarrow 1$  to rows do
9:       sample  $\leftarrow$  data[j,:]  $\triangleright$  all features of sample  $j$ 
10:      weights  $\leftarrow$  Update(sample, weights,  $\eta$ )
11:    end for
12:  end for
13:  eigen_vectors  $\leftarrow$  transpose(weights)
14:  eigen_values  $\leftarrow$  mean((cov  $\times$  eigen_vectors) / eigen_vectors)
15:  return eigen_vectors, eigen_values
16: end procedure
```

Algorithm 5 Update function for generalized hebbian algorithm.

```
1: procedure UPDATE(sample, weights,  $\eta$ )
2:   cols  $\leftarrow$  size(sample) ▷ number of features in the sample
3:   Y  $\leftarrow$  zeros(cols) ▷ vector of cols elements
4:   delta_weights  $\leftarrow$  zeros(shape(weights)) ▷ Initialize it to zeros
5:   Y[0]  $\leftarrow$  weights[0, :]  $\odot$  sample ▷ inner product multiplication
6:   delta_weights[0, :]  $\leftarrow \eta \times$  (sample - (Y[0]  $\times$  weights[0, :]))
7:   for  $i \leftarrow 1$  to cols do
8:     Y[i]  $\leftarrow$  weights[i, :]  $\odot$  sample ▷ inner product multiplication
9:     temp  $\leftarrow$  zeros(shape(weights[i, :]))
10:    for  $j \leftarrow 1$  to  $i + 1$  do
11:      temp  $\leftarrow$  temp + Y[j]  $\times$  weights[j, :]
12:    end for
13:    delta_weights[i, :]  $\leftarrow \eta \times$  Y[j]  $\times$  (sample - temp)
14:  end for
15:  weights  $\leftarrow$  weights + delta_weights
16:  return weights
17: end procedure
```

Algorithm 6 Main algorithm for SOM

```
1: procedure SELFORGANIZINGMAP(data, map_height , map_width , epochs,
   initial_learning_rate)
2:   map_size  $\leftarrow$  map_width  $\times$  map_height
3:   nodes_weight  $\leftarrow$  random small values
4:   initial_neighborhood_radius  $\leftarrow \frac{\max(\text{map\_height}, \text{map\_width})}{2}$ 
5:   time_constant  $\leftarrow \frac{\text{epoch}}{\log(\text{initial\_neighborhood\_radius})}$ 
6:   for  $i \leftarrow 1$  to epochs do
7:     neighborhood_radius  $\leftarrow$  initial_neighborhood_radius  $\times \exp(\frac{i}{\text{time\_constant}})$ 
8:     learning_rate  $\leftarrow$  initial_learning_rate  $\times \exp(-(\frac{i}{\text{time\_constant}}))$ 
9:     shuffle data
10:    for  $j \leftarrow 1$  to size(data) do
11:      sample  $\leftarrow$  data[j]
12:      find the BMU for sample  $j$  using eq. 4.22
13:      for  $l \leftarrow 1$  to map_size do
14:        distance  $\leftarrow$  distance( $l$ , BMU) using eq. 4.22
15:        if distance < neighborhood_radius then
16:          influence  $\leftarrow \exp(-\frac{\text{distance}^2}{2 \times \text{neighborhood\_radius}^2})$ 
17:          nodes_weight += influence  $\times$  learning_rate  $\times$  (sample -
            nodes_weight)
18:        end if
19:      end for
20:    end for
21:  end for
22:  return nodes_weight
23: end procedure
```

Chapter 5

Experiments and Results

In this chapter, we begin with a description of the humanoid robot used for all of our experiments. In section 5.2 we then describe some motor exploration or motor babbling techniques we have tried, and in particular the method upon which the rest of the experiments are based. In section 5.3, we then describe a set of experiments with a random start and end point, which aims to find a right dimension reduction approach and a suitable construction for a 2D neural map for our model. In section 5.4, we list some evaluation metrics utilized for testing the quality of planned motions. In section 5.5, then, we introduce a new set of experiments with a fixed start point, and through these experiments, we investigate the effects of bundle width, training size, and the dimension of the reduced space on the accuracy of the planned motions. Next, in section 5.6 we explain another set of experiments with multiple fixed start positions. Here, we examine some modifications of bundle formation and their impacts on the planned motions. We also evaluate how different resolutions of the neural map effect the accuracy of these planned movements. Finally, in section 5.7, we review and further analyze our findings.



Figure 5.1: Rosie, Humanoid Robot.

5.1 Humanoid Robot

The robot used in our experiments is a Meka Robotics M3 mobile humanoid called Rosie (Fig. 5.1), which is a humanoid robot with two 7-DOF arms attached to a 0-DOF torso. Two 6-DOF end effectors or hands, with four fingers each, are attached to the arms. Two 6-DOF force and torque sensors are mounted on the end of each arm's compliant manipulator. The torso is connected to a Zlift, a linear actuator that mounts on the top plate of the omni-base and allows the robot's upper body to traverse a large vertical distance. The omni-base utilizes a Holomni Powered Caster to provide omnidirectional capabilities. A 2-DOF neck connects the head to the body. Two 3D Prime Sense cameras, one Kinect and one Bumble Bee camera, are mounted on the head.

5.2 Motor Babbling

In developmental robotics, motor babbling is defined as the random exploration of motor space, but there is no unified opinion on what level the arbitrary motor commands should be issued. Moreover, the exact process of motor babbling in infants is not known, and even mapping the process to robot babbling is not well defined. To choose a proper approach for our model, we examined three different methods of motor exploration. Also, to avoid damaging the robot during these experiments, we issued the motor commands at the position and not the torque level.

The first approach to simulate infant motor babbling was to increment (decrement) random joint positions of the arm by focusing on one joint at a time. Table 5.1 shows the ranges of 7 joint positions in the left arm (in radians). Focusing on the first joint of the shoulder, we randomly added/subtracted the current joint position to/from a hard-coded small value; we then commanded this new joint position to the robot's arm while the positions of other joints were fixed during the motor exploration. We observed that this random movement produces short and discrete movements which didn't resemble an infant's motor babbling stage. We also simultaneously explored multiple joints of the shoulder while the other joints were fixed. This exploration strategy also produced short and very discrete movements that could not be used for later path planning in our model. We need to mention that, considering the range of each joint, it is not computationally feasible to freely explore the entire joint space of the arm. These problems suggested that exploration in the joint space is not suitable for testing our model. Therefore, we focused on motor exploration in the Cartesian space instead.

To explore in the Cartesian space, we defined a safe working space in the shape of a box which is assumed to be in front of the robot. Due to the lack of a built-in collision avoidance system, this abstract box or safe working space is defined to minimize any possible damage to the robot. In the second approach, the end-effector is only able to explore random points inside this working area. We generated a

Table 5.1: Range of joint positions of the left arm.

Joint Number	Joint Range in Radians
0	[-1.3962634016, 3.49065850399]
1	[-0.418879020479, 2.61799387799]
2	[-1.4835298642, 1.4835298642]
3	[-0.00486908464063, 2.31810267927]
4	[-0.343829862643, 3.48542251623]
5	[-0.343829862643, 1.02677719895]
6	[-1.0471975512, 1.0471975512]

list of random end-effector target positions, (x, y, z) , within this confined space and calculated the desired commands using Rosie’s Inverse Kinematic package. We then issued the commands to Rosie’s arm to move it from the current position to the target position. This method produced long reaching trajectories that were more suitable for the rest of our experiments, but there are no intermediate motor actions between the start of a motion and the end. This issue motivated us to use third-party software that produces intermediate motor actions to perform motor babbling.

In the third method, similar to the previous one, a list of random end-effector target positions (x, y, z) was generated within the confined space. We then iterated through this list and planned/executed the motions with “MoveIt!,” which is built by [Sucan and Chitta \(2014\)](#). This final approach generated long arm trajectories along with the intermediate motor actions that are necessary for our model. Therefore, for the rest of the experiments in this chapter we used the last approach for motor babbling. Table 5.2 summarizes the different strategies that we have tried.

5.3 Random Start Points and End Points

In this section, we explain a set of experiments that allows us to both compare different dimension reduction techniques and to investigate various ways of constructing a 2D neural map for implementing our model. In this set of experiments, both the start

Table 5.2: Motor babbling strategies.

Approach	Trajectory Smoothness	Intermediate Points
Joint exploration	No	Yes
End-effector exploration using Meka’s Inverse Kinematic	Yes	No
End-effector exploration using “MoveIt!”	Yes	Yes

and end positions of each trajectory are selected randomly. Here, we first generate some random end-effector points (x, y, z) located within the boundaries of a safe area and then use the “MoveIt!” package to plan/execute arm trajectories after the provided paths are found. Fig. 5.2 shows a sequence of points sequentially planned and executed within this box, where the end of one trajectory is the start of the next one. This conceptual figure doesn’t show the exact number of tried target points, however the boundaries of this box are defined as $x = [0.6m, 0.9m]$, $y = [0.3m, 0.6m]$, and $z = [1.0m, 1.5m]$ in the robot’s coordinate system.

During the motor exploration phase, the communication with the robot is placed through the Robot Operating System (ROS). The zlift height was set to 640 during the experiment sessions. For this experiment, we only use the left arm of Rosie, where the motor commands are the joint positions and joint velocities of the left arm. The used sensors are, the joints’ positions, velocities, and force, and also the 6-axis force-torque. This force-torque sensor is mounted on the left arm’s wrist. The frame of reference for the movements is the world, which is a point located behind the robot in the omni-base. Tables 5.3 and 5.4 respectively show the sensor and motor values used in this set of experiments.

While the arm is moving from a start to an end point, we store all of the motor and sensor values along with their time stamps, which are received through the channels of the Robot Operating System (ROS). We only need to store these values when a

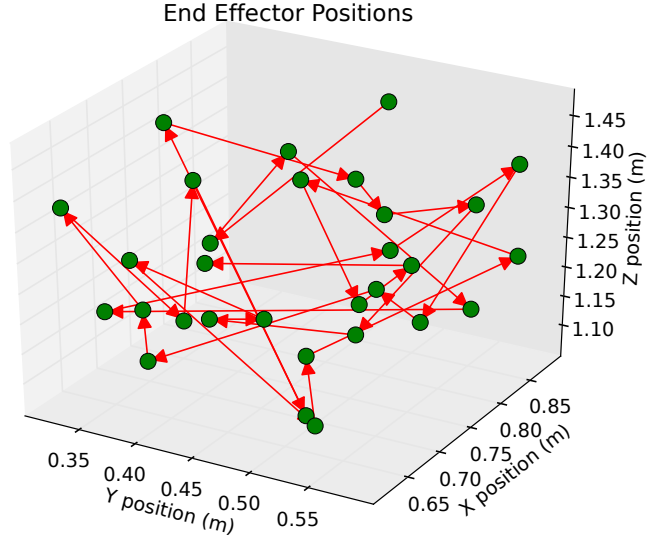


Figure 5.2: Random trajectory babbling inside a box (safe area) in front of Rosie.

Table 5.3: Sensor values.

Sensor Type	
Joint position	Left arm (for 7 Joints)
Joint effort	Left arm (for 7 Joints)
Joint velocity	Left arm (for 7 Joints)
Force and Torque sensor	Force (x, y, z) -Torque (x, y, z)

Table 5.4: Motor values.

Motor Type	
Joint position	Left arm (for 7 joints)
Joint velocity	Left arm (for 7 joints)

path from the current position to the goal position is determined. Otherwise, another target pose is submitted to “MoveIt!”. We need to mention that these channels might have different frequency rates, and the sensor distribution may not be uniform. After

Table 5.5: Motors and sensors during the motor babbling creates a motor-sensory trajectory.

Time step	Sensor values	Motor Values	Motor-Sensory
t=0	$S_0 = \text{start}$	$M_0 = \text{Stay still}$	$M_0 S_1$
t=1	$S_1 = S_0$	M_1	$M_1 S_2$
t=2	S_2	M_2	$M_2 S_3$
t=3	S_3	M_3	$M_3 S_4$
t=4	S_4	M_4	$M_4 S_5$
t=5	S_5	$M_5 = \text{Stay still}$	$M_5 S_6$
t=6	$S_6 = \text{goal}$	$M_6 = \text{Stay still}$	$M_6 S_7$
t=7	$S_7 = \text{new start}$	M_7	$M_7 S_8$
t=8	S_8	M_8	$M_8 S_9$
t=9	S_9	M_9	$M_9 S_{10}$
t=10	S_{10}	M_{10}	$M_{10} S_{11}$
t=11	S_{11}	$M_{11} = \text{Stay still}$	$M_{11} S_{12}$
t=12	$S_{12} = \text{goal}$	$M_{12} = \text{Stay still}$	$M_{12} S_{13}$

the arm stops at the goal, we repeat the process for new target positions and issue another random target position within the safe area. Table 5.5 shows how motor babbling creates two motor-sensory trajectories with two start and goal pairs. To create a motor-sensory trajectory, we combine the motor commands sent at time-stamp t with the sensor values read at time-stamp $t + 1$. At the beginning and the end of a trajectory, a command called “stay still” is issued. This command reads the current joints’ positions from the sensors and sends them to the left arm. This command ensures both the start and the target states are also preserved in each motor-sensory trajectory. Since all of the sensor and motor values are recorded into separate files throughout the experiment they need to be merged into one file to create the motor-sensory trajectories.

In the first phase of training using babbling trajectories, we then use the Generalized Hebbian Algorithm (GHA) or Principal Component Analysis (PCA) to transfer the motor-sensory trajectories to a lower dimensional space. As we mentioned in the previous chapter, the sensor and motor values must be normalized to the range

of $[-1, 1]$ before the dimension reduction stage. Alg. 4 in chapter 4 shows the offline GHA, where “offline” in this context means that the neural network is trained with the training data collected over all the sessions at once after the last motor babbling session. We compared the resulting eigenvalues and eigenvectors from the GHA with those from the PCA. To determine the size of the reduced space, we pick the first few principal components and transform the trajectories into the new space. To find the needed eigenvectors, we sort them by their corresponding eigenvalues in decreasing order, calculate their sum eigenvalues, and then normalize them. Finally, we pick the eigenvectors which capture the most variance.

The second phase of training is to construct a 2D neural map using the rescaled trajectories. Each neuron in this map contains a weight vector with the same number of features as the reduced space. At this stage, there is no connections between the neurons until the trajectory bundle formation stage (the third phase of training, Alg. 1 from chapter 4,) constructs the connections.

5.3.1 Dimension Reduction

In one experiment with dimension reduction, results of GHA were compared with PCA. We implemented GHA in C++ and OpenMP library to support multi-threading. At each epoch, the training set was divided among multiple threads. Each thread assigned to a chunk of samples iterated over them and updated the eigenvectors. The training set was composed of 5,000 babbling trajectories, using 147,279 samples overall, where each sample has 41 features. For GHA, the learning rate was 0.01 and the epochs were 1000. Calculation of eigenvalues was not originally part of the GHA, but it can be achieved using the covariance matrix and the eigenvector. Fig. 5.3a and Fig. 5.3b respectively show the first and the second eigenvalues which are calculated with both PCA and GHA. Both graphs show that the eigenvalues calculated by the GHA are not stable, and they could drastically deviate from the true values in some of the epochs. The first two true eigenvalues that are gained from PCA are 7.024,

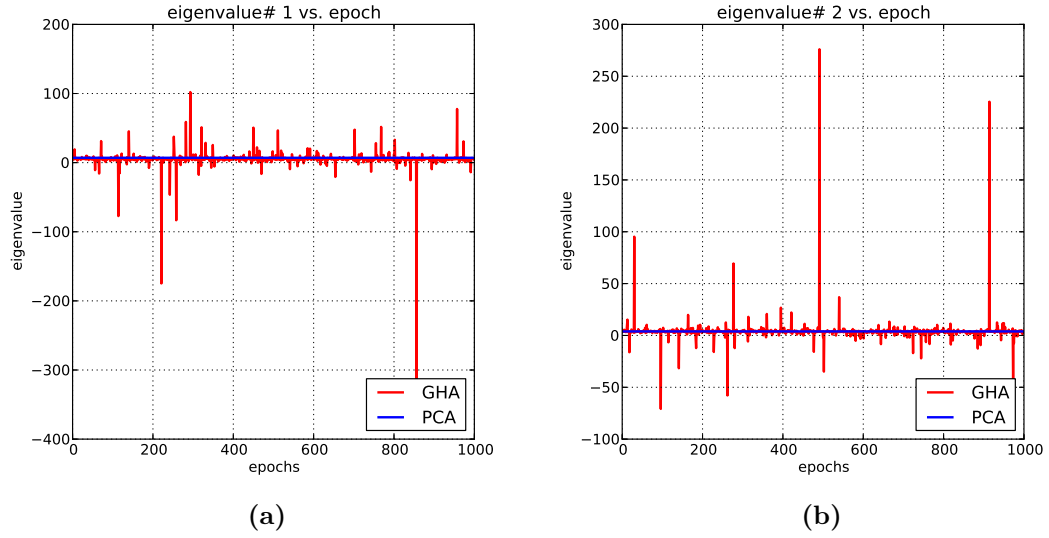
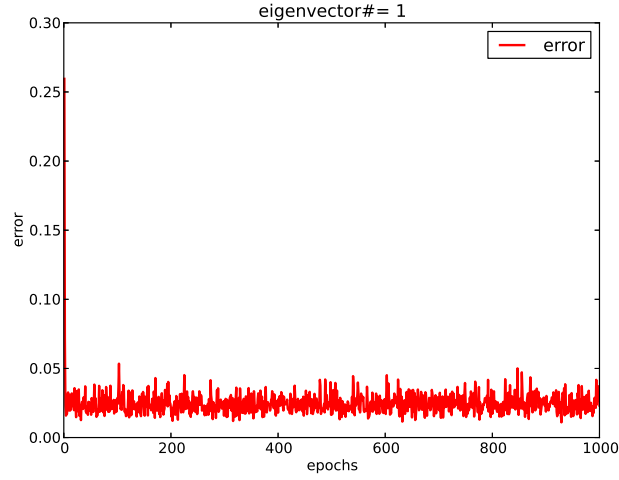


Figure 5.3: (a) The first and (b) the second eigenvalues resulting from PCA and GHA.

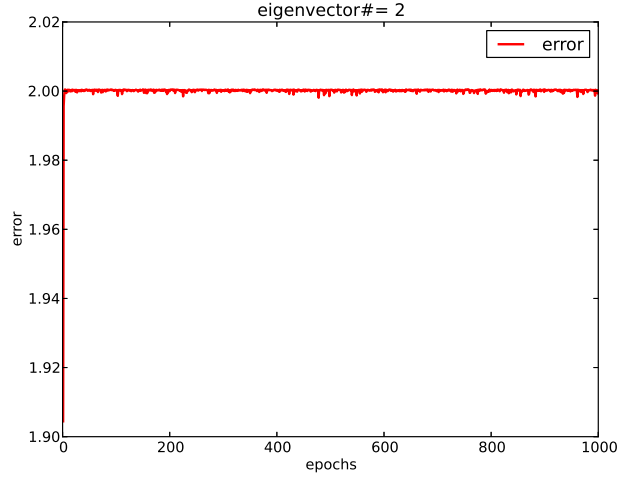
and 3.88, but in these graphs they look rather small and almost zero. This drastic change in the eigenvalues is even worse for the second eigenvalue, which suggests that even a slight change in the eigenvector from one epoch can drastically impact the eigenvalues in GHA.

Fig. 5.4a shows the difference between the first eigenvector calculated with GHA and the first eigenvector calculated with PCA as time increases, measuring the error with Euclidean distance between the two eigenvectors. This value nears, but, never reaches zero. Similarly, Fig. 5.4b shows the error of the second eigenvector of GHA over time. The error value of 2.0 implies that the second eigenvector of GHA has opposite sign of the eigenvector of PCA. Although we have used a multi-threading implementation of GHA, this algorithm could take hours to produce the desired number of principal components. Moreover, since the calculation of eigenvalues in GHA is not reliable, we use the results of PCA instead of GHA for the following experiments.

In the next experiment, we evaluated the results of PCA using *variance* by varying the size of the training data, which is a method of determining the percentage of



(a)



(b)

Figure 5.4: (a) The y axis shows the difference between the (a) first and (b) second eigenvectors calculated by PCA and the ones calculated by GHA.

variance preserved after reducing the dimensionality. To measure the number of features that are needed for \mathcal{A}' space, we varied both the size of the space \mathcal{A} and the size of the training data set. To determine the desirable size of the reduced space, we plot the cumulative sum of eigenvalues (representing variance) in descending order and then divide each eigenvalue by the total sum of eigenvalues. This plot shows the fraction of total variance retained versus the number of eigenvalues, and it allows us

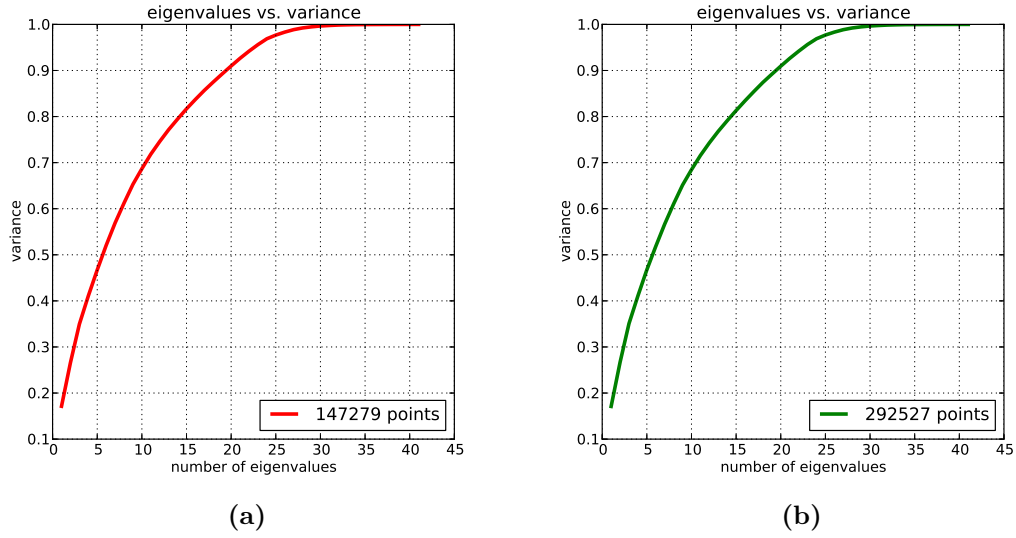


Figure 5.5: Variance versus number of eigenvalues after applying PCA to (a) a training set of 5,000 babbling trajectories with 41 features and (b) a training set of 10,000 babbling trajectories with 41 features.

to determine the number of eigenvalues that should be kept to maintain a desired level of variance. For instance, in Fig. 5.5a, we can see that for a variance of 80%, a total of 15 eigenvalues out of 41 should be considered in the space \mathcal{A}' , and for a variance of 95%, almost 23 eigenvalues should be retained. We then doubled the size of training data from 5,000 to 10,000 babbling trajectories and observed that having a larger training set didn't reduce the suitable dimension of the \mathcal{A}' space. Fig. 5.5b shows that for a variance of 95%, almost 23 eigenvalues must be preserved even though the size of the training data was doubled.

In another experiment, we repeated the previous test by removing some of the features of the training set while keeping some of the necessary features. Here, we looked at the suitable dimension of the reduced space \mathcal{A}' given 20 features of the \mathcal{A} space. The 20 features are the position and velocity of 5 joints of the left arm both for motor and sensor values. The five joints include 3 joints in the shoulder and 2 joints in the elbow joints. Fig. 5.6a shows that 9 components out of the 20 features should be retained for a variance of 95%. We also increased the size of the training

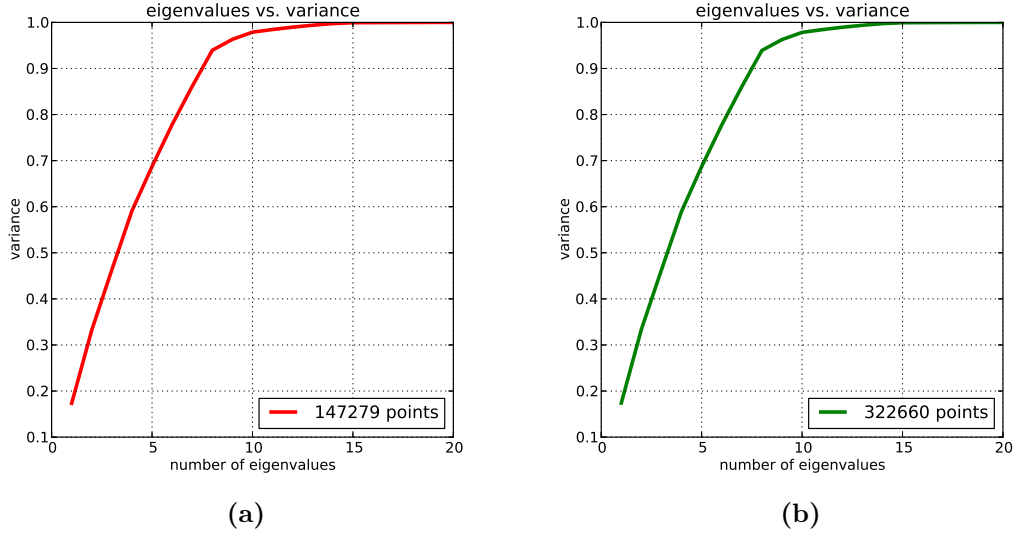


Figure 5.6: Variance versus number of eigenvalues after applying PCA to (a) a training set of 5,000 babbling trajectories with 20 features and (b) a training set of 10,000 babbling trajectories with 20 features.

set while retaining only these 20 features. This test also confirms that doubling the number of babbling trajectories has not helped PCA, and we still must preserve 9 components (Fig. 5.6b).

In the final experiment with PCA, we measured the dimension of the reduced space given only 10 features of the \mathcal{A} space, which are the position and velocity of 5 joints in the left arm for the motor commands. These five joints include the 3 joints in the shoulder and 2 joints in the elbow. Fig. 5.7a shows that 8 out of the 10 features should be retained for a variance of 95%. This experiment also confirms that doubling the number of babbling trajectories has not had any impact on the number of features left in the reduced space (Fig. 5.7b).

5.3.2 Self-Organizing Maps

In this section we examine SOM as a technique to construct a 2D neural map for implementing our model. In the experiment with SOM, the training set was 5,000 babbling trajectories with 8 features. Re-scaling the trajectories from the space \mathcal{A}

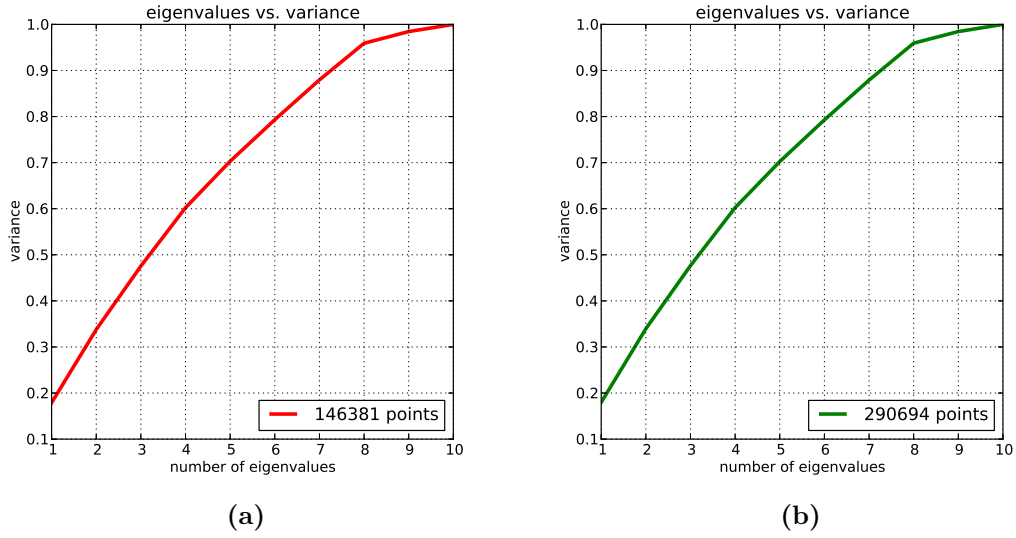


Figure 5.7: Variance versus number of eigenvalues after applying PCA to (a) a training set of 5,000 babbling trajectories with 10 features and (b) a training set of 10,000 babbling trajectories with 10 features.

with 20 features to the reduced space of size 8 was done using Principal Component Analysis. After reducing the dimension of the data set, we used SOM to represent the re-scaled babbling trajectories into a two-dimensional map. We implemented SOM in C++ and OpenMP to facilitate multi-threading. At each epoch, the training set was divided among multiple threads. Each thread assigned to a subset of samples iterates over them and updates the best-matched unit and its neighbors. The learning can be performed in batch or online.

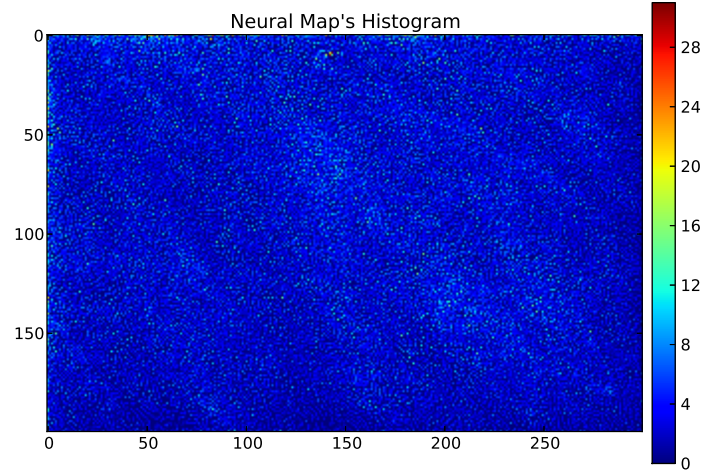
In this experiment, a neural map of size of 200×300 was initialized to random weights and trained online using learning rate of 0.2 and 1000 as the number of epochs. After the neural map was trained, a 2D histogram was plotted by iterating over the babbling trajectories (Fig. 5.8a). For each point in the data, the most active neuron was determined and the counter of that neuron was incremented by one. This histogram was shown by a 2D color map: each of its pixels represents a neuron of the neural map, and the color of that pixel depicts the number of times the corresponding neuron was fired for points in the babbling trajectories. For example, the dark blue

Table 5.6: Sequence of neurons fired for two sample trajectories in a 200×300 neural map.

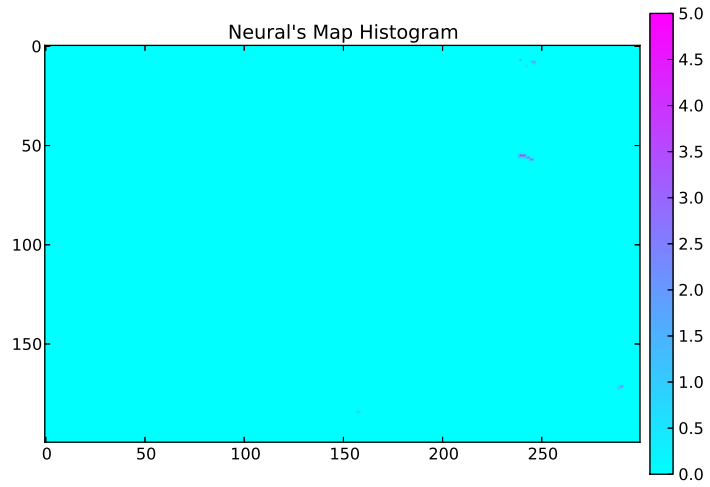
Trajectory	Neurons' ID
1	51888 51889 51589 51590 51590 51590 55358 55357 17038 16738 16739 16739 16739 16739 16739 16740 16740 16740 17040 16741 16741 16741 16741 16741 17042 17042 17043 17043 17043 17344 17344 17344 17345 17345 17345 2645 2646 2646 2645 2946 2339 2339 3242
2	11700 11701 11701 11701 11701 11701 27997 27997 27998 27697 27697 27998 27697 27697 27697 27697 27697 27697 17812 17812 17812

in this plot shows that a neuron was not fired maximally for any points, or in other words, it was created but never used in the neural map. The color bar next to the histogram is a guideline for interpreting the colors of pixels.

To gain insight into the behavior of the self-organizing map, we recorded the sequence of firing neurons as points along a trajectory. Fig. 5.8b visualizes a histogram for one trajectory where the color bar shows how motor-sensory points in that trajectory are mapped to neurons in the neural map. Here, a light blue color means no activity while the darker blue indicates more activity. We can see that the neurons in different parts of the map were fired for this trajectory and not all the firing neurons were neighbors. We also observed that the nearby points in one trajectory correspond with only one neuron, indicating that the neural map could not discriminate nearby points. This lack of discrimination in the neural map created discrete and jerky motions. This observation aligns with the fact the SOM is widely used in clustering and generalizing tasks rather than separating data points. Table 5.6 shows two sample trajectories along with the corresponding activated neurons for the points along those trajectories. Multiple sequences of neurons have been coded in red or blue for the sake of clarity only. From this table, we can see that a sequence of different points in one trajectory was mapped to the same neuron.



(a)



(b)

Figure 5.8: (a) A 2D histogram for neural map of 200×300 neurons (a) for the entire babbling data set and (b) for one babbling trajectory.

Table 5.7: Sequence of neurons fired for sample trajectories in a 350×400 neural map.

Trajectory	Neurons' ID
1	66293 66293 66293 66293 66694 66694 102166 102165 102965 103765 103764 103764 103764 103764 104163 104162 104161 104161 104161 104160 104560 104560 104560 104959 105360 104958 104958 105357 105356 105757 105756 105756 105354 105753 105753 106553 106151 105750 106950 105746 102536 102134 102534 100315 100716
2	100716 101118 101118 101518 101518 101518 116413 116413 116413 116413 116413 116413 116412 116411 116409 116409 116409 116408 116405 116405 115601

In another test, we examined the map discrimination by adding more neurons to the map to answer this question: would a self-organizing map perform better in our experiments by having access to more neurons? In this test, we increased the neural map from 60,000 neurons to 140,000 neurons. Here, we trained a map of 350×400 neurons with a learning rate of 0.2 and 600 epochs. Table 5.7 shows the same sample trajectories as in table 5.6 along with the sequence of neurons that are fired for these samples. We see that lack of discrimination even persists with a larger neural map. This test confirms that a self-organizing map is not a suitable technique for motor-sensory representation in our experiments because this lack of discrimination leads to discrete and jerky movements.

5.3.3 Manually Constructing a Neural Map

In this section, we evaluate another proposed method to manually create a neural map for motor-sensory representation. In one test, we created a neural map following section 4.3.4.4. The training set was 5,000 babbling trajectories with 41 features. These trajectories were re-scaled from space \mathcal{A} with 41 features to the reduced space of size 15 using Principal Component Analysis. We arranged the first and

the second features of the data in the reduced space from *min* to *max* values where the resolution was the average difference between the consecutive points of babbling trajectories. This arrangement creates a 2D map with 150 rows and 154 columns, hence 150×154 patches. Each patch represents samples of the training set falling within the boundaries of that patch. Each patch was expanded to more neurons using the statistics of the training samples falling within the boundaries of the patch. We created 3 neurons per each sample to represent those samples. Fig. 5.9 shows a 2D histogram for the map of 150×154 patches. In this figure, a dark blue pixel indicates that the corresponding patch was not activated for any points in the motor-sensory space. We observed that the neurons that residing in the four corners of the histogram were not employed in the motor-sensory space; in other words, these patches (neurons) were useless. Another interesting observation is the diamond shape of the histogram and the fact the neurons in the center are activated more often than the ones on the edges of the diamond. This histogram suggests that pruning the neural map in the four corners could effectively reduce future computation.

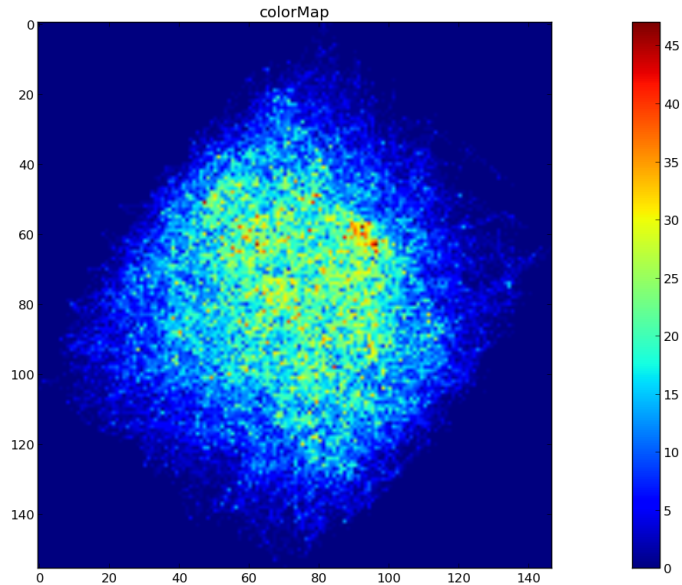


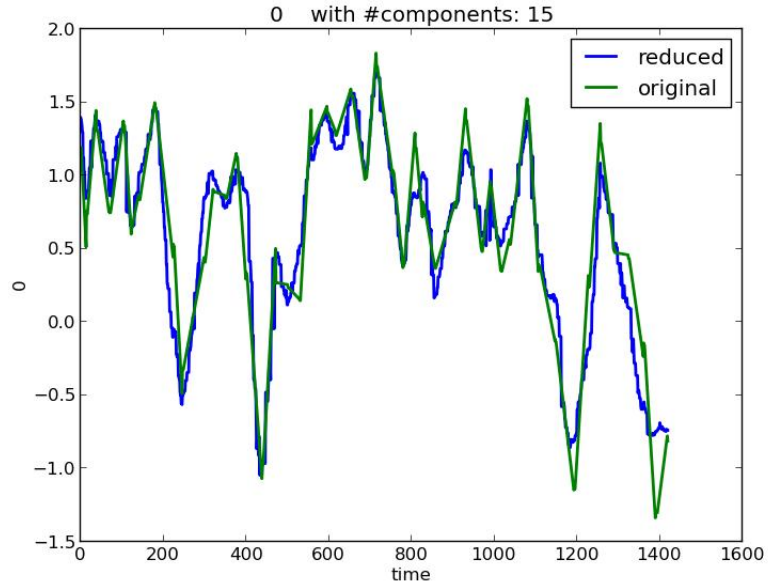
Figure 5.9: A 2D histogram for a 150×150 neural map.

Besides the 2D histogram, another visualization technique to evaluate the neural maps and dimension reduction is to compare the original trajectories with the reconstructed ones. The first test is to inspect test trajectories after expansion from space \mathcal{A}' to space \mathcal{A} . In the process of “ $data \rightarrow PCA \rightarrow PCA^{-1} \rightarrow reconstructed_data$ ”, a training or test trajectory is transferred to the reduced space using PCA ; then it is projected back to the original space using PCA^{-1} . Fig. 5.10 compares a sequence of original test trajectories (before PCA) with their reconstructed ones (after PCA^{-1}). Fig. 5.10a and Fig. 5.10b respectively show the first and the second joint position of the shoulder (two features out of 41). The original space has 41 features and size of space \mathcal{A}' is 15.

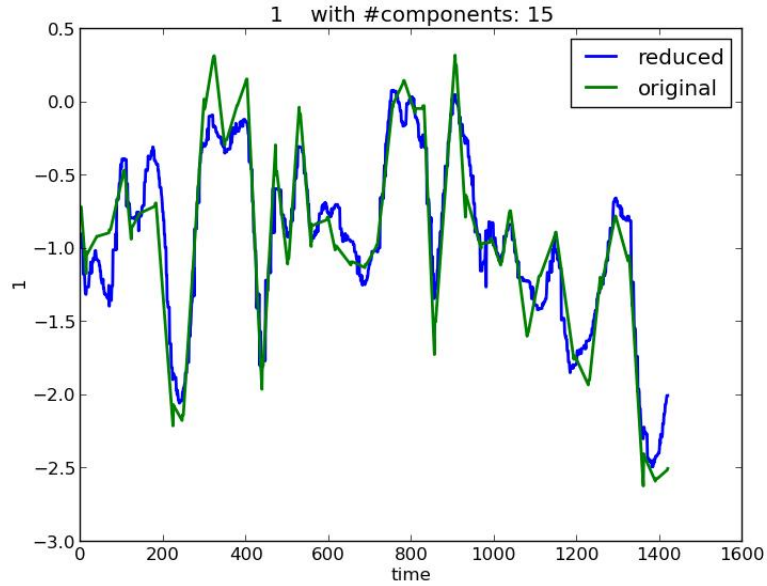
In the next test, each point of a trajectory is mapped to the most active neurons in the neural map after projection to the reduced space. This time, the weights of the firing neurons are projected back to the space \mathcal{A} to reconstruct the trajectories. This test can be summarized by the process “ $data \rightarrow PCA \rightarrow neural_map \rightarrow PCA^{-1} \rightarrow reconstructed_data$ ”. Fig. 5.11 and Fig. 5.12 respectively show a seen and an unseen sequence of trajectories reconstructed with the aforementioned test. Both graphs confirm that the neural map has failed to retain the smoothness of motions, and the reconstructed trajectories have become rather jerky. The neural map is not able to represent the details of the seen trajectories. This problem is even more severe for the unseen (test) trajectories.

Together both graphs show that this approach for creating neural maps is not adequate for our experiments. The issue with this method is that we are trying to project 15 dimensions of space \mathcal{A}' to a 2D map while only keeping an excellent resolution for the first two components. The other 13 features of the reduced space are not well represented in the 2D map. We even increased the size of the neural map by allowing each patch to be divided to more neurons, but the problem of resolution was not resolved.

The results of this section reveal that this experiment is not suitable for testing planned motions in our model. We suspect that this reaching task needs more

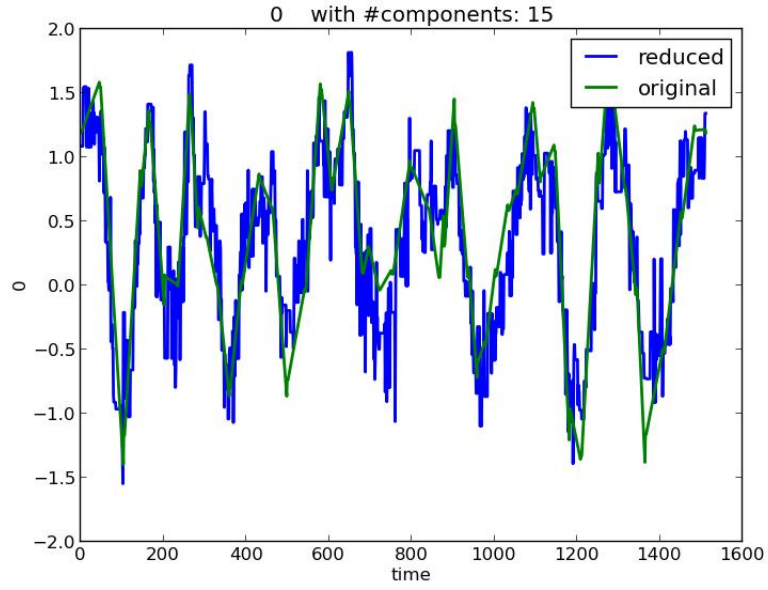


(a)

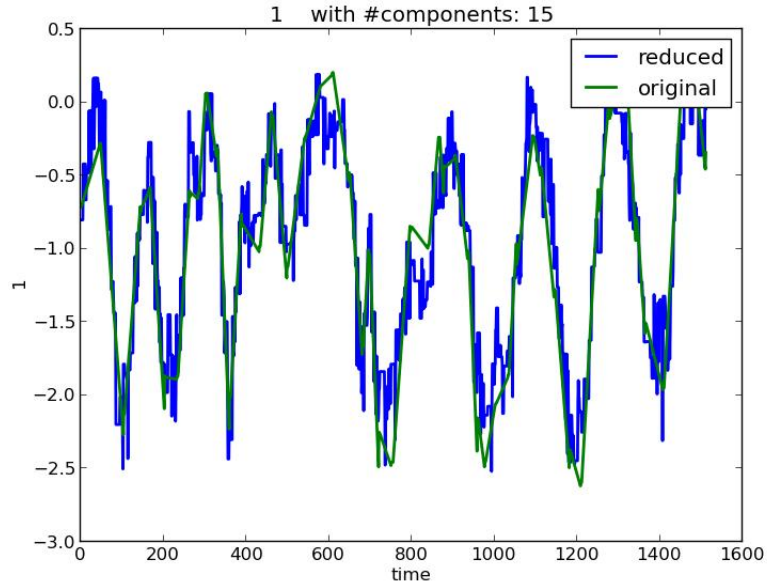


(b)

Figure 5.10: (a) A sequence of the (a) first and (b) second joint of the shoulder's positions where various seen arm trajectories are chained one after another. The green line represents the original seen trajectory and the blue line represents the same trajectory after the process of $data \rightarrow PCA \rightarrow PCA^{-1} \rightarrow reconstructed_data$.

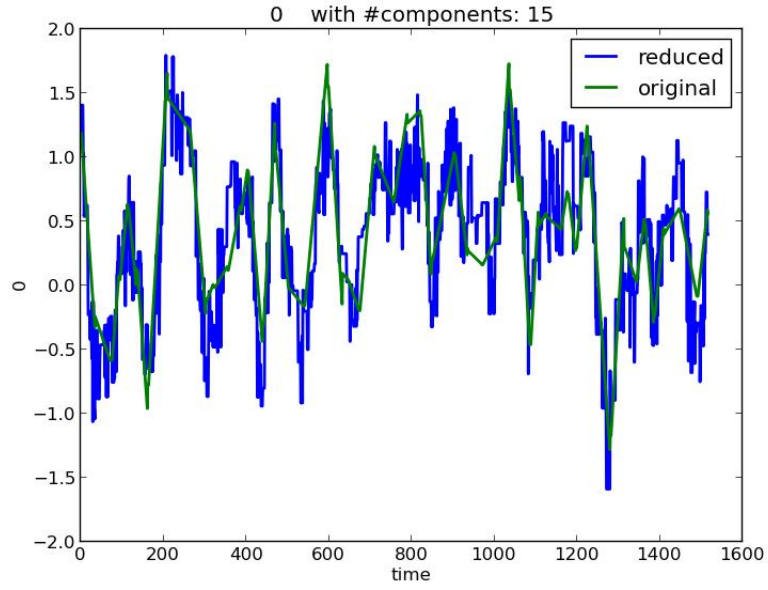


(a)

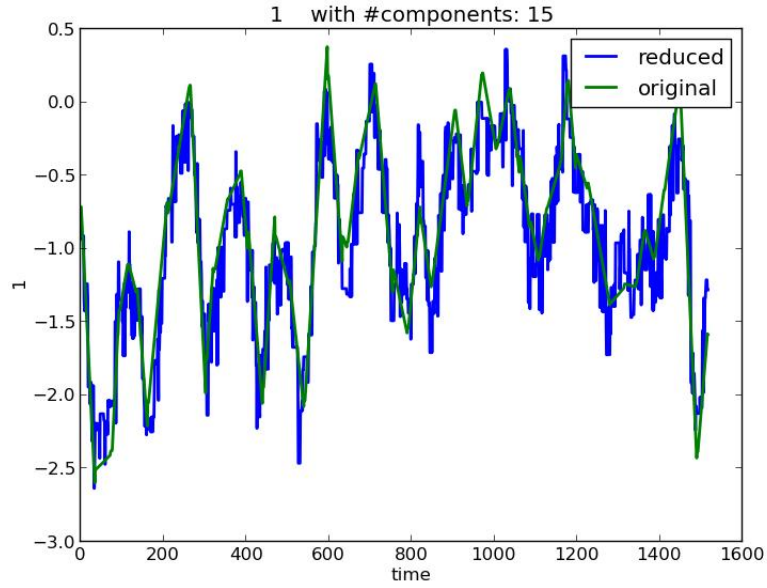


(b)

Figure 5.11: (a) A sequence of the (a) first and (b) second joint of the shoulder's positions where various seen arm trajectories are chained one after another. The green line represents the original seen trajectory and the blue line represents the same trajectory after the process of $data \rightarrow PCA \rightarrow neural_map \rightarrow PCA^{-1} \rightarrow reconstructed_data$.



(a)



(b)

Figure 5.12: (a) A sequence of the (a) first and (b) second joint of the shoulder's positions where various unseen arm trajectories are chained one after another. The green line represents the original unseen trajectory and the blue line represents the same trajectory after the process of $data \rightarrow PCA \rightarrow neural_map \rightarrow PCA^{-1} \rightarrow reconstructed_data$.

babbling trajectories than we have already collected; this requires hours of exploration that is simply not feasible with our robotic system. Another possibility is the features which we have included in the current motor-sensory space are uncorrelated and not informative for this reaching task. After taking these factors into account, we designed a less complex reaching task, which is described in section 5.5. A simpler reaching task would allow us to test the motion planning of our model and learn about different aspects of the model. This reaching task requires less motor babbling than the random exploration of the abstract box. Moreover, we considered a smaller motor-sensory space than 41 features. To do so, we omitted all the sensor values from the current space \mathcal{A} and only kept the motor commands of 5 joints in the left arm.

5.4 Evaluation Metrics of Planned Motions

In this section, we explain the three metrics that are used in sections 5.5 and 5.6 for evaluation of the planned arm motions during the test. These metrics are defined as:

- *End effector distance* estimates the accuracy of the reaching test as the distance in Cartesian space between the target position and the end effector after reaching is complete, that is, $\|g_{(x,y,z)} - \tilde{g}_{(x,y,z)}\|$, where $g_{(x,y,z)}$ is the desired location of the end effector and $\tilde{g}_{(x,y,z)}$ is the resulting location. This measurement is expanded as $\sqrt{(g_x - \tilde{g}_x)^2 + (g_y - \tilde{g}_y)^2 + (g_z - \tilde{g}_z)^2}$. This metric evaluates the planned motions in terms of closeness of the end effector with the target position in the Cartesian space, so a smaller error indicates a more accurate motion. In this chapter, the unit of this metric is the meter (m).
- *Norm jerk* evaluates the smoothness of the reaching trajectories in the joint space based on the time derivative of the joint angle acceleration; it is defined $\text{jerk} = \frac{1}{f} \sum_{t=1}^f \|\ddot{\mathbf{a}}_t\|$. In this equation, $\mathbf{a}_t \in \mathcal{A}$ is a point along a planned trajectory and f is the number of points in the trajectory. This metric evaluates the planned motions in terms of the smoothness in the joint space and a smaller

jerk is indicative of a smoother motion. In this chapter, the unit of this metric is rad/s^3 .

- *Overshooting Index* measures smoothness of reaching trajectories in the joint space. To measure the actual jerk, the exact time of each sub-movement of testing paths must be known, which is not easy to gauge accurately. Therefore, we defined an overshooting index, a unit-less metric, to measure the smoothness (jaggedness) of planned trajectories. For our reaching task, joint positions of a planned test trajectory can be viewed as increasing or decreasing straight lines in a 2D plane, in which the y axis is the joint position in radians and the x axis is the time in seconds. The overshooting index is defined as the rate of deviation from a straight line.

A straight line is the shortest possible line from the start position at time s , (J_s, T_s) , to the end position at time e , (J_e, T_e) . Here, J is the position of a particular joint in joint space at the time stamp T . The total length of this shortest line is $\sqrt{(J_s - J_e)^2 + (T_s - T_e)^2}$. A more jagged line will be a less ideal connection from the point (J_s, T_s) to (J_e, T_e) and thus be inevitably longer than the shortest line. The length of a longer line can be measured by adding the distance of consecutive points, i.e. $\sqrt{(J_i - J_{i+1})^2 + (T_i - T_{i+1})^2}$. The x and y values are normalized to the range $[-1, 1]$; this normalization is critical in adding different values with different units and possibly ranges.

Therefore, the overshooting index is the total length of a line divided by the length of the perfect line:

$$\text{Overshooting Index} = \frac{\sum_{i=1}^{e-1} \sqrt{(J_i - J_{i+1})^2 + (T_i - T_{i+1})^2}}{\sqrt{(J_s - J_e)^2 + (T_s - T_e)^2}} \quad (5.1)$$

A straight line has an overshooting index of 1.0, and more jagged lines have an index larger than 1.0. We individually calculate this index for the joints located in the shoulder and average them.

Table 5.8: Left arm’s configuration of the fixed start.

Joint Configuration	End-effector Position $[x, y, z]$
$[1.58, 0.0, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.986, 0.211, 1.336]$

5.5 Fixed Start Point

To reduce the complexity of our experiment in terms of number of needed training samples and the size of neural map, we performed a new set of experiments with a simple reaching task on an arc (Mahoor et al., 2016). Also, we didn’t include the sensors in the \mathcal{A} space. We used Rosie’s left arm for this experiment. The \mathcal{A} space, motor commands, consists of the position and velocity of joints in the shoulder and elbow. In total, the size of \mathcal{A} is 10 in the following experiments. 1000 random goal positions were generated in an arc in front of the robot for training (babbling) and testing our reaching controller in 3D space. 70% of these positions were used for training and 30% for testing. We used a single fixed start point, with an outstretched arm, for all trajectories (Fig. 5.13a and Fig. 5.13b). Table 5.8 shows the configuration of left arm for the fixed start along with the end-effector position of the start in the robot’s coordinate system. The zlift location was fixed at 670 in the following experiments. All communication with the robot is through the Robot Operating System (ROS). For training, trajectories are generated with “MoveIt!” (Sucan and Chitta, 2014).

In the first pass through the training trajectories, we trained an autoencoder using backpropagation with the tanh activation function. Before backpropagation, the training data was normalized to the range $[0, 1]$ by dividing each feature by its maximum value and subtracting its minimum value. The learning rate was 0.1 and the number of epochs was 10^6 .

In the second pass of training, neural maps were constructed by calculating a desired resolution for each feature (section 4.3.4.3 in chapter 4). This resolution was

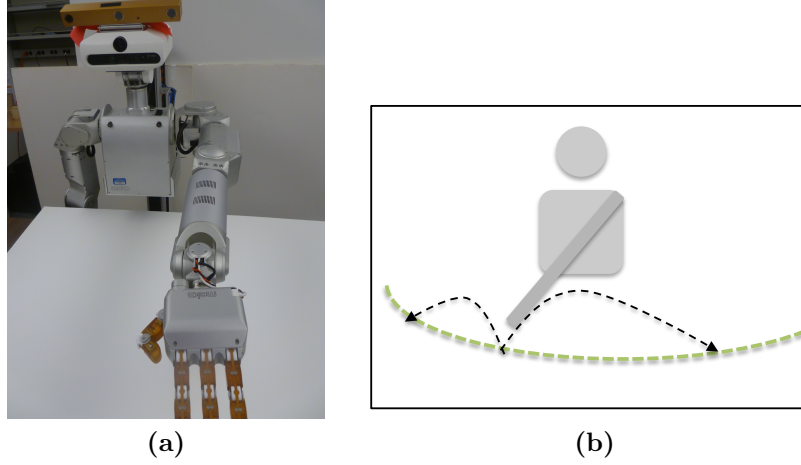


Figure 5.13: Experimental settings: (a) Meka Robotics M3 mobile humanoid robot “Rosie.” (b) The humanoid robot randomly explores an arc in front of its left arm with a fixed start pose. The initial and final positions of babbling trajectories are on this arc, but there is no constraint on points between the initial and final positions.

set to the median distance of consecutive points of the training trajectories in space \mathcal{A}' . The third pass through the babbling trajectories was to make connections among neurons in the neural map based on Alg. 1.

The goal of our path planning is to find a trajectory in the high-dimensional space \mathcal{A} that leads the arm from the current state to the goal state. The first step is to define the goal and start states in space \mathcal{A} and transform them to the reduced space \mathcal{A}' using the first half of the trained autoencoder. The start and goal points in \mathcal{A}' are used to directly locate the start and goal neurons in the neural maps. These points, along with path planning parameters, are passed to Alg. 3. These parameters are $\eta_B = 0.1$, $\tau_B = 10^3$, $\lambda = 10^3$, and $\text{max_steps} = 80$. As soon as a path is found by the winning neurons in map **C**, path execution is started. The winning neuron’s weight is translated to the motor commands using the second half of the autoencoder at each time step.

Table 5.9: Path Planning Parameters.

Parameter	Value
η_B	0.1
τ_B	10^3
λ	10^3
<i>max_steps</i>	80

5.5.1 Autoencoder as Dimension Reduction

In the first phase of training, we utilized PCA to find the suitable dimension of the space \mathcal{A}' . The training set contains 700 trajectories and number of features are 10. The eigenvalues resulted from PCA showed that for a variance of 95%, 8 features out 10 must be preserved. This number is rather high for creating a 2D neural map with a good resolution. This issue with PCA motivated us to instead use an autoencoder for this phase of training. We trained the autoencoder with an architecture of $10-30-15-10-3-10-15-30-10$ for layers, where $10-30-15-10$ represents the encoder part and $3-10-15-30-10$ represents the decoder section of the autoencoder. The autoencoder was trained with back-propagation with a learning rate of 0.1 and 1000 epochs. The activation function is hyperbolic tangent for neurons in all layers. The variance of data shows that for a variance of 95%, 3 features out 10 must be preserved. This means that the autoencoder is able to capture the same amount of variance within the data with a smaller number of features.

Here, we describe another interesting unexpected outcome of using the autoencoder for dimension reduction. A training trajectory was projected to the reduced space of size 3 using the encoder section and back to the original space using the decoder part. Fig. 5.14 shows the joints' positions and velocities of that trajectory along with its reconstructed version using process of "*data* \rightarrow *encoder* \rightarrow *decoder* \rightarrow *reconstructed_data*". We can see in this figure that the reconstructed joint positions differ from the original ones in two joints, one in the shoulder and one in the elbow,

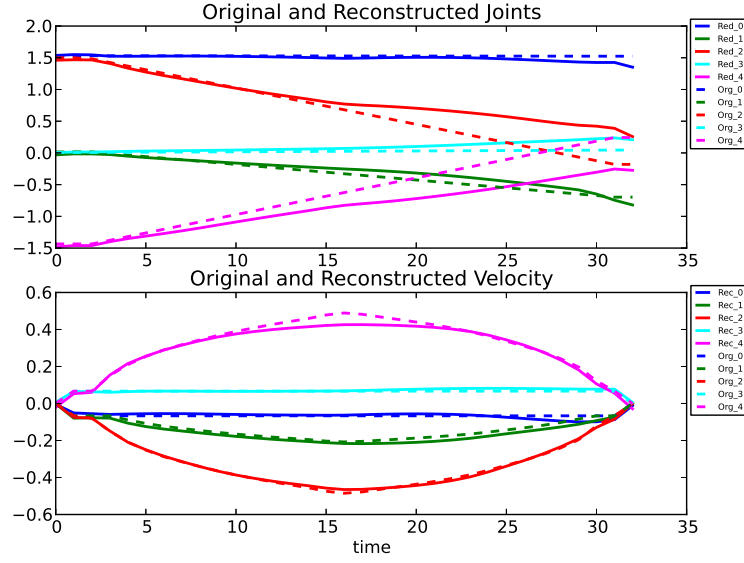


Figure 5.14: A babbling trajectory along with its reconstruction using “ $data \rightarrow encoder \rightarrow decoder \rightarrow reconstructed_data$ ”. The solid and dashed lines represent the reconstructed trajectories and the original ones respectively.

but the end positions are the same for these two trajectories. Interestingly the autoencoder has learned that the third joint of the shoulder and the second joint of the elbow compensate for each other. The autoencoder has discovered that multiple arm configurations with these two joints map to the same position of the end-effector in Cartesian space. We didn’t observe the same results in an autoencoder with the bottleneck of 4 or 5. A small bottleneck has forced the autoencoder to capture this interesting feature within the training data. In contrast, a bigger bottleneck has enough resources to store these configuration separately; therefore, it was not able to discover such a property within the training set.

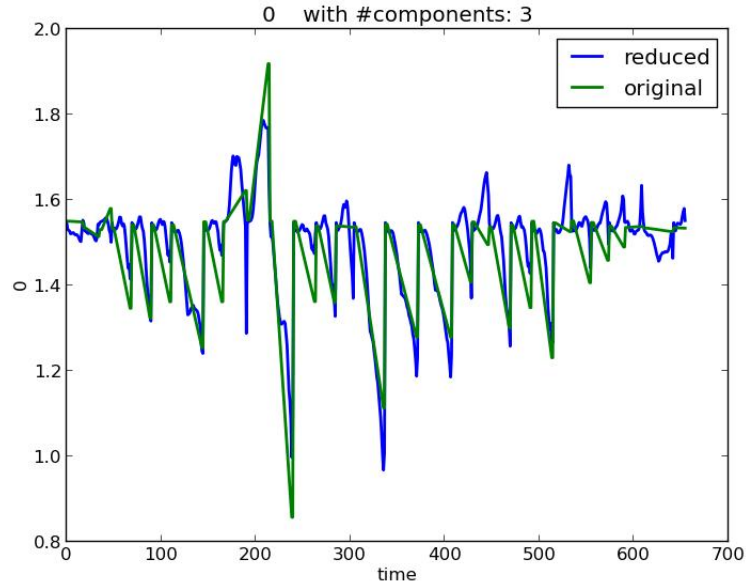
5.5.2 Construction of the Neural Map using Cartesian Product of Features

In this section, we explored creating a neural map using Cartesian product of features following section 4.3.4.2. We used the training data from the experiment with a

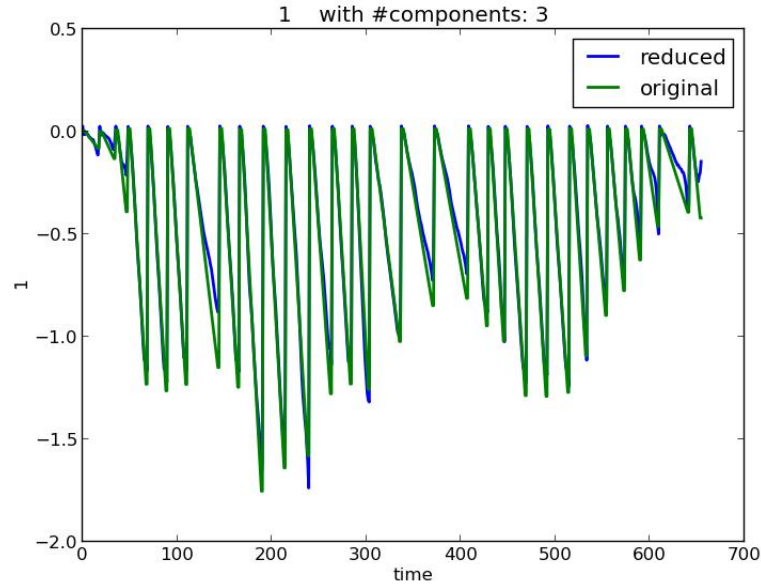
fixed start on the arc reaching. The size of babbling trajectories was 700 and the number of features was 10. The training set was projected to the reduced space using an autoencoder. For the reduced space of 3, the Cartesian product creates a neural map of 50×10^3 neurons. We evaluated the unseen trajectories using the test “*data* \rightarrow *encoder* \rightarrow *neural_map* \rightarrow *decoder* \rightarrow *reconstructed_data*”. Fig. 5.15 compares multiple original unseen (test) trajectories with their reconstructed ones. These two graphs show that the neural map has been able to successfully retain the statistics of the babbling data. This neural map is able to preserve the second joint of the shoulder even better than the first joint of the shoulder (comparing Fig. 5.15a with Fig. 5.15b). The second joint has a smaller range of position values opposed to the first joint. The first joint spans into both positive and negative values while the second joint has mostly negative values or positive values very close to 0.0. When a Cartesian product is used, a neural map can grow exponentially by adding more features to the reduced space. For example, for the reduced space of size of 5, a neural map of 60×10^6 neurons is needed to represent the motor-sensory space. This approach has multiple limitations. First, creating a neural map of such a size is not computationally tractable; second, a majority of those neurons will not be used in the bundle formation stage. For these reasons, we used the approach explained in section 4.3.4.3 for creating a neural map instead of the Cartesian product approach for the rest of experiments.

5.5.3 Results of Varying Dimension of Reduced Space

In this section, we examined the effect of reduced space size on motion planning. In one experiment we tested three levels of dimensionality for space \mathcal{A}' (*i.e.*, the number of neurons in the bottleneck of the autoencoder), while holding other parameters fixed. Different neural maps were built for each experimental dimensionality of the reduced space \mathcal{A}' . In all trials, the width parameter ϕ was set to 1 to create narrow



(a)



(b)

Figure 5.15: (a) A sequence of the (a) first and (b) second joint of the shoulder's positions where various unseen trajectories are chained one after another, mapping the original ones and after the reconstruction by “*data* \rightarrow *encoder* \rightarrow *neural_map* \rightarrow *decoder* \rightarrow *reconstructed_data*”.

Table 5.10: Accuracy of autoencoders with different bottleneck size.

Architecture	Train		Test	
	RMSE	Variance	RMSE	Variance
10-30-15-10-3	0.11	0.95	0.12	0.95
10-30-15-10-4	0.07	0.96	0.077	0.96
10-30-15-10-5	0.048	0.97	0.049	0.97

bundles. We evaluate the autoencoder using *root-mean-square error* according to

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n \|\mathbf{a}_i - \mathbf{a}_i^{\text{rec}}\|^2}{n}}. \quad (5.2)$$

In this equation, \mathbf{a} stands for a given point from the training set or test set, and \mathbf{a}^{rec} is an approximation of the same point calculated by feeding \mathbf{a} consecutively through the encoder and decoder parts.

Because the autoencoder is used to reduce the dimension of trajectories, another criterion,

$$\text{Variance} = 1 - \frac{\sum_{i=1}^n \|\mathbf{a}_i - \mathbf{a}_i^{\text{rec}}\|}{\sum_{i=1}^n \|\mathbf{a}_i\|}, \quad (5.3)$$

is introduced for determining how many components should be included and how many should be ignored. The variance metric is a method of determining the percentage of variance preserved after reducing the dimensionality. We evaluate different architectures by looking at the aforementioned metrics for both the training and test trajectories. Table 5.10 shows that as the size of the reduced space increases, the root-mean-square error decreases and the variance captured by the autoencoder increases. This observation holds for both the training and test sets.

Fig. 5.16a illustrates the norm jerk of planned trajectories for different dimensionalities of the reduced space \mathcal{A}' . As shown in Fig. 5.16a, the median of norm jerk decreases as the size of space \mathcal{A}' increases. This metric suggests that the smoothness of planned motions depends on the accuracy of the autoencoder. However, the range of norm jerk shows a nonlinear drop from size 3 to sizes 4 and 5. This nonlinear drop

in the norm jerk suggests that increasing the size of space \mathcal{A}' might not change the smoothness of motions after a certain size.

Fig. 5.16b shows the range of end effector position error across the three dimensionalities of space \mathcal{A}' . The error metric linearly decreases as the dimensionality of space \mathcal{A}' increases. Together, Figs. 5.16a and 5.16b show that the model has successfully learned to plan motions from the test set accurately and smoothly when the size of space \mathcal{A}' is 4 or 5.

Fig. 5.16c shows the mean of norm jerk across the y of final position of trajectories in the robot’s coordinate system (across the torso) along with the standard deviation. The motions are smoother across y for \mathcal{A}' of size 4 and 5. Fig. 5.16d displays the mean of the Euclidean distance of end effector positions from goal positions across the final position’s y along with the standard deviation. This metric also indicates that a space \mathcal{A}' of size 4 or 5 tends to produce more accurate planned motions; however, there is no particular trend in the error as y increases. An interesting anomaly in this figure is that the error is higher for small y values— that is, short trajectories. After investigating the planned motions, we found that a set of movements to the left side of the arc involves positive values of the second joint of the shoulder. It seems that these border values were not learned accurately in the autoencoder, and the model is not able to represent the poses that are located to the right of the arc as well as those on the left side.

Fig. 5.17a shows the overshooting of the trajectories across the three trials. We measured the overshooting metric (jaggedness) only for the joints in the shoulder, and the values shown in this figure represent the averaged jaggedness of these three joints. The results of this metric is compatible with the results of the norm jerk. This smoothness metric also confirms that a space \mathcal{A}' of size 4 or 5 create smoother motions compared to a space of size of 3. Fig. 5.17b shows the same smoothness metric across the final position’s y value of the testing trajectories. There is no specific pattern for trajectories as y is increasing, although for the dimensionality of 3 the overshooting metric is worse for y near 0.7. These trajectories that land on the far left side of the

arc ($y \sim 0.7$) are composed of the arm configurations that an autoencoder with a bottleneck of 3 is not able to capture their features correctly.

5.5.4 Results of Varying Bundle Width

In this section we examine the effect of bundle width on the generalization within both the neural map and the motion planning. In one experiment, we tested four different bundle widths of ϕ , while the rest of the parameters were fixed. The tested bundle widths are 1, 3, 6, and 10. In all trials of this experiment, the dimensionality of the reduced space was 5.

In the first pass through the training trajectories, an autoencoder with a bottleneck of 5 was created and the training trajectories were transformed to the reduced space using the encoder part. In the second pass of training, four neural maps were constructed for each experimental bundle width by taking the required width into consideration, meaning that, a larger neural map was created for a larger experimental bundle width. The resolution of the neural map was set to the median distance of consecutive points of the training trajectories in space \mathcal{A}' . The third pass through the babbling trajectories was to make connections among neurons in the neural map based on Alg. 1 with different bundle widths.

Fig. 5.18a demonstrates the norm jerk of the testing trajectories for four different bundle widths ϕ . We can see in Fig. 5.18a, as the bundle ϕ grows wider, the median of norm jerk increases slightly. Fig. 5.18b shows that the range of end effector error slightly decreases as the bundle becomes wider. The two figures show that increasing bundle width ϕ from 1 to 3 doesn't have any impact on either the end effector position error or the smoothness. This observation holds for changing a bundle width of 6 to bundle width of 10. However, we do observe significant improvement in the end effector error when the bundle width changes from 3 to 6. This shows that the model was allowed to generalize better with wider bundles, but the broader bundle also has had a negative impact on the overall smoothness of trajectories. The jerk

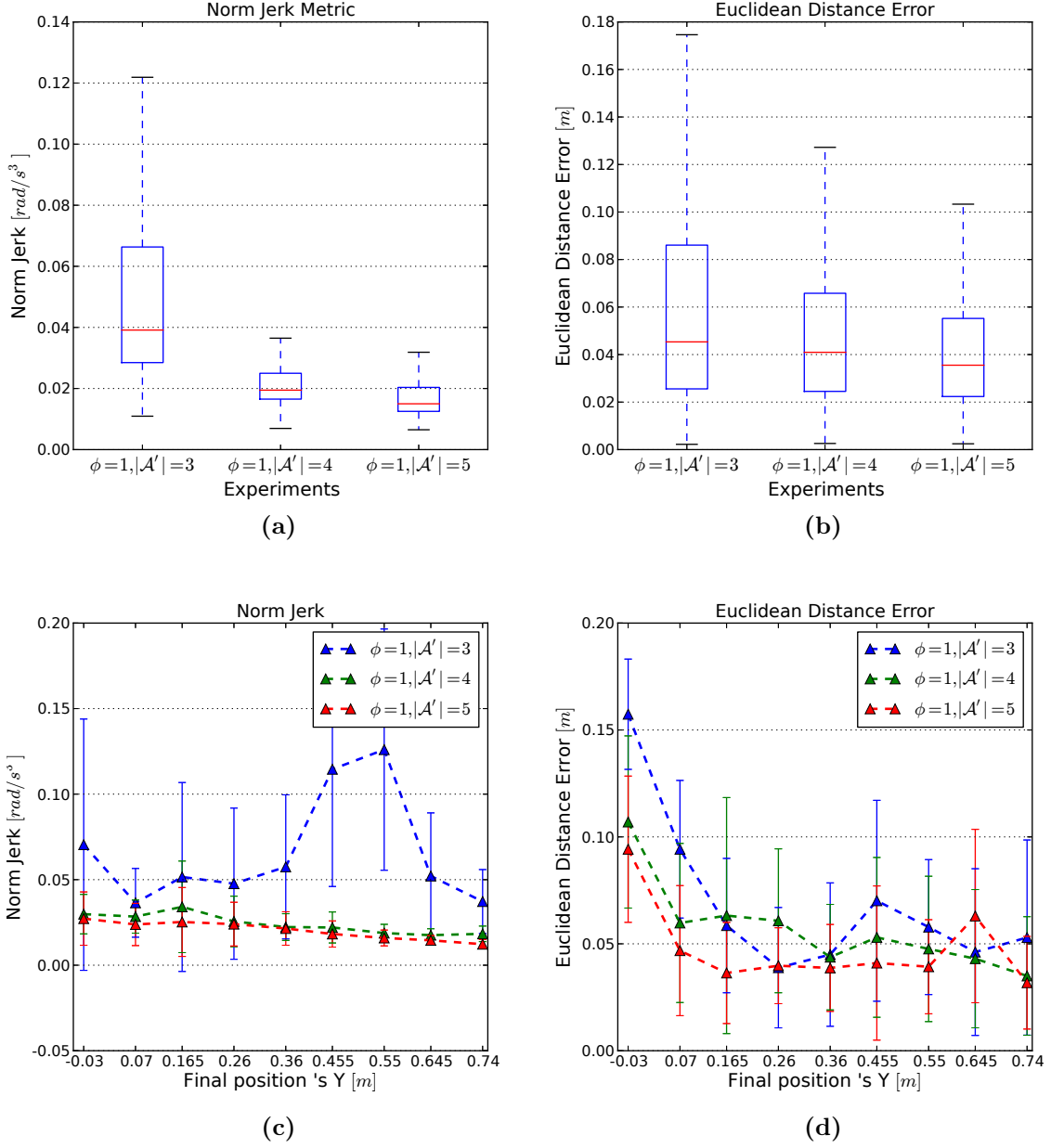


Figure 5.16: Evaluation of planned motions for three trials, where $|\phi| = 1$ is for the trials and the training data is 70%: (1) $|\mathcal{A}'| = 3$, (2) $|\mathcal{A}'| = 4$, and (3) $|\mathcal{A}'| = 5$. (a) Smoothness of planned trajectories based on norm jerk metric across three trials. (b) Euclidean distance error of end effector position across three trials. (c) Smoothness of planned trajectories based on norm jerk metric across the y value of the final position in robot's coordinate system. (d) Mean Euclidean distance error of end effector position across the y value of the final position in robot's coordinate system.

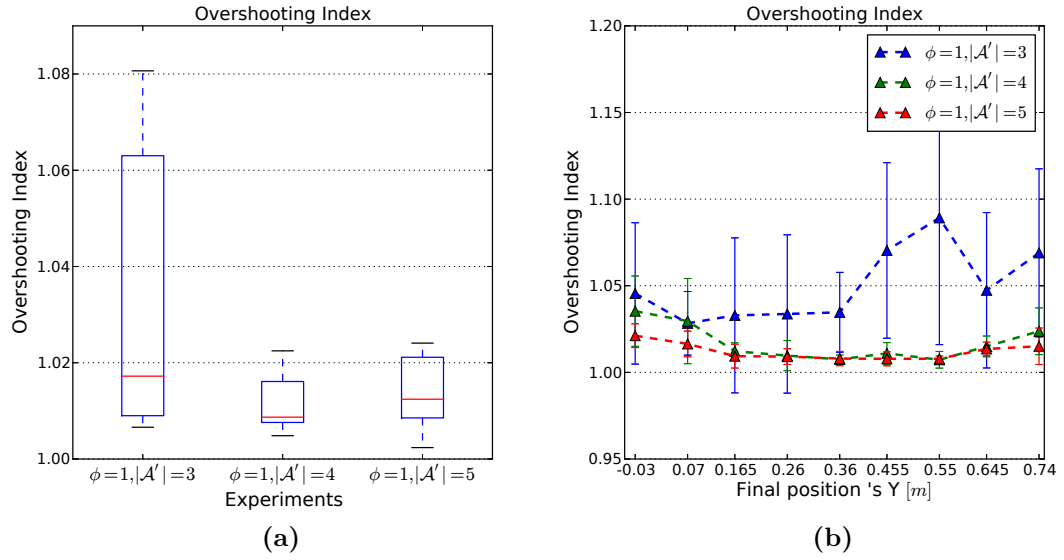


Figure 5.17: Evaluation of planned motions for three trials, where $|\phi| = 1$ is for the trials and the training data is 70%: (1) $|\mathcal{A}'| = 3$, (2) $|\mathcal{A}'| = 4$, and (3) $|\mathcal{A}'| = 5$. (a) Smoothness of planned trajectories based on overshooting index across three trials. (b) Smoothness of planned trajectories based on overshooting index across y value of the final position robot's coordinate system.

shown in this graph is an estimated jerk rather than an actual jerk. We assumed all the sub-movements of a planned motion took the same amount of time to execute. For this particular experiment, we also measured the actual execution time of each sub-movement of the planned motions and calculated the norm jerk with the exact time. The actual jerk showed the same trend as the estimated jerk, so for the rest of the experiments we only calculated the estimated jerk.

Fig. 5.18c shows the mean norm jerk across the final position's y in the robot's coordinate system along with the standard deviation. As we can see, the smoothness of trajectories consistently in four trials improves as y increases (longer motions). This figure also shows that the smaller bundle widths of 1 or 3 have led to smoother motions. The fact that the shorter trajectories are less smooth than the longer ones suggests that the autoencoder was not able to learn some of the joint positions that are more prevalent on the right side of the arc.

Fig. 5.18d displays the mean of the Euclidean distance of end effector positions from goal positions across the y component of the final position along with the standard deviation. This metric also indicates that a bundle width ϕ of size 6 or 10 tends to produce more accurate planned motions across the component y of the final positions except for the movements that land to the left side of the arc ($y \sim 0.07$). These shorter trajectories are in the boundary, and generalizing the babbling trajectories will not help to find a better motion on the edge side.

Fig. 5.19a shows the overshooting of the trajectories across the four trials with differing bundle width. The results of this metric is compatible with the results of the Euclidean distance and not the jerk. This smoothness metric also confirms that smaller bundles tend to create smoother motions compared to the wider bundles. Fig. 5.19b shows the same smoothness metric across y component of the final position of the testing trajectories. It can be seen in this graph that the overshooting is the worst for the testing trajectories where their final y poses are located in the far left ($y \sim -0.03$) or the far right of the arc ($y \sim 0.7$). The overshooting metric measures the jaggedness or deviation from a straight line for the joint positions in

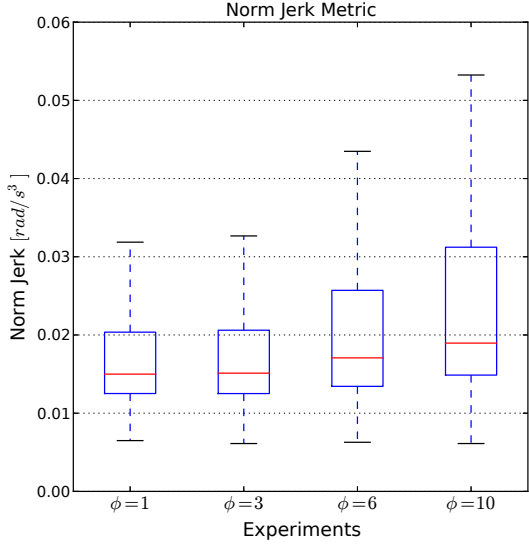
the shoulder. The fact that patterns of overshooting across y are similar to Euclidean distance suggests that the jaggedness has mostly occurred toward the end of the trajectories where the end configurations are defined. This test again confirms that the autoencoder needs more training samples to learn the boundaries of the working space.

5.5.5 Results of Varying Training Size

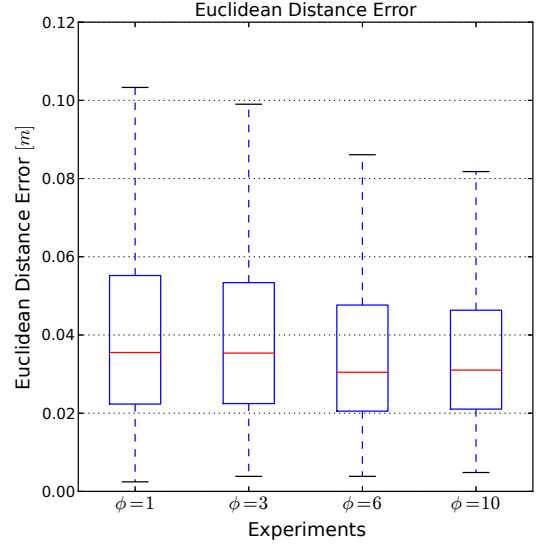
To find the number of babbling trajectories needed for the system to master this simple task, we trained the system with different portions of 1000 babbling trajectories. In one experiment, we varied the training portion while the rest of parameters were fixed during the test and the train. The training sets are 10%, 20%, 30%, 50%, and 70% of the 1000 trajectories. We should also mention that the testing size is fixed with 300 trajectories for the five trials. In all these trials, the bundle width was 1 and the dimensionality of the reduced space (the bottleneck of our autoencoder) was 5.

In the first pass through the training trajectories, we trained multiple autoencoders with a bottleneck of 5 using different training portions. We tested those autoencoders with a fixed test set. Table 5.11 shows that as the size of the training set increases, the accuracy of autoencoder improves both in the variance and the root-mean-square error of the test set. However, the accuracy of the autoencoder doesn't change significantly after the 30% training size. We therefore use this training size as the basis of the next experiment, which uses multiple fixed starting points.

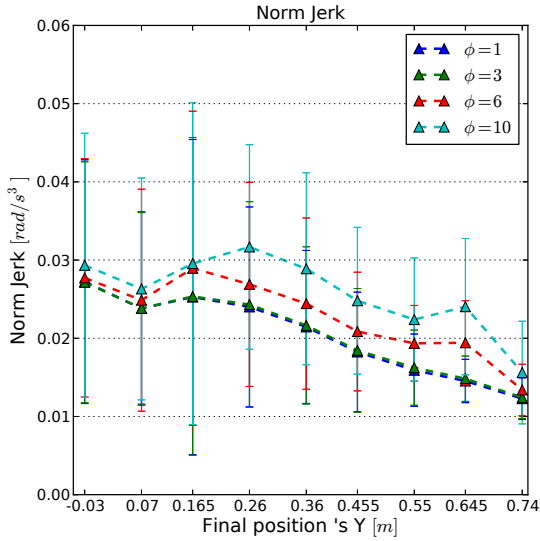
In the second pass of training, five neural maps were constructed using the corresponding training portion. The resolution of the neural map was set to the median distance of consecutive points of the training trajectories in space \mathcal{A}' . The third pass through each experimental training set was to make connections among neurons in the neural maps with the fixed bundle width of 1.



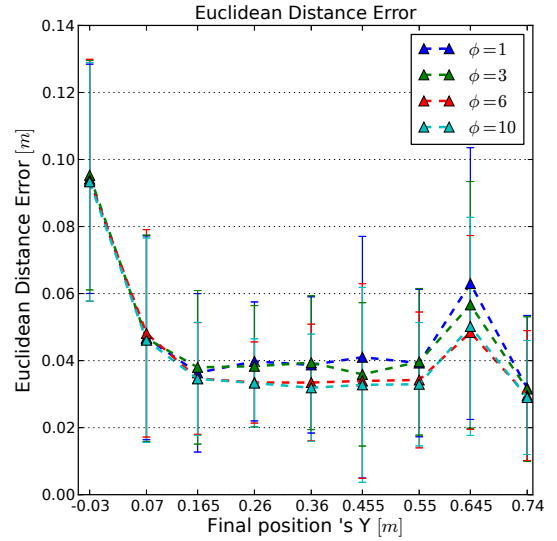
(a)



(b)



(c)



(d)

Figure 5.18: Evaluation of planned motions for four trials, where $|\mathcal{A}'| = 5$ is for the trials and the training data is 70%: (1) $|\phi| = 1$, (2) $|\phi| = 3$, (3) $|\phi| = 6$, and $|\phi| = 10$. (a) Smoothness of planned trajectories based on norm jerk metric across four trials. (b) Euclidean distance error of end effector position across four trials. (c) Smoothness of planned trajectories based on norm jerk metric versus y value of the final position the robot's coordinate system. (d) Mean Euclidean distance error of end effector position across the y values of the final position.

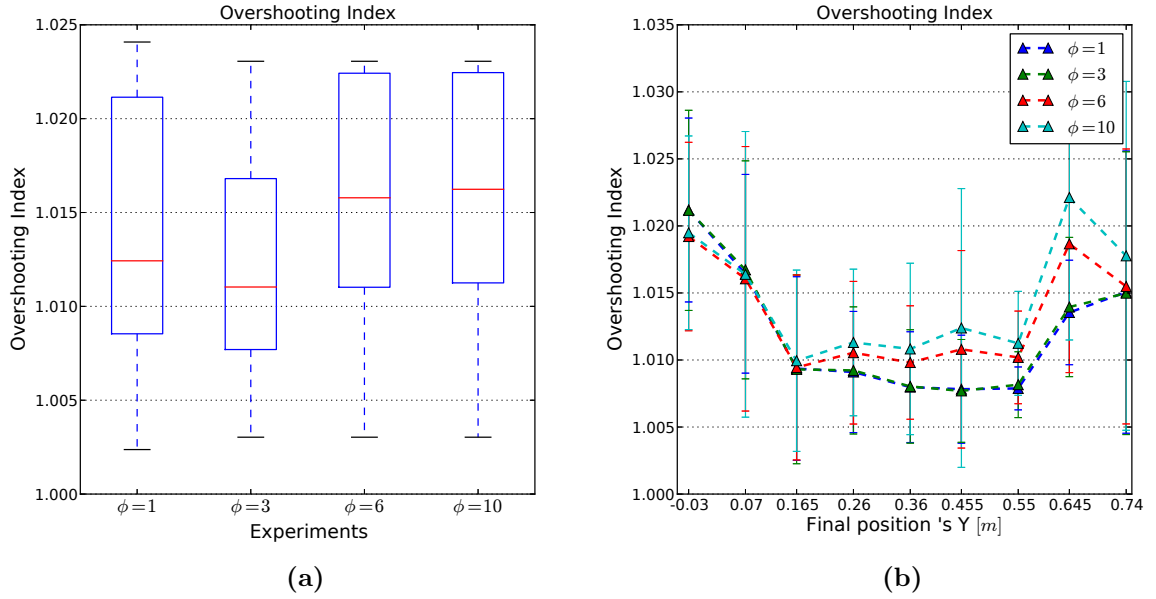


Figure 5.19: Evaluation of planned motions for four trials, where $|\mathcal{A}'| = 5$ is for the trials and the training data is 70%: (1) $|\phi| = 1$, (2) $|\phi| = 3$, (3) $|\phi| = 6$, and $|\phi| = 10$. (a) Smoothness of planned trajectories based on overshooting index across three trials. (b) Smoothness of planned trajectories based on overshooting index across y axis of robot's coordinate system.

Table 5.11: Accuracy of autoencoders with different training size.

Train size	Test RMSE	Test Variance
10%	0.21	0.91
20%	0.21	0.91
30%	0.04	0.96
50%	0.049	0.97
70%	0.049	0.97

Fig. 5.20b shows that the range of error in the end effector position for the testing trajectories slightly decreases as the size of training set increases. This graph also shows that 30% of the training set is a proper portion for the experiments in the next section. Fig. 5.20a demonstrates the norm jerk of planned trajectories for five trials with this training size. This graph shows that the jerk is not strictly correlated with the portion of the training set. For example, for the trial with 30% the median jerk is higher than the other trials, and we are not able to explain this result.

Fig. 5.20c shows the mean of norm jerk across the y value of the final location of the arm in the robot’s coordinate system along with the standard deviation. We can see that the smoothness of trajectories landing in places with y near -0.03 is worse than the other trajectories. This pattern interestingly persists with all the five trials. Fig. 5.20d displays the mean of the Euclidean distance of end effector positions from goal positions across the final position’s y value along with the standard deviation. The only interesting pattern in the figure is that the trajectories landing on the left side of the arc ($y \sim -0.03$) have had the biggest end effector error for all the five trials.

Fig. 5.21a shows the overshooting of the trajectories across the five trials with training size. From this figure, it can be seen that overshooting (similar to jerk) is not strictly correlated with the portion of training samples. Fig. 5.21b shows the same smoothness metric across y value of the final position of testing trajectories. The only pattern observed in this figure is the overshooting consistently is higher for both the

far left and far right sides of the arc, and having less or more training samples has not changed these patterns.

From all the measured metrics in this experiment, the Euclidean distance of the end effector and the accuracy of the autoencoders conclusively displayed the right amount of babbling needed for this reaching task. Based what we learned from these results, we set up a set of experiments with multiple starting points on the arc.

5.5.6 Diffusion-Based Path Planning versus Breadth First Search

In this section, we investigated the question: would spreading activation find a shortest path in the neural map? We compared implementation of path planning based on the spreading activation described in Section 4.3.2 with a traditional path planning, Breadth First Search (BFS). The neural map \mathbf{B} can be viewed as a directional graph; hence a shortest path from the goal neuron to the start neuron can be found using BFS. A shortest path has the minimum number of connections from the goal to the start. Fig. 5.22a shows a trajectory (the joints' positions and velocities) found by means of both spreading activation and bread first search. We can see that path planning using spreading activation doesn't necessarily lead to a shortest path from the goal to start. In this example, the planned trajectory is both stretched out through the time and longer; making it smoother than the one found using BFS.

5.5.7 Initial Bundle Formation

In section 4.3.1 of chapter 4, we introduced a modification to the bundle formation approach, which is to create initial connections among neurons before the bundle formation. Here, the initial bundle formation was performed to connect nearby neurons in the neural map; then we used the training trajectories to add more connections to the map with $\phi = 1$. In the initial bundle formation stage, the nearby

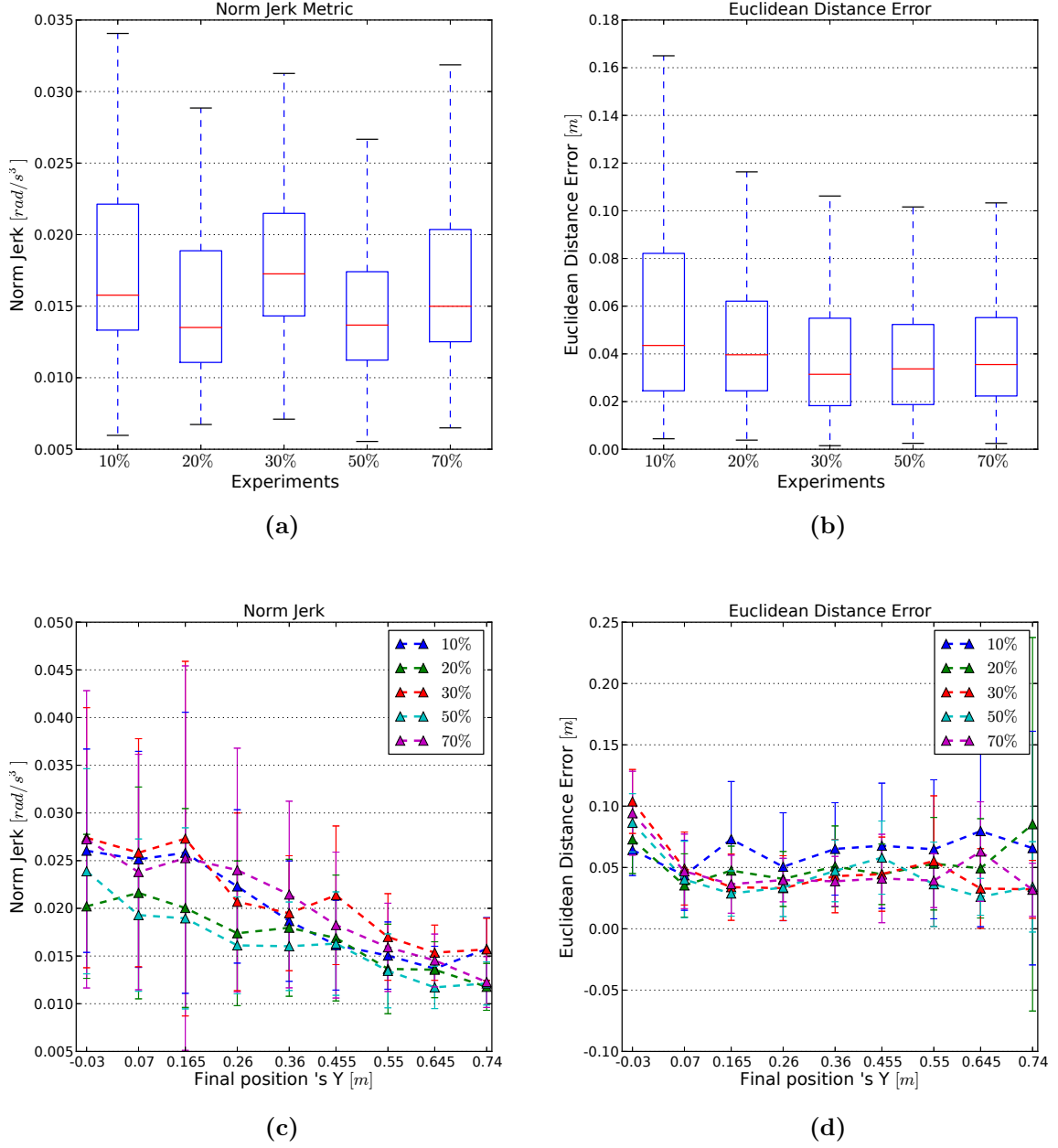


Figure 5.20: Evaluation of planned motions for five trials, where $|\phi| = 1$ and $|\mathcal{A}'| = 5$ are for the five trials: (1) $train = 10\%$, (2) $train = 20\%$, (3) $train = 30\%$, (4) $train = 50\%$, and (5) $train = 70\%$. (a) Smoothness of planned trajectories based on norm jerk metric across five trials. (b) Euclidean distance error of end effector position across five trials. (c) Smoothness of planned trajectories based on norm jerk metric across y value of final position of the arm in robot's coordinate system. (d) Mean Euclidean distance error of end effector position across y value of the final position.

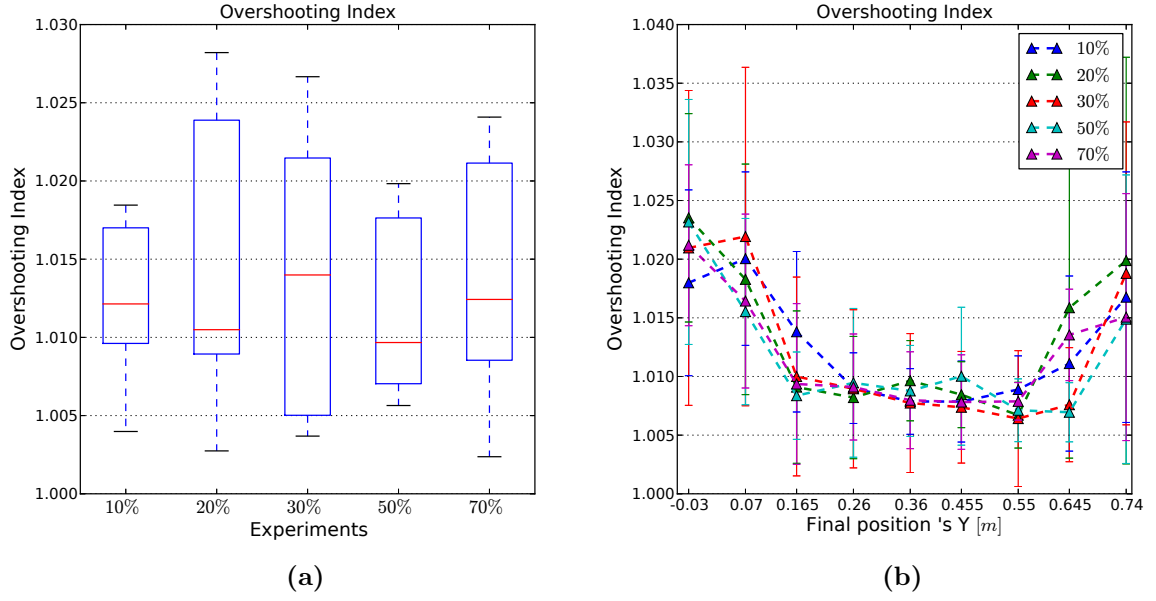
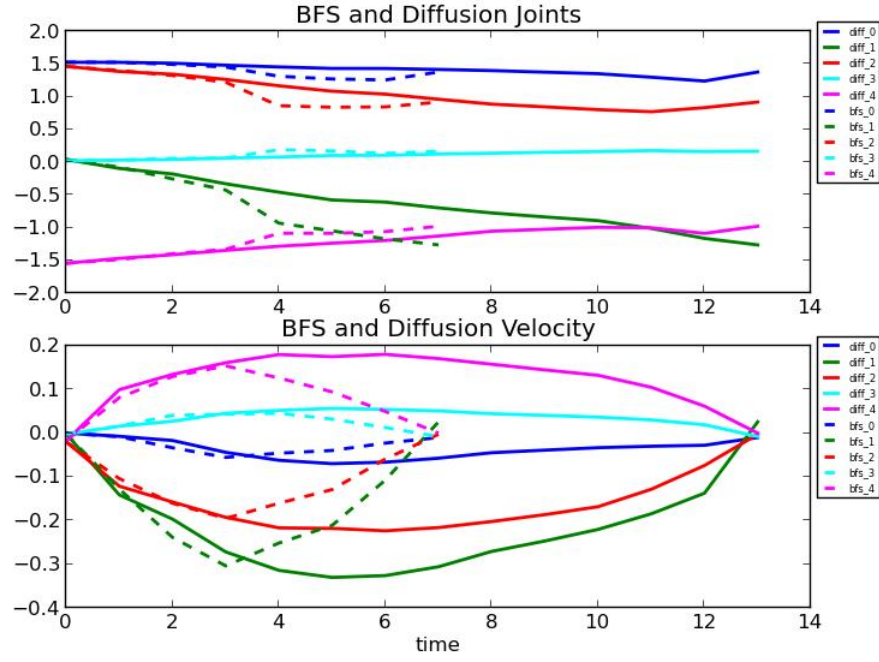
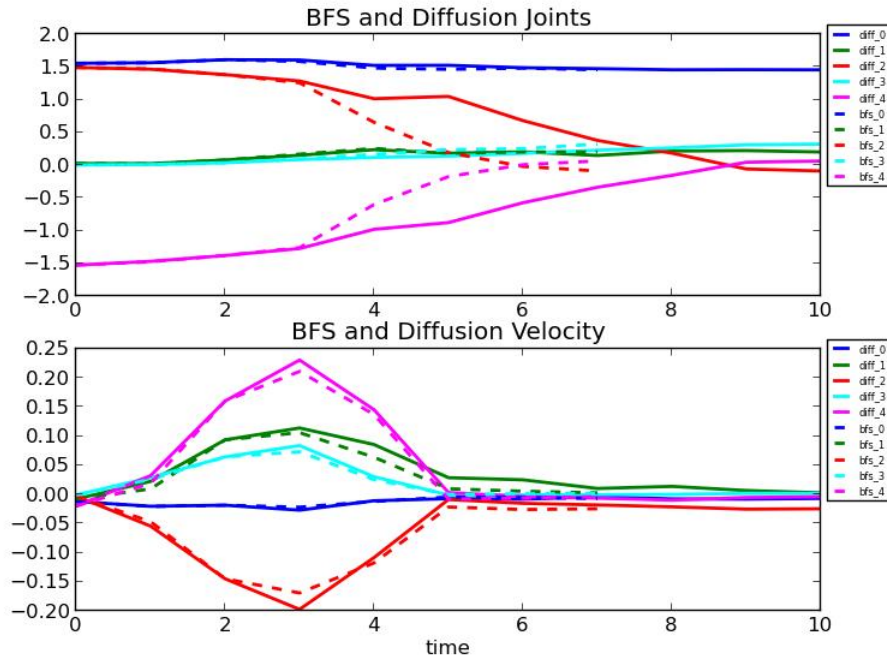


Figure 5.21: Evaluation of planned motions for five trials, where $|\phi| = 1$ and $|\mathcal{A}'| = 5$ are for the five trials: (1) $train = 10\%$, (2) $train = 20\%$, (3) $train = 30\%$, (4) $train = 50\%$, and (5) $train = 70\%$. (a) Smoothness of planned trajectories based on overshooting index across five trials. (b) Smoothness of planned trajectories based on overshooting index across y value of final position of the arm in robot's coordinate system.



(a)



(b)

Figure 5.22: (a) The dashed and solid lines are planned joint positions and velocities using breadth first search and the spreading activity and (b) a planned trajectory where the bundle formation stage was combined with Alg. 2.

neurons are found by measuring the Euclidean distance of the center of every pair of RBF neurons. After the bundle formation phase, we planned test motions using the path planning algorithm. Fig. 5.22b shows one of the planned trajectories within such a neural map, which shows that the velocity of this trajectory was near zero and stable for multiple time steps (from step 5 to step 10) while the joint positions were still changing. This happens because the initial connection stage has connected neurons that represent points of space \mathcal{A} with the velocity of zero even though they embed different joint positions. These motor-sensory states with a velocity of zero could be the start or end of motions. We observe this problem with multiple other planned trajectories. Since it is not clear how neurons are connected in this approach, we don't perform any pre-bundle formation for the rest of experiments.

5.6 Multiple Fixed Start Points

In another experiment with a slightly more difficult reaching task, we use multiple fixed start points, with the outstretched arm in eight different starting points (Fig. 5.23a). Table 5.12 shows the configuration of the left arm for those fixed eight start points along with their location of end effector in the robot's coordinate system (Cartesian space). The arm configuration varies only for the second joint of the shoulder which spans the interval of $[0.0, -1.4]$ with a step size of -0.2 . However, the location of start is changing in uniform fashion in the joint space, but it is not uniform in the Cartesian space. This lack of uniformity can be explained by the elliptical shape of the arch. For each of the eight start positions, we collected 300 training trajectories that stop at random points on the same arc. The training size is in total 2400 trajectories. For the purpose of testing this experiment, we collected 150 testing points on the arc for each of the eight start positions. The test set size is in total 1200 points.

Table 5.12: Left arm’s configuration for the starts.

Start	Joints’ Position	End-effector Position $[x, y, z]$
1	$[1.58, -0.0, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.986, 0.211, 1.336]$
2	$[1.58, -0.2, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.970, 0.321, 1.336]$
3	$[1.58, -0.4, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.932, 0.426, 1.336]$
4	$[1.58, -0.6, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.874, 0.521, 1.335]$
5	$[1.58, -0.8, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.798, 0.602, 1.334]$
6	$[1.58, -1.0, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.708, 0.667, 1.334]$
7	$[1.58, -1.2, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.607, 0.713, 1.333]$
8	$[1.58, -1.4, 1.50, 0.0, -1.50, 0.0, 0.0]$	$[0.498, 0.737, 1.332]$

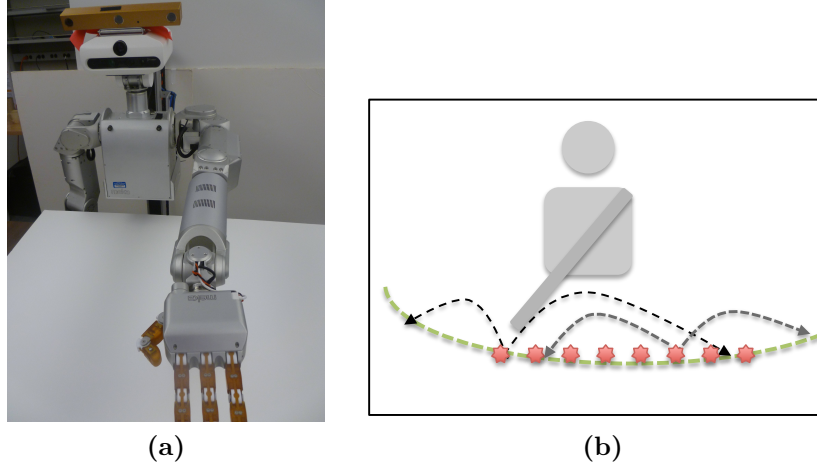


Figure 5.23: Experimental settings: (a) Meka Robotics M3 mobile humanoid robot “Rosie.” (b) The humanoid robot randomly explores an arc in front of its left arm from 8 different start poses. The initial and final positions of babbling trajectories are on this arc, but there is no constraint on points between the initial and final positions.

5.6.1 Results of Varying Bundle Width

In this section we examine the effect of bundle width on both the generalization within the neural map and the motion planning. In this experiment, we varied the bundle width in three trials with $|\phi|$: 1, 3, 6. The other parameters were fixed during the test and the train for the three trials. Here, we trained an autoencoder with the reduced space size (bottleneck size) of 6. In this part of the study, we examined the effects of

bundle width on the accuracy of motion planning while the rest of parameters were fixed. The remaining parameters of the training and path planning are the same as the experiment with the single fixed start (Section 5.5).

During the first pass through the training trajectories, an autoencoder with a bottleneck of 6 was created, and the training trajectories were transformed to the reduced space using the encoder part of the autoencoder. In the second pass of training, three neural maps were constructed for each experimental bundle width. The resolution of the neural map was set to the median distance of consecutive points of the training trajectories in space \mathcal{A}' . The third pass through the babbling trajectories was to make connections among neurons in the neural map based on Alg. 1 with different bundle widths of 1, 3, and 6.

Fig. 5.26c demonstrates the norm jerk of planned trajectories for three different bundle widths. We can see in Fig. 5.26c that while the bundle ϕ is increasing from 1 to 3 or 6, the median of norm jerk increases slightly. Fig. 5.24c shows that the range of end effector position error slightly lowers as the bundle width increases from 1 to 3. The two figures show that increasing the bundle width ϕ from 3 to 6 has changed neither the end effector position error nor the smoothness. However, we observe significant improvement in end effector position error for bundle widths of 3 or 6. This shows that the model was allowed to generalize better with wider bundles, but the broader bundle also negatively impacts the overall estimated smoothness of the trajectories.

Fig. 5.26b shows the mean of norm jerk across the absolute change in the final position's y in the robot's coordinate system along with the standard deviation. As we can see, the smoothness of trajectories in three trials consistently diminishes as the y increases (longer motions). This figure also shows that the smaller bundle widths of 1 or 3 have produced smoother movements.

Fig. 5.24b displays the mean of the Euclidean distance of end effector positions from goal positions across the final position's y component along with the standard deviation. This metric also indicates that a bundle width of size 3 or 6 tends to

produce more accurate planned motions except for the movements that land on the right side of the arc ($y \sim 0.07$). This experiment also confirms that the shorter trajectories are in the boundary and that generalizing by using the wider bundle width will not help to find a better motion on the edge side.

Fig. 5.25c shows the overshooting of the trajectories across the three trials with bundle width. We see here that the overshooting metric slightly increases as the bundle width becomes wider. Fig. 5.25b shows the same smoothness metric across the y component of the final position of testing trajectories. We didn't observe any particular pattern for the three bundle widths in the overshooting across final position's y component.

Fig. 5.24a, Fig. 5.25a, and Fig. 5.26a show the three metrics for planned motions across the y of start positions. We plotted these graphs to check whether the quality of planned motions is different under different start points. We did not observe any difference from one start over the others, so we didn't plot this type of graph for the rest of experiments using multiple start points.

5.6.2 Results of Modifying Bundle Formation

In this section, we investigate two important aspects of the implementation of bundle formation in chapter 4. Since the first investigation focused on the synapse's strength, we wanted to also see what would happen if all the synapses in a bundle had the same strengths. Secondly, we focused on the number of synapses in bundles: in other words, what would happen if the bundle had fewer synapses?

In this experiment, we addressed these questions using three trials with an implementation of bundle formation called *lnrConnections*, *fixConnections*, and *parConnections* in terms of accuracy and smoothness. In these trials, $|\phi| = 3$, $|\mathcal{A}'| = 6$, and the path planning parameters were $\eta_B = 0.1$, $\tau_B = 10^3$, $\lambda = 10^3$, and $\text{max_steps} = 80$. The first pass through the training trajectories, an autoencoder with a bottleneck of 6 was created, and the training trajectories were transformed

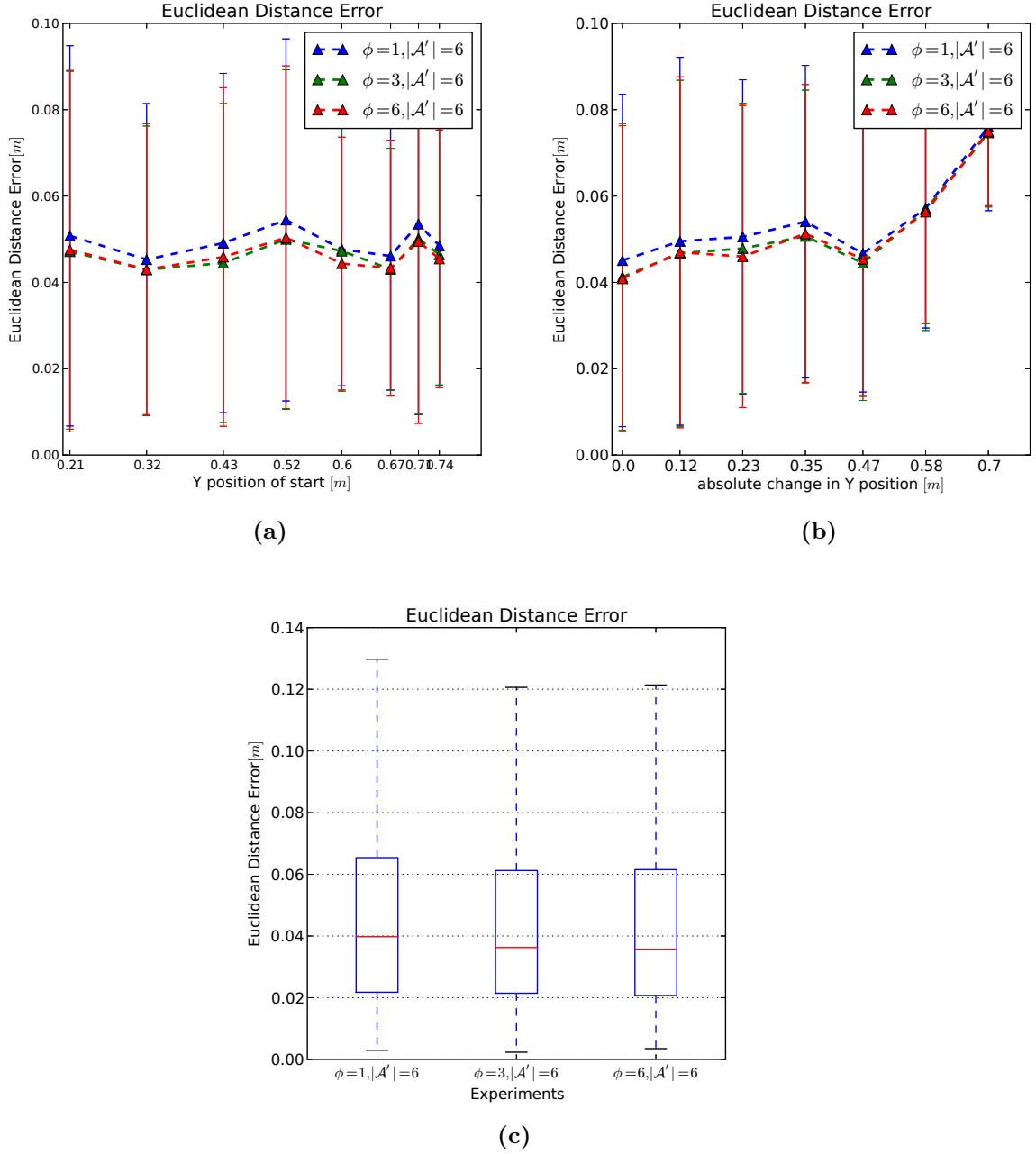


Figure 5.24: Evaluation of planned motions for three trials, where $|\mathcal{A}'| = 6$ is set for the trials and training is performed for 8 different start points: (1) $|\phi| = 1$, (2) $|\phi| = 3$, (3) $|\phi| = 6$. (a) Euclidean distance error of end effector position of planned trajectories across start position's y in the robot's coordinate system for 8 start poses. (b) Euclidean distance error of end effector position of planned trajectories across absolute change in y position in the robot's coordinate system. (c) Euclidean distance error of end effector position across three trials with ϕ .

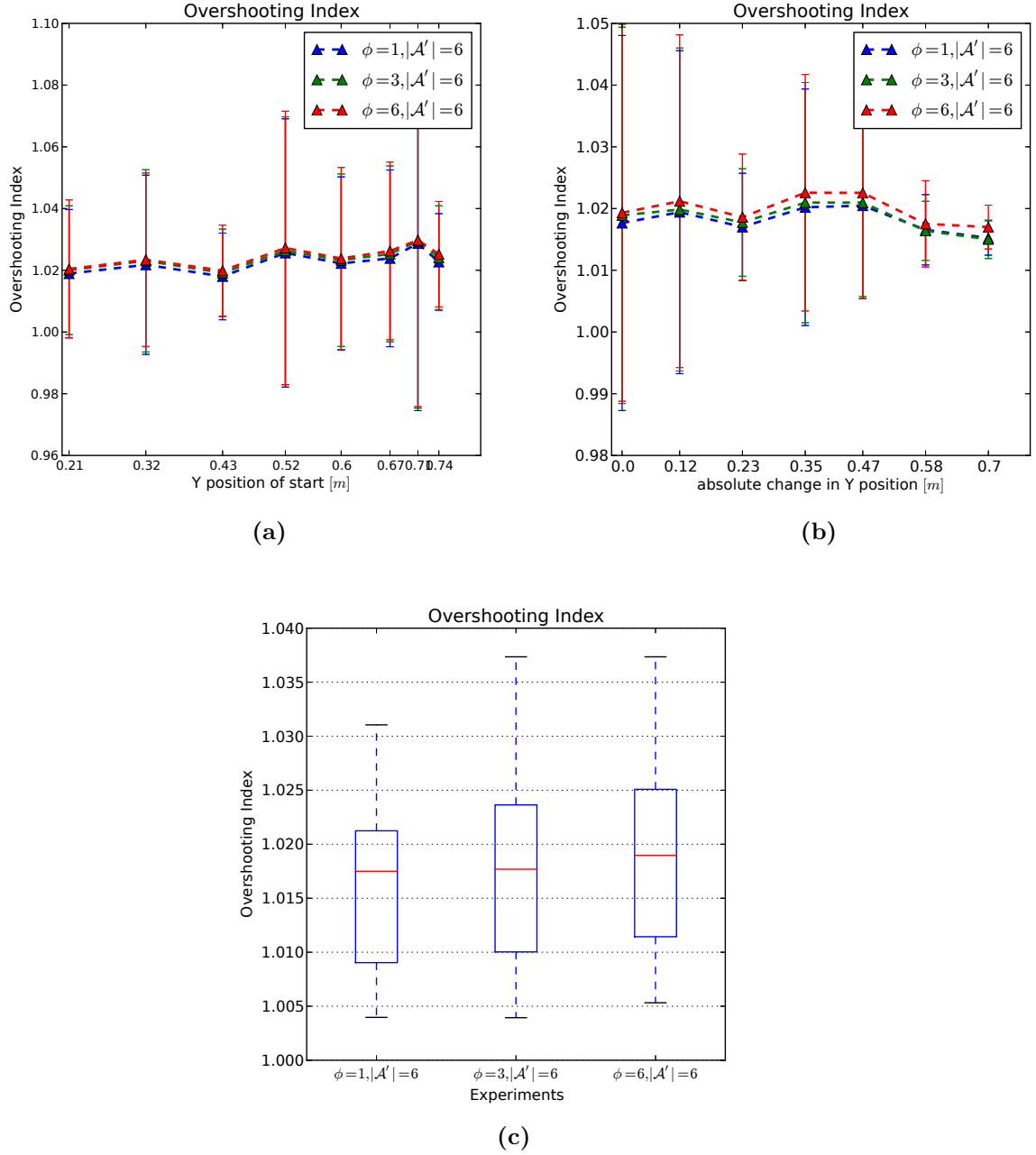


Figure 5.25: Evaluation of planned motions for four trials, where $|\mathcal{A}'| = 6$ is set for the trials and training is performed for various fixed start points: (1) $|\phi| = 1$, (2) $|\phi| = 3$, (3) $|\phi| = 6$. (a) Smoothness of planned trajectories based on overshooting index across start position's y in the robot's coordinate system for 8 start poses. (b) Smoothness of planned trajectories based on overshooting index across absolute change in y position in the robot's coordinate system. (c) Smoothness of planned trajectories based on overshooting index across three trials with ϕ .

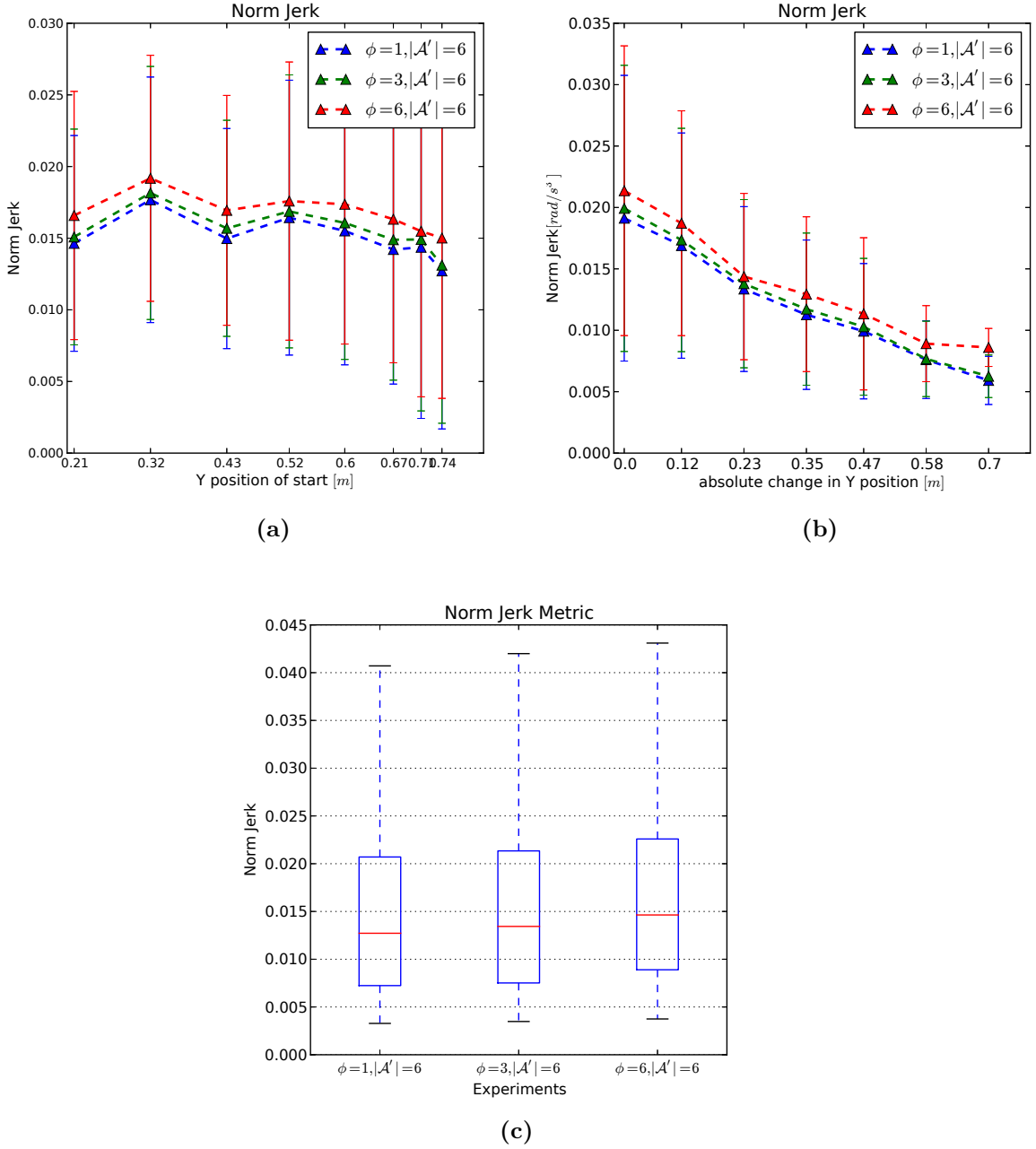


Figure 5.26: Evaluation of planned motions for four trials, where $|\mathcal{A}'| = 6$ is set for the trials and training is performed for various fixed start points: (1) $|\phi| = 1$, (2) $|\phi| = 3$, (3) $|\phi| = 6$. (a) Smoothness of planned trajectories based on norm jerk metric across start position's y in the robot's coordinate system. (b) Smoothness of planned trajectories based on norm jerk metric across absolute change in y position in the robot's coordinate system. (c) Smoothness of planned trajectories based on norm jerk metric across three trials with ϕ .

to the reduced space using the encoder part of the autoencoder. In the second pass of training, a neural map was constructed that will be used in the three trials. The resolution of the neural map was set to the median distance of the consecutive points of the training trajectories in space \mathcal{A}' . The third pass through the babbling trajectories was to make connections among neurons in the neural map using the *lnrConnections*, *fixConnections*, and *parConnections* approaches in three separate trials.

In the trial *lnrConnections*, bundles of width of 3 are formed with the connections in the center of bundles weighted more than the connections on the side of the bundle. The change from the center to the side is a linear drop in this trial. In the next trial, *fixConnections*, bundles of the width of 3 are formed, but the connections are uniform and have the weight of 1.0 throughout the bundles. In the trial *parConnections*, bundles with a width of 3 are formed but in a slightly different way. During bundle formation in the two previous trials, we set up connections among all the neurons that are activated at time i with all the neurons which are fired at time $i+1$ (section 4.3.1). The connections from stage i to stage $i+1$ create a bipartite directional graph. In this trial, we instead ranked the firing neurons at time i and time $i+1$ based on their rates of firing. The connections from step i to step $i+1$ are only created among the neurons with the same rank. This approach creates parallel rivers within a bundle.

To analyse the results, we considered trial *lnrConnections* as the base case and compared the results of other two trials against it. The accuracy of the planned motions in terms of Euclidean distance metric for the three trials were the same, so we only compare them based on the other two metrics.

Fig. 5.27a and Fig. 5.27b show that smoothness of these three trials regarding estimated norm jerk and the overshooting index. The trial *fixConnections* has the worst motion smoothness compared to the other trials. This result could mean that it is more biologically plausible to have non-uniform bundles where neurons have different synapse strength throughout the bundles. Fig. 5.27a and Fig. 5.27b show that trial *parConnections* has produced less smooth motions compared to the

base case. The average of norm jerk is 0.015 for the base case and 0.02 for trial *parConnections*. The closeness of the mean of norm jerk suggests that the bundle formation in the test *parConnections* has been unfavorable for some of the testing motions and not all of them. This result could also mean that it is more biologically plausible to have denser bundles regarding neurons' connectivity. The presence of more connections within in a bundle gives the path planning and spreading activation a better chance to find an optimal path.

Fig. 5.27c and Fig. 5.27d demonstrate the smoothness of planned motions across absolute difference in the y component of the end-effector in the robot's coordinate system. The x axis of these plots captures the estimated length of the planned trajectories, or in another words, the traveling distance of planned motions from the start point to the end point. In Fig. 5.27c, we observe a familiar pattern with the norm jerk: as the traverse length increases, the norm jerk decreases for the three trials. We didn't observe any other specific trends in this graph. Fig. 5.27d demonstrates the overshooting metric for the three trials across absolute difference in the y component of the end-effector position in the robot's coordinate system. An interesting observation from this plot is that the trial *fixConnections* has performed worse for the shorter trajectories compared to the base case.

5.6.3 Results of Varying the Resolution of the Neural Map

In this section we investigate the effect of neural map's resolution on the quality of testing motions. Specifically, we are interested in how the neural map's structure affects generalization.

In this experiment, we varied the resolution of the neural map in three trials while the rest of the parameters were fixed. The bundle width was 3 ($|\phi| = 3$) and the size of reduced space was 6 ($|\mathcal{A}'| = 6$). In the first trial or base case (*medRes*), the resolution of the neural map was set to the median of difference between features of the consecutive points in the trajectories. For the second ($2 \times \text{medRes}$) and third trial

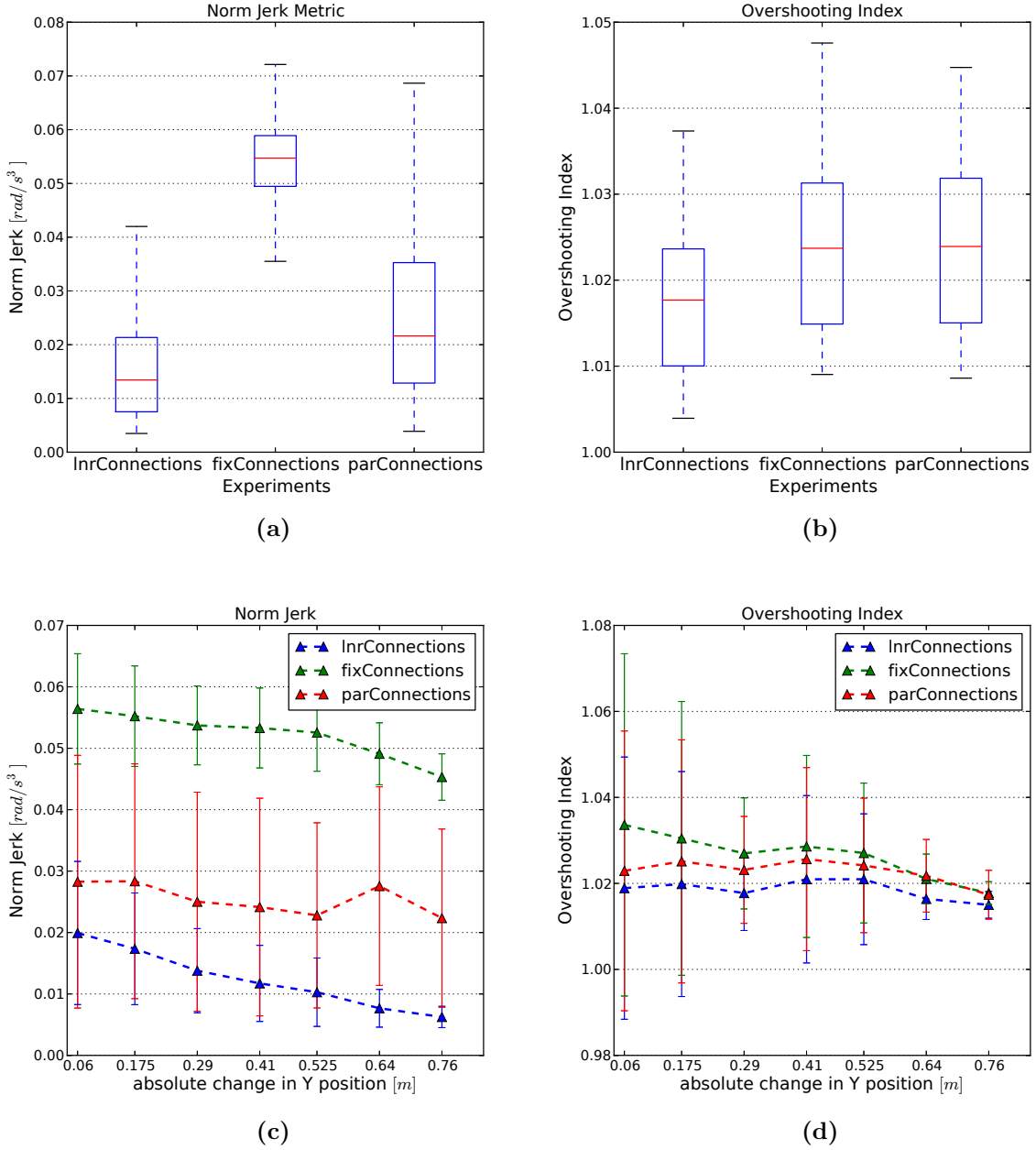


Figure 5.27: Evaluation of planned motions for three trials, where $|\phi| = 3$ and $|\mathcal{A}'| = 6$ are for the three trials called: *lnrConnections*, *fixConnections*, and *parConnections*. (a) Smoothness of planned trajectories based on the norm jerk metric across four trials. (b) Smoothness of planned trajectories based on the overshooting index across four trials. (c) Smoothness of planned trajectories based on the norm jerk metric across absolute difference in y position of end-effector in the robot's coordinate system. (d) Smoothness of planned trajectories based on the overshooting index across absolute difference in the y component of end effector position in the robot's coordinate system.

($3 \times medRes$), the resolution was set respectively to 2 and 3 times the resolution of the base case, in order to make the resolution more coarse than the base.

In the first pass through the training trajectories, an autoencoder with a bottleneck of 6 was created, and the training trajectories were transformed to the reduced space using the encoder part of the autoencoder. In the second pass of training, three different neural maps with different resolutions were constructed which would be used in the three trials. The third pass through the babbling trajectories was to make connections among neurons in the neural maps based on Alg. 1 with a bundle width of 3.

Fig. 5.28a demonstrates the norm jerk of the planned trajectories for these three trials with the neural map resolutions. The average smoothness of trajectories has increased as the resolution of maps becomes more coarse. This finding follows our expectation that the coarser resolution would produce less accurate motions. Fig. 5.28c shows the mean of the norm jerk across the absolute change in y of end-effector position in the robot’s coordinate system along with the standard deviation. This figure also confirms that both smoothness diminishes as the resolution becomes more coarse and that the smoothness is decreasing as y increases. This figure implies that shorter trajectories have the worst smoothness in this experiment.

Fig. 5.28b shows the overshooting of the trajectories across the three trials with the resolution of the neural map. The average of the overshooting metric has not changed for the three trials. Fig. 5.28d shows the same smoothness metric across the absolute change in y value of the final position of the testing trajectories. The average of the overshooting doesn’t change as y increases so this metric is the same for the planned trajectories with different lengths. The overshooting is the highest for the trial with $3 \times medRes$.

Fig. 5.29a shows the error in the end effector for the three different trials. In this figure we don’t see any major difference in the Euclidean error when the resolution of the map has changed. Fig. 5.29b displays the mean of the Euclidean distance of end effector positions from the goal positions across the absolute change in the y in the

robot’s coordinate system along with the standard deviation. We see the Euclidean distance has been worse in all three trials when $y < 0.7$, or when the length of travel for the motions is longer.

Based on the jerk and overshooting metrics, it is fair to say that in a map with coarse resolution, neurons are more general and cannot represent the fine details of reaching. In the neural maps with resolutions of $2 \times medRes$ or $3 \times medRes$, neurons respond to a larger area of the motor-sensory space. Thus, the planned motions can become less smooth. On the other hand, in a map with resolution of $medRes$, neurons represent more details of motor-sensory space, so the test-planned motions are smoother.

5.7 Discussion

In this chapter, we analyzed some important aspects of our model including but not limited to motor-sensory space representation, dimension reduction, bundle formation, amount of babbling needed, and the neural map’s resolution.

In this dissertation, we first chose GHA as our dimensionality reduction technique because it is neurally plausible and provides results that match PCA’s results. The main limitations of this method are that it is globally linear and computationally slow. The sub-spaces that are found by PCA or GHA are therefore linear, and they could be of higher dimension than necessary if the true underlying structure of the data is not linear. Therefore, we then switched to autoencoders, which are one of the non-linear dimensional reduction techniques. Autoencoders have shown to capture the properties of the data in a more constructive way than PCAs while the dimension of the found space is smaller than PCA. It could be interesting to test the performance of other variations of autoencoders in terms of accuracy of planned motions.

One important piece of our model is to design the neural representation of motor-sensory space that effectively represents this space. In the first try, we used the Self-organizing Map to automatically create such a map. The main limitation of

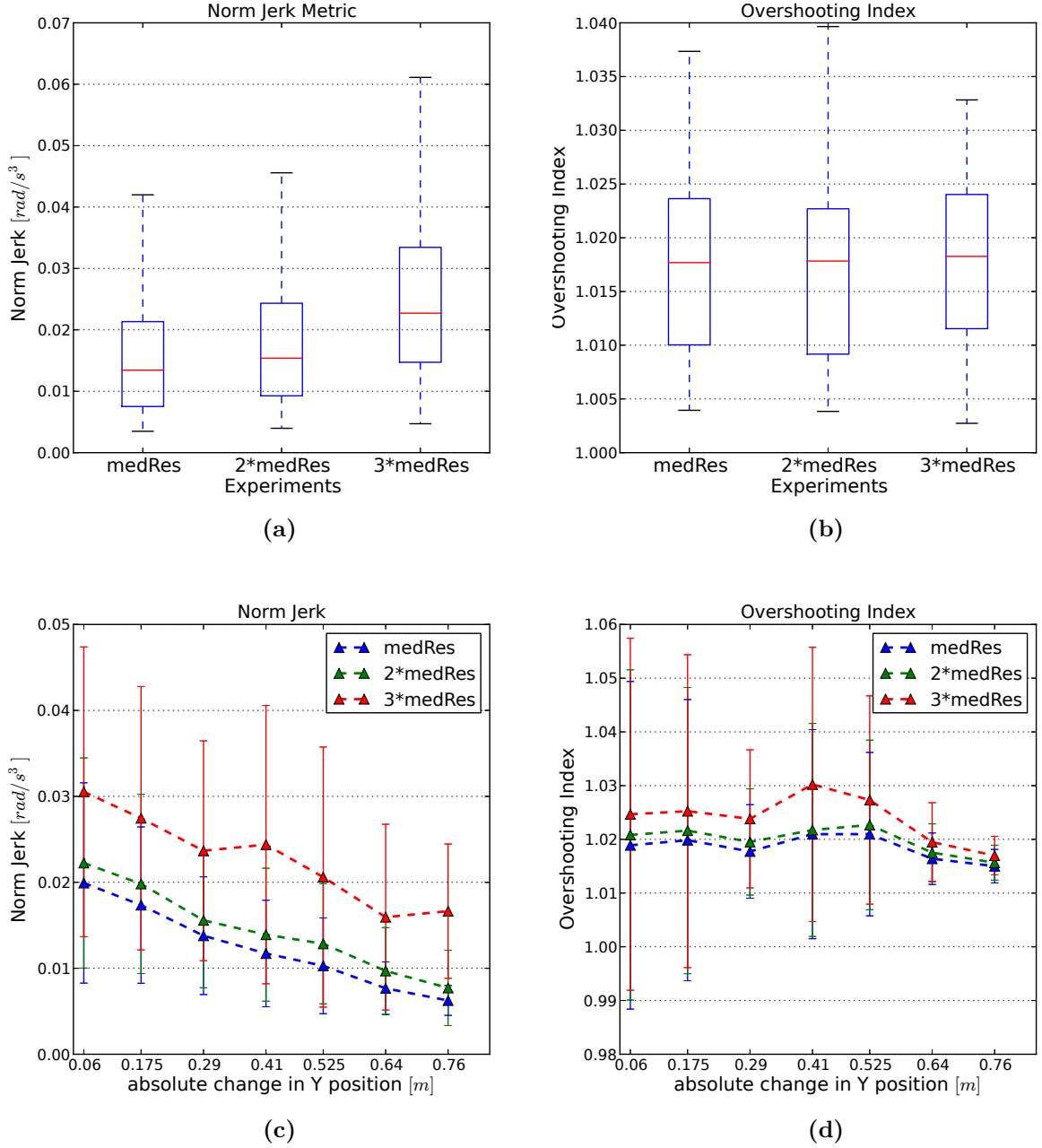


Figure 5.28: Evaluation of planned motions for three trials, where $|\phi| = 3$ and $|\mathcal{A}'| = 6$: (1) the neural map’s resolution was the median of difference between features, (2) $2\times$ median of features, and (3) $3\times$ median of difference between features in terms of (a) Smoothness of planned trajectories based on norm jerk metric across three trials. (b) Smoothness of planned trajectories based on overshooting index across three trials. (c) Smoothness of planned trajectories based on norm jerk metric across absolute difference in y position of end-effector in the robot’s coordinate system. (d) Smoothness of planned trajectories based on overshooting index across absolute difference in y position of the end-effector in the robot’s coordinate system.

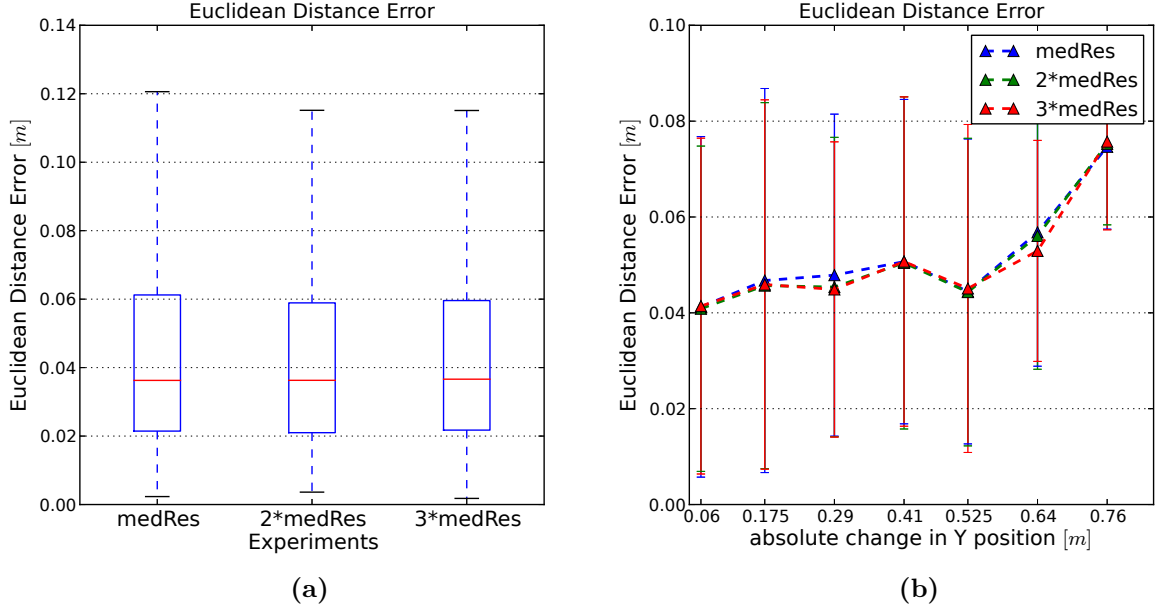


Figure 5.29: Evaluation of planned motions for four trials, where $|\phi| = 3$ and $|\mathcal{A}'| = 6$ are for the three trials: (1) the neural map's resolution was median of features (2) the neural map's resolution was $2\times$ median of features. (3) the neural map's resolution was $3\times$ median of features in terms of (a) Euclidean distance error of end effector position of planned trajectories across three trials. (b) Euclidean distance error of end effector position of planned trajectories across the absolute change in y position in the robot's coordinate system.

this technique is that while some neurons respond to points in the motor-sensory space some other neurons don't fire for any motor-sensory points and essentially were ineffective. So we decided not to use the self-organizing map technique in our model, although there is strong evidence of self organization in the brain, mostly due to the existence of various somatosensory, tonotopic, and retinotopic maps.

One important facet of our model is choosing the right design for the neural map and its resolution. We believe there is a trade-off between the resolution of the neural map and the required time for motor exploration. Imposing a fine resolution for a neural map requires more neurons to fill up the motor-sensory space while a coarse resolution demands fewer neurons. In a map with coarse resolution, neurons are more general and cannot represent the fine details of the reaching because multiple motor-sensory points are mapped to one neuron. On the other hand, these neurons tend to respond to a larger area of the motor-sensory space, and thus the planned motions can become less accurate. Conversely, a map with a fine enough resolution to represent more details of motor-sensory space demands more exploration effort, but it produces more accurate movements at the same time.

Another challenging aspect of our model is to create neurons such that they are uniformly responsible for representing the reduced space, because the presence of any ineffective neurons would make the computation more expensive in our model by making it harder to search for a firing neuron. One possible solution to the problem with having useless neurons is to prune the neural map; this parallels with the process of apoptosis in infants, which is known to be responsible for the death of overproduced neurons. The pruning process of the neural map could therefore mimic the apoptosis process.

Another important factor in our model is the trade-off between the amount of babbling needed and generalization that should be placed into the neural map. It is clear that more motor babbling allows an agent to explore the motor-sensory space thoroughly, but what is the right amount of babbling, and how much of this space do infants search through motor exploration? We believe that it is not possible for an

agent to explore the entire motor-sensory space, which can be vast in real life. Hence, the bundle width in our model is an attempt to generalize movements without the need to explore the entire motor-sensory space. The notion of fuzziness or bundle formation gives an agent a chance to generalize and avoid longer motor exploration. We are not entirely sure how infants generalize from one motion to other feasible motions at an unconscious level where the other motions are as acute and accurate as the one seen. In our model, the concept of wider bundles tries to compensate for the motor babbling by impacting more areas of the neural map and potentially creating similar possible trajectories during learning. In our model, the bundle formation could be improved to address the generalization problem in a more fruitful way to reduce the amount of learning required without losing the correctness of the learned motions.

We claim that our model is inspired by infant development, but not all the details of our model are the results of that development, as multiple passes through the motor babbling data to create neurons is not neurally possible. In our defense, we are trying to mimic the effects of evolution at the same time as the results of development. An infant is born with a brain which has not solely emerged from development, and the structures of the brain have more or less evolved to be the one infants are born with. In our model, the bundle formation or adding of connections into the neural maps using the motor babbling is the stage we claim to be the result of such development. Also, the model presented here is focused on a single developmental stage, while it could potentially be expanded to more stages, for example, by imposing more structures on the motor-babbling stage.

In our proposed bundle formation, connections in the center of bundles weighed more than the synapses on the side, or in other words, the synapses' weight decrease linearly from the center to the edges. In the experiments section, we offered some modifications to the bundle formation which target either the connections' weight or the number of the links in the bundles. The first change was to set up uniform connections with constant weight throughout the bundles. The second change was

to create a less dense bundle, so we ranked the firing neurons at each time (based on the firing rate) and then created connections between the neurons using those ranks. We observed that these changes have made the smoothness of planned motions even worse. This suggests that the presence of more links within a bundle or non-uniform connection gives the diffusion in our model a better chance of finding an optimal path.

The spreading activation responsible for motion planning in the brain requires a careful choice of both the activation and decay rate because a wrong combination of these two rates could negatively affect the success of this process. A significant activation rate would cause the neural map to saturate, resulting in an absence of gradient in the neural map. Without any gradient from a goal to a start, the path planning process will be lost. On the other hand, a significant dissipation rate causes the activation from the goal to die even before it reaches the start. It is worthwhile to look at any diffusion algorithms that don't require this adjustment, for example, a diffusion type technique where the source is not responsible for producing the gradient. Another problem with this type of path planning is that the best parameters for long trajectories might not even work for the short ones. So, the parameter choice demands careful investigation with a variety of lengths in the motion planning.

In this dissertation, in our implementation of path planning and execution, one neuron represents the start and the goal rather than a group of neurons. We don't claim that individual neurons represent the motor-sensory points. In reality, a population of neurons should fire to create a motor-sensory state. For the sake of simplicity, however, we assumed that one neuron is responsible for each motor-sensory state of planned motions. We anticipate expanding this implementation to multiple neurons for the future work.

In the real world, real-time autonomous robots expect to learn from novel circumstances; this makes it important to design an online learning mechanism. For our model to learn in an online fashion, we must make the representation of the motor-sensory space independent of the motor babbling stage. This way, we can use the online motor babbling to update the synapses in the neural maps.

Interestingly, we found that the autoencoder doesn't reproduce the end state of babbling trajectories well, meaning that in the motor-sensory space, states with a velocity of zero are not as well represented as the others. This issue made it difficult for an autoencoder to represent these states correctly from the original space in the bottleneck. It would be interesting to see how long an infant is holding its arm in a non-moving position where the velocity is zero when it reaches for an object.

To summarize, experiments in this chapter have provided a platform to investigate some critical aspects of our neurally plausible reaching model which is inspired by infant development. The model was based on motor-babbling.

Chapter 6

Conclusion

In this dissertation, we presented an embodied, developmental, and neurally-plausible reaching model inspired by motor-sensory interaction of an infant and its environment. At the core of this model, three neural maps represent the same motor-sensory space and play different roles in the arm motion planning. The motion planning occurs through the collaboration of these three neural maps representing the trajectory bundles by means of spreading activation from a goal state.

Before we could test this model in a standard condition of reaching, implementation of separate pieces of this model was examined. We need to emphasize that our conceptual model can be implemented in different ways and that the suitable approach can only be decided through trials. We conducted some experiments to find the right implementation of motor babbling, the neural representation of motor-sensory space, dimension reduction, path planning, and path execution.

We compared multiple motor-babbling strategies to identify an approach that satisfies both the limitations we face in using the robot and the requirements of our model. The adopted motor babbling approach had to be safe for the robot in our experiments. The robot we used for our experiments was not reliable and that made the process of motor exploration and testing very tedious. It would be more appropriate to develop a simulation of a robotic arm, hand, and a camera with a

physical engine software simulator– for example, Open Dynamic Engine (ODE)– to replace the actual hardware. However, it is not straightforward to simulate torque and force sensors for a simulated arm. One solution to this limitation is to mimic proprioceptive sensors similar to the approaches used in [Lee et al. \(2007a\)](#).

Through these experiments, we also evaluated multiple approaches for dimension reduction in our model and finally adopted the autoencoder, a nonlinear technique that satisfies our design requirements. We also learned from the experiments that not any sensors would help our model, or in other words, only the sensors that reasonably correlated with reaching motion parameters are helpful. Including non-correlated sensors creates an unnecessarily large motor-sensory space and even an autoencoder technique can not discover the essential features of the arm dynamics. We then compared various ways for the neural map representation to be built. From multiple different approaches, we adopted one that leads to a high-resolution map and tends to produce smooth motion trajectory with activating different neurons for different motor-sensory points.

To show that this model is computationally feasible, we tested it in two simple reaching tasks using a humanoid robot. The model used motor babbling to acquire successfully smooth and precise reaching behavior. We then investigated various aspects of this model through multiple experiments by isolating features. In each experiment, one feature or parameter of the model was varied while the rest of characteristics were fixed. We compared the learned reaching motions under different dimensionalities of the reduced motor-sensory space. As we expected, a smaller dimension leads to less smooth and accurate movements; nevertheless the relationship between the dimension of reduced space and the accuracy of the motions is not linear. We also observed that the model has not managed to learn the poses in the right side of the arc due to insufficient babbling in that direction. To calculate the norm jerk, we must measure actual time of each sub-movement of a planned motion during the execution on our robot. However, it was difficult to measure the actual time because,

occasionally, one or two joints in the arm would not move to the right positions during the test without any specific reason.

In two different experiments, we studied the effect of bundle width on quality of arm motions since a larger bundle width is more biologically plausible and may allow the model to generalize better. The results confirmed that the larger bundles lead to a smaller error of end-effector position for unseen targets but at the same time the smoothness of planned motions was sacrificed. This double-edged effect of wide bundle suggests that there should be more investigations on the implementation of bundle formation in our model.

To check the amount of motor babbling that is necessary for the model to master this simple reaching task, we conducted an experiment with the an autoencoder with a fixed bottleneck size, trained with different training sizes. This test confirmed that the autoencoder performs better as the training size increases but, at some point, the accuracy doesn't change much when adding more babbling trajectories. Moreover, this test allowed us to start with a reasonable babbling size for a more complicated version of this experiment, reaching with multiple start points.

In an experiment with the resolution of neural maps in our model, we compared motion planning with the neural maps of different resolutions. This experiment showed that a neural map with a coarse resolution tends to produce less smooth motions compared to a neural map with a fine resolution.

We hope these findings do shed light on the design of future robotic systems where autonomous systems must cope with unknown environments with the same level of competence as the known situations. We expect the results of this dissertation to serve as foundation for designing such a robotic system.

6.1 Future Research

In this section, we make some recommendations for the future research:

- Experiments incorporating sensors feedback: in the current experiments with reaching, we didn't use feedback from the sensors to adjust the underlying neural maps when a planned motion was not correct or the final position of the arm was not close to the desired destination. These reaching errors should be incorporated to update the synapses of the neural maps. This adjustment would be necessary in the situations in which the arm could unexpectedly be pushed away during a movement or its property (e.g. shape or size) could be changed after a while.
- In the current experiments, the reaching occurs in the normal situation without the presence of any unexpected perturbation. It would be interesting to test our model with reaching in the Cartesian space with clamped joints to simulate a perturbation.
- Another unexpected perturbation that can be introduced to the system is to modify the dynamics of the arm. For example, in one experiment, the system should be trained under the normal condition and then an object can be strapped around the robot's arm to modify its dynamics. This new modified system should learn to reach for the same target points after an adjusting period.
- Experiment with reaching in the Cartesian space using tools: another modification to the arm's property is to attach a tool to the robot's hand (e.g. extend the arm). A tool that is attached to the arm subsequently changes the proprioceptive sensors or the location of the end effector.
- In the current experiments with our model, the cameras were not used but it would be interesting to incorporate vision (e.g. cameras) in our system to find the end effector's location in the Cartesian space. This location can be

translated to the image of completion or the motor-sensory representation that starts path planning and execution in our model. The creation of the image of completion was outside the scope of this dissertation, and the process of learning this image was not included in this model.

- Experiments with motor babbling strategies, especially it would be interesting to find a way to safely send motor commands to joints in terms of torque instead of position.
- Experiments with non-reachable points in the space or presence of an obstacle: during path planning and execution in our proposed model, neurons representing the goal spread activities within the neural map and those activities produce a gradient. Similarly, neurons that represent the location of an obstacle could exert inhibitory activities or repulsive forces; therefore at each step, the path planning algorithm should determine the direction of motions by including those repulsive forces from the obstacle.
- Online learning instead of offline learning: in the current model, the neural maps are created using the training trajectories resulting from the motor exploration stage. For our model to be able to learn in an online style, the representation of the motor-sensory space or neural map creation must be independent of the motor babbling phase. Given a neural map before this phase, we can use the motor babbling trajectories in an online fashion to update the synapses of the neural maps.
- Modifying implementation of path planning and path execution: for the sake of simplicity in the current implementation of our model, we assumed one neuron is responsible for each motor-sensory state throughout the planned motions. However, it is more neurally plausible for a population of neurons to fire and represent a motor-sensory state rather than a single neuron.

- It is worthwhile to perform random or structured motor babbling with a simulated arm instead of using an actual robot. The random joint exploration involves simultaneously changing all the desired joints. In contrast, a structured motor babbling focuses on exploration of one joint at a time. It has been observed that an assembly phase, in which motor-sensory processes are generated and combined, precedes a tuning phase in which they are refined to accomplish a task (Lungarella and Berthouze, 2004; Goldfield, 1995). Also, it has been observed that degrees of freedom may be frozen or freed in successive stages to accelerate learning (Lungarella and Berthouze, 2004). For example, fixing joint angles or phase relations among movements may allow learning gross motor skills, which can then be refined or elaborated upon by freeing these constraints.
- Including proprioceptive sensors: information about the location of the arm could be utilized to mimic proprioceptive sensors. The calculated proprioceptive feedback from the raw sensors of the robot would be an interesting start. The underlying biological processes of proprioceptive feedback are not entirely known and the simulation is just an attempt to incorporate these sensors in developmental robotics. For example, Lee et al. (2007a) calculated these sensors in multiple different ways.

Bibliography

- Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., and Yoshida, C. (2009). Cognitive developmental robotics: A survey. *IEEE Transaction on Autonomous Mental Development*, 1(1). [14](#)
- Asuni, G., Teti, G., and et al. (2005). A bio-inspired sensory-motor neural model for neuro-robotic manipulation platform. In Asuni2005, editor, *International Conference on Advanced Robotics, ICAR' 05. Proceedings., 12th.* [18](#)
- Bernstein, N. (1967). *The Co-ordination and Regulation of Movements*. London: Pergamon Press. [15](#)
- Berthouze, L. and Lungarella, M. (2004). Motor skill acquisition under environmental perturbations: on the necessity of alternate freezing and freeing of degrees of freedom. *ADAPTIVE BEHAVIOR*, 12:47–64. [17](#)
- Bouganis, A. and Shanahan, M. (2010). Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity. In *WCCI 2010 IEEE World Congress on Computational Intelligence.* [xiii](#), [25](#), [26](#)
- Brooks, R. A. (1997). Intelligence without representation. In Haugeland, J., editor, *Mind Design II: Philosophy, Psychology, Artificial Intelligence, Rev. and Enl. Ed.* Cambridge: MIT Press. [3](#)
- Caligiore, D., Ferrauto, T., Parisi, D., Accornero, N., Capozza, M., and Baldassarre, G. (2008). Using motor babbling and hebb rules for modeling the development of reaching with obstacles and grasping. In *International Conference on Cognitive Systems.* [xiii](#), [22](#), [23](#)

- Caligiore, D., Parisi, D., and Baldassarre, G. (2014). Integrating reinforcement learning, equilibrium points and minimum variance to understand the development of reaching: A computational model. *Psychological Review*, 121(3):389–421. [30](#)
- Demiris, Y. and Dearden, A. (2005). From motor babbling to hierarchical learning by imitation: a robot developmental pathway. [19](#)
- DeWolf, T. and Eliasmith, C. (2011). The neural optimal control hierarchy for motor control. *J. Neural Eng.*, 8. [xiii](#), [28](#), [29](#)
- Earland, K., Lee, M., Shaw, P., and Law, J. (2014). Overlapping structures in sensory-motor mappings. *PLoS ONE*. [30](#)
- Gabbard, C. P. (2004). *Lifelong Motor Development*. Pearson. [xiii](#), [6](#), [7](#), [8](#), [10](#), [11](#), [12](#)
- Gallahue, D. L. (1982). *Understanding Motor Development In Children*. John Wiley and Sons. [xii](#), [8](#), [9](#), [10](#)
- Georgopoulos, A. P. (1996). On the translation of directional motor cortical commands to activation of muscles via spinal interneuronal systems. *Cogn. Brain Res.*, 3:151–155. [4](#)
- Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin. [2](#)
- Goldfield, E. C. (1995). *Emergent Forms: Origins and Early Development of Human Action and Perception*. Oxford University Press: New York. [129](#)
- Gomez, G. and Hotz, P. E. (2004). Investigation on the robustness of an evolved learning mechanism for a robot arm. In *Proc. of the 8th Int. Conf. on Intelligent Autonomous Systems*, pages 818–827. [17](#)
- Gomez, G., Lungarella, M., Hotz, P. E., Matsushita, K., and Pfeifer, R. (2004). Simulating development in a real robot: on the concurrent increase of sensory,

- motor, and neural complexity. In in Robotic Systems, M. C. D., editor, *Fourth International Workshop on Epigenetic Robotics*. [17](#)
- Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *SCIENCE*, VOL 313. [48](#)
- Husle, M., McBird, S., and J. Law, M. L. (2010). Integration of active vision and reaching from a developmental robotics perspective. *IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT*, 2:355–367. [25](#)
- Husle, M., McBird, S., and Lee, M. (2011). Developmental robotics architecture for active vision and reaching. In *IEEE International Conference on Development and Learning (ICDL)*. [27](#), [30](#)
- Iida, F., Pfeifer, R., Steels, L., and Kuniyoshi, Y. (2004). *Embodied Artificial Intelligence*. Berlin: Springer–Verlag. [2](#)
- Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems*, pages 1523–1530. MIT Press. [28](#)
- Josh Bongard, Victor Zykov, H. L. (2006). Resilient machines through continuous self-modeling. *SCIENCE*, 314. [19](#)
- Kohonen, T. (2001). *Self-Organizing Maps*. Springer Series in Informations Science, third edition. [50](#)
- Kraft, D., Derty, R., and et al. (2010). Development of object and grasping knowledge by robot exploration. *IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT*, 2:368–383. [27](#)
- Kuniyoshi, Y., Yorozu, Y., Ohmura, Y., Terada, K., Otani, T., Nagakubo, A., and Yamamoto, T. (2004). From humanoid embodiment to theory of mind. In Iida, F.,

- Pfeifer, R., Steels, L., and Kuniyoshi, Y., editors, *Embodied Artificial Intelligence*, pages 202–218. Berlin: Springer–Verlag. [2](#), [3](#)
- Kuniyoshi, Y., Yorozu, Y., Ohmura, Y., and et. al. (2004). From humanoid embodiment to theory of mind. In Iida, F., R. Pfeifer, L. S., and Kuniyoshi, Y., editors, *Embodied Artificial Intelligence*, pages 202–218. Berlin: Springer–Verlag. [18](#)
- Laschi, C., Asuni, G., Guglielmelli, E., Teti, G., Johansson, R., Konosu, H., Wasik, Z., Carrozza, M. C., and Dario, P. (2008). A bio-inspired predictive sensory-motor coordination scheme for robot reaching and preshaping. *Auton Robot*, 25:85–101. [23](#)
- Law, J., Lee, M., Hülse, M., and Tomassetti, A. (2011a). The infant development timeline and its application to robot shaping. *ADAPTIVE BEHAVIOR*, 19:335–358. [xiii](#), [1](#), [21](#), [22](#), [30](#)
- Law, J., Lee, M., Husle, M., and Shaw, P. (2011b). Infants and icubs: applying developmental psychology to robot shaping. In IEEE, editor, *The European Future Technologies Conference and Exhibition*. [30](#)
- Law, J., Shaw, P., Earland, K., Sheldon, M., and Lee, M. (2014a). A psychology based approach for longitudinal development in cognitive robotics. *Front. Neurorobot*. [30](#)
- Law, J., Shaw, P., Lee, M., and Sheldon, M. (2014b). From saccades to grasping: A model of coordinated reaching through simulated development on a humanoid robot. *IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT*, 6(2). [29](#), [30](#)
- Lee, M. and Meng, Q. (2005). Psychologically inspired sensory-motor development in early robot learning. *Int. Advanced Robotic Systems*. [20](#)
- Lee, M. H. (2011). Intrinsic activity: from motor babbling to play. In *IEEE International Conference on Development and Learning (ICDL)*, volume 2. [28](#)

- Lee, M. H., Meng, Q., and Chao, F. (2007a). Staged competence learning in developmental robotics. *Adaptive Behavior*, 15:241–255. [20](#), [21](#), [125](#), [129](#)
- Lee, M. H., Mengb, Q., and Chaoa, F. (2007b). Developmental learning for autonomous robots. *Robotics and Autonomous Systems*, 55(9):750–759. [20](#)
- Lungarella, M. and Berthouze, L. (2002a). On the interplay between morphological, neural and environmental dynamics: A robotic case-study. *Adaptive Behavior*, 10(3/4):223–241. [3](#)
- Lungarella, M. and Berthouze, L. (2002b). On the interplay between morphological, neural and environmental dynamics: a robotic case-study. *ADAPTIVE BEHAVIOR*. [15](#)
- Lungarella, M. and Berthouze, L. (2004). Robot bouncing: On the synergy between neural and body-environment dynamics. In Iida, F., Pfeifer, R., Steels, L., and Kuniyoshi, Y., editors, *Embodied Artificial Intelligence*, pages 86–97. Berlin: Springer-Verlag. [3](#), [129](#)
- Lungarella, M. and Gomez, G. (2009). *Encyclopedia of Artificial Intelligence*, chapter Developmental Robotics. Information SCI. [3](#)
- Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *CONNECTION SCIENCE*, 15:151–190. [xii](#), [3](#), [14](#), [16](#)
- MacLennan, B. J. (1997). Field computation in motor control. In *P. G. Morasso and V. Sanguineti, editors, Self-Organization, Computational Maps and Motor Control*, pages 37–73. Elsevier. [4](#)
- MacLennan, B. J. (2009). Field computation in natural and artificial intelligence. In *Robert A. Meyers et al., editor, Encyclopedia of Complexity and System Science*, pages 3334–3360. Springer. [4](#)

- MacLennan, B. J. (2011). Bodies — both informed and transformed: Embodied computation and information processing. In Dodig-Crnkovic, G. and Burgin, M., editors, *World Scientific Series in Information Studies*, volume 2, pages 225–253. Singapore: World Scientific Publishing. [3](#)
- Mahoor, Z., MacLennan, B., and McBride, A. (2016). Neurally plausible motor babbling in robot reaching. In *The Sixth Joint IEEE International Conference on Developmental Learning and Epigenetic Robotics*. [86](#)
- Meltzoff, A. N. and Moore, M. K. (1997). Explaining facial imitation: Explaining facial imitation: A theoretical model. *Early Development and Parenting*, 16:179–192. [2](#)
- Nagai, Y., Hosoda, K., Morita, S., and Asada, M. (2003). A constructive model for the development of joint attention. *CONNECTION SCIENCE*, 15(4):211–229. [15](#)
- Neville, D. B. . H. J. (2002). Cross-modal plasticity: where and how? *Nature Reviews Neuroscience*, 3:443–452. [4](#)
- Oja, E. (1982). A simplified neural model as a principal component analyzer. *J. Math. Biology*, 15:267–273. [46](#)
- O’Reilly, R. C., Munakata, Y., Frank, M. J., and Hazy, T. E. (2012). *Computational Cognitive Neuroscience*. Wiki Book, 1 edition. [48](#)
- Oztop, E. and Arbib, M. (2001). A biologically inspired learning to grasp system. In in Medicine, E. and Society, B., editors, *Proceedings of othe 23th Annual International Conference of the IEEE*. [17](#)
- Oztop, E., Bradley, N. S., and Arbib, M. (2004). Infant grasp learning: a computational model. *Exp Brain Res*, 158:480–503. [17](#)
- Pfeifer, R. and Iida, F. (2004). Embodied artificial intelligence: Trends and challenges. *F. Iida et al. (Eds.): Embodied Artificial Intelligence*, pages 1–26. [2](#)

- Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *Robotics*, 318. [2](#)
- Piek, J. P. (2006). *Infant Motor Development*. Human Kinetics. [2](#), [13](#)
- Rolf, M., Steil, J. J., and Gienger, M. (2010a). Goal babbling permits direct learning of inverse kinematics. In *IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT*. [28](#)
- Rolf, M., Steil, J. J., and Gienger, M. (2010b). Mastering growth while bootstrapping sensorimotor coordination. In *Int. Conf. on Epigenetic Robotics*. [28](#)
- Rolf, M., Steil, J. J., and Gienger, M. (2011). Online goal babbling for rapid bootstrapping of inverse models in high dimensions. In *IEEE Int. Conf. Development and Learning and on Epigenetic Robotics*. [28](#)
- Saegusa, R., Metta, G., and Sandini, G. (2009a). Active learning for multiple sensorimotor coordination based on state confidence. In *IEEE/RSJ international conference on Intelligent robots and systems*. [24](#)
- Saegusa, R., Metta, G., and Sandini, G. (2010). Own body perception based on visuomotor correlation. In IEEE, editor, *IEEE/RSJ International Conference on Intelligent Robots and Systems*. [24](#)
- Saegusa, R., Metta, G., and Sandini, G. (2012). Body definition based on visuomotor correlation. *IEEE Transaction on Industrial Electronics*, 59(8):3199–3210. [25](#)
- Saegusa, R., Metta, G., Sandini, G., and Sakka, S. (2009b). Active learning for sensorimotor coordinations of autonomous robots. In IEEE, editor, *2nd Conference on Human System Interactions. HSI '09*. [24](#)
- Saegusa, R., Sakka, S., Metta, G., and Sandini, G. (2008a). Autonomous learning evaluation toward active motor babbling. In IEEE, editor, *2008 IEEE/RSJ*

- International Conference on Intelligent Robots and Systems (IROS2008) Workshop: From motor to interaction learning in robots.* [23](#), [24](#)
- Saegusa, R., Sakka, S., Metta, G., and Sandini, G. (2008b). Sensory prediction learning – how to model the self and the environment –. In *12th IMEKO TC1 and TC7 Joint Symposium on Man Science and Measurement*. [24](#)
- Sausser, E. L. and Billard, A. G. (2006). Dynamic updating of distributed neural representations using forward models. *Biol. Cybern.*, 95:567–588. [27](#)
- Schaal, S., Mohajerian, P., and Ijspeert, A. (2007). Dynamics systems vs. optimal control-a unifying view. *Progress in Brain Research*, 165:425–445. [27](#)
- Shlens, J. (2014). A tutorial on principal component analysis. *CoRR*, abs/1404.1100. [45](#)
- Smitsman, A. W. and Corbetta, D. (2010). *The Wiley-Blackwell Handbook of Infant Development*, volume 1. Chichester, West Sussex, UK: Wiley- Blackwell Ltd, 2th edition. [11](#)
- Snow, C. W. (1989). *Infant Development*. Prentic Hall. [7](#), [8](#), [10](#), [12](#), [13](#)
- Sporns, O. and Pegors, T. K. (2004). Information-theoretical aspects of embodied artificial intelligence. In Iida, F., Pfeifer, R., Steels, L., and Kuniyoshi, Y., editors, *Embodied Artificial Intelligence*, pages 74–85. Berlin: Springer-Verlag. [2](#)
- Steels, L. (2004). The autotelic principle. *Embodied Artificial Intelligence*, pages 231–242. [18](#)
- Sucan, I. A. and Chitta, S. (2014). *MoveIt!* [66](#), [86](#)
- Thill, S. and Ziemke, T. (2010). Learning new motion primitives in the mirror neuron system: A self-organising computational model. In *11th International Conference on Simulation of Adaptive Behavior, SAB 2010, Paris - Clos Lucé, France*, volume 6226, pages 413–423. [27](#)

Vihman, M. M., Macken, M. A., Miller, R., Simmons, H., and Miller, J. (1985). From babbling to speech: A re-assessment of the continuity issue. *Language*, 61(2):397–445. [2](#)

Vita

Zahra Mahoor earned her B.S. degree with honors in 2005 and M.S. degree in computer engineering in 2008 from the Isfahan University of Technology in Iran. Before attending the University of Tennessee, she taught computer science to undergraduate students at Azad University in Iran for two years.

Zahra completed her Ph.D. degree in computer science in 2016 from the University of Tennessee where she worked as a graduate teaching assistant in the Department of Electrical Engineering and Computer Science. During her time at the University of Tennessee, she co-founded a student organization called Systers: Women in EECS @ UTK and served as the outreach chair of Systers in 2013. She received two travel scholarships from Anita Borg Institue to attend Grace Hopper Conference, one travel award from Google to attend Google I/O, and one travel grant from Graduate Senate Student to present her research at an international conference. Her research interests include developmental robotics, machine and human learning, and neuroscience.